Towards a Better Understanding of Agent-based Airport Terminal Operations Using Surrogate Modeling

Master of Science Thesis

Benjamin C.D. De Bosscher

TU Delft

Towards a Better Understanding of Agent-based Airport Terminal Operations Using Surrogate Modeling

Master of Science Thesis

by

Benjamin C.D. De Bosscher

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday January 9, 2023 at 2:00 PM.

Student number:	4429885	
Thesis committee:	Dr. B.F. Lopes Dos Santos	TU Delft, Chair
	Dr. O.A. Sharpans'kykh	TU Delft, Supervisor
	Dr.ir. E. van Kampen	TU Delft, Examiner

Acknowledgments

This thesis was the final challenge on my way to obtain a Master of Science in Aerospace Engineering at the Delft University of Technology. It marks the end of my time as a student and leads to future endeavors, whatever they may be. While I feel it is time for the next steps, I deeply cherish the experiences and opportunities of the past seven and a half years. Some highlights include my exchange to the Nanyang Technological University in Singapore, an Internship at TUI, the intermediate year at the Rotterdam School of Management, but above all, the valuable skills and further ambitions I gained during my journey in Delft. It has shaped me into the person I am today, for which I am most grateful.

Close support is one of the key ingredients to make a thesis successful, so therefore a sincere thank you to Dr. Alexei Sharpans'kykh, my supervisor. Despite inconceivable circumstances, he was always there to guide me into the right direction, to be critical of my work, and to provide constructive feedback. He has taken the quality of the research to a substantially higher level. Not only did this enhance the learning process, but it also created a pleasant and open working environment. His guidance is much appreciated; I wish him all the best and lots of strength. Furthermore, I also want to express my gratitude to senior researcher Seyed Sahand Mohammadi Ziabari, who co-supervised me along the way. I enjoyed our fruitful meetings, which helped me a lot. I wish him every success in his further career. Finally, it is always nice to be surrounded by fellow students who are going through the same thing. Therefore, many thanks to former and current members of the AATOM research group, in particular Benyamin, Didier, Klemens, Adin, Antonio, Misha, and Matthijs.

Besides academic support, it would not have been possible without family. On the one hand, I acknowledge my parents, to whom this work is dedicated. They facilitated the practical arrangements and encouraged me to realize my ambitions. I am eternally grateful for their devotion. On the other hand, there are also my brothers. The physical distance between us may not always be small, but I admire their aura of optimism and fortitude. I wholeheartedly treasure the moments we spend together. Lastly, my father has been the most loyal proofreader. He has always helped me with linguistics, despite these kinds of technical subjects are not really among his interests. I appreciate his assistance.

Over the years, and especially during this thesis, I was fortunate to be surrounded by the best people. They endured the — perhaps exaggerated — complaining about how tough everything was, but more importantly, they were always there for me. In particular, I want to thank my girlfriend, roommates, and friends for being the anchors in my life. I am looking forward to what the future might bring to us.

Benjamin C.D. De Bosscher
Gavere–Asper, December 2022

The image on the front page is taken from Meyer [69].

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AATOM	Agent-based airport terminal operations model.
ABM	Agent-based modeling.
ALE	Accumulated local effects.
BO	Bayesian optimization.
CI	Check-in.
CI	Confidence interval.
COVID-19	Coronavirus disease 2019.
CV	Cross-validation.
DOE	Design of experiments.
EIGF	Expected improvement for global fit.
ETD	Explosive trace detector.
GB	Gradient boosting regression.
GP	Gaussian Process.
GUI	Graphical user interface.
IATA	International Air Transport Association.
ICAO	International Civil Aviation Organization.
ICE	Individual conditional expectation.
KPI	Key performance indicator.
LASSO	Least absolute shrinkage and selection operator.
LHS	Latin hypercube sampling.
LIME	Local interpretable model-agnostic explanations.
LR	Linear regression.
MAE	Mean absolute error.
MAPE	Mean absolute percentage error.
MARS	Multivariate adaptive regression splines.
MEPE	Maximizing expected prediction error.
MIPT	Monte Carlo-intersite-proj-th.
MSE	Mean squared error.
N/A	Not applicable.
NN	Neural networks.
OAT	One-at-a-time.
OLS	Ordinary least squares.
PDP	Partial dependence plot.
QBC	Query-by-committee.

RBF	Radial basis functions.
ReLU	Rectified linear unit.
RF	Random forests.
RMSE	Root-mean-square error.
RTHA	Rotterdam The Hague Airport.
SA	Sensitivity analysis.
SC	Security check.
SD	Standard deviation.
SHAP	Shapley additive explanations.
SVM	Support-vector machines.
SVR	Support-vector regression.
UTC	Coordinated universal time.
WP	Work package.
WTMD	Walk-through metal detector.
XAI	Explainable artificial intelligence.

List of Symbols

B	Basis function.
C	Tolerance hyperparameter in support-vector regression.
D	Parameter space.
D	Dimension.
G	Set of transparent models.
H^2	H-statistic to examine feature interactions.
I	Indicator function.
M	Number of basis functions.
R^2	Coefficient of determination.
S	Sobol sensitivity index.
S	Feature subset.
#	Number.
%	Percentage.
Δ	Difference operator.
Ω	Model complexity.
Φ	Activation function.
α	Regularization hyperparameter.
α	Balancing factor.
β	Standardized regression coefficient.
$\boldsymbol{x}^*$	Sampled input vector closest to a point in the domain.
$\boldsymbol{x}$	Input vector.
$\hat{\sigma}^2$	Variance of the surrogate.
$\hat{f}$	Surrogate function.
$\intercal$	Transpose.
λ	Regularization hyperparameter.
$\mathbb{E}$	Expected value.
$\mathbb{N}$	Set of natural numbers.
$\mathbb{R}$	Set of real numbers.
$\mathscr{C}$	Committee of surrogate models.
AF	Acquistition function.
PD	Partial dependence.
arg max	Argument of the maximum.
arg min	Argument of the minimum.
μ	Base term in support-vector machines.

ν	Smoothness hyperparameter of the Matérn kernel.
$\overline{f}$	Average sample response.
ϕ	Shapley value.
π_x	Proximity to a point x in the domain.
σ_i^2	Variance of a particular regression coefficient i.
AvgQueueTime_{CI}	Average time in queue at the check-in counters.
AvgQueueTime_{SC}	Average time in queue at the security checkpoint.
AvgTimeToGate	Average time to reach the gate.
$\text{CI}_{\text{strategy}}$	Check-in strategy.
$\text{CTG}_{\text{strategy}}$	Call-to-gate strategy.
Gate_t	Departure gate of the flight on time slot t.
IQR	Interquartile range.
$\text{MaxPaxInQueue}_{CI}$	Maximum queue size at the check-in counters.
$\text{MaxPaxInQueue}_{SC}$	Maximum queue size at the security checkpoint.
NumMissedFlights	Number of missed flights.
PaxCompleted_{CI}	Throughput at the check-in counters.
PaxCompleted_{SC}	Throughput at the security checkpoint.
Pax_t	Number of passengers of the flight on time slot t.
Q_1	Lower quartile.
Q_3	Upper quartile.
$\text{SC}_{\text{strategy}}$	Security check strategy.
TotalExpenditure	Total expenditure during non-aeronautical activities.
min	Minutes.
ε	Accepted error hyperparameter in support-vector regression.
ξ	Local explanation.
ξ	Slack variable.
c_v	Coefficient of variation.
e	Euler’s number.
e	Error.
f	Target function.
g	Transparent model.
h	Hyperparameter of a machine learning model.
i, j, k	Index.
k	Covariance function.
k	Number of decision rules from a tree ensemble.
l	Length scale hyperparameter.
m	Mean function.
m	Sample size.
n	Number of dimensions.
p	p-value.

p	Marginal probability density function.
r	Decision rule.
s	Seconds.
s	Sample standard deviation.
s	Subset of feature values.
t	Size of committee.
t	Time slot number.
w	Regression coefficients.
z	Standard score.
$\lvert \cdot \rvert$	Absolute value.
$\lvert \cdot \rvert$	Cardinality of a set.
$\lvert\lvert \cdot \rvert\rvert$	Euclidean norm.
€	Euro.

Introduction

Airport terminals are complex sociotechnical systems, where cognitive, social, technical, and organizational factors play a major role [51, 68]. Scholars have shown great interest in modeling their operations, because they are subject to stochasticity and non-trivial complexity inherent in natural human behavior [74]. However, existing models suffer from heavy computational requirements and reveal their emergent properties only after running several simulations. These are typical challenges of agent-based modeling, the principle according to which current models are usually built [60, 74]. To address this limitation, a worthy alternative is the consideration of surrogate modeling, also known as meta-modeling. A surrogate mimics model responses through so-called black-box approximation functions [11]. The benefit is that, once a meta-model is trained, it can approximate the output at a fraction of the computational effort that would otherwise be required. Moreover, it facilitates the analysis of the underlying system [28]. The purpose of our research is therefore to accurately abstract and explain the dynamics of airport terminal operations by means of computationally efficient and interpretable surrogate models, based on an existing agent-based simulation model.

To achieve the objective, we introduce a two-stage methodology to analyze such systems in a more efficient way. The first stage involves the development of faithful surrogate models, whereafter the second stage applies techniques from the emerging field of explainable artificial intelligence to these abstractions. The novelty of our methodology lies thus in the amalgamation, rather than in the respective research fields themselves. Indeed, we have explored their common ground to take advantage of synergies. A successful application reveals the properties of the focal system, which in the case of a sociotechnical system mainly concerns emergent phenomena. Apart from the scientific contributions, the outcome of this study is particularly relevant for airport and airline managers. It leads to detailed insights into terminal processes, and provides them with efficient and effective decision-making tools. This enables them to act agile and adapt quickly to changing conditions, thereby maximizing service against available resources.

This report documents all materials of the thesis; it is organized as follows. Part I contains the scientific paper — the research's end product. The objective is explored in relation to existing work, followed by a description of the proposed methodology and evidence of its effectiveness by means of two case studies. Notwithstanding, this was not possible without a prior literature review, which is presented in Part II. Finally, Part III relates to supplementary work: it consists of six appendices to support the scientific paper. The first four cover, respectively, the preparation of the investigated agent-based model, the stabilization of its responses, a precursory impression into sampled data sets, and the optimization of meta-model hyperparameters. The last two are annexes to two sections of the paper, namely the description of the agent-based model and the results.

I

Scientific Paper

Towards a Better Understanding of Agent-based Airport Terminal Operations Using Surrogate Modeling

B.C.D. De Bosscher*

Delft University of Technology, Delft, The Netherlands

Abstract

Airport terminals are complex sociotechnical systems, in which humans interact with diverse technical systems. A natural way to represent them is through agent-based modeling. However, this method has two drawbacks: it entails a heavy computational burden and the emergent properties are often difficult to analyze. The purpose of our research is therefore to accurately abstract and explain the dynamics of airport terminal operations by means of computationally efficient and interpretable surrogate models, based on an existing agent-based simulation model. We propose a methodology consisting of two stages. Stage I involves the development of faithful surrogates. A sample is collected according to an active learning strategy, upon which Gaussian process regression, higher-order polynomials, gradient boosting, and random forests are fitted. Stage II then applies state-of-the-art techniques from the emerging field of explainable artificial intelligence to these models. Both model-agnostic and model-specific methods are considered, and their results are synthesized in order to explain the emergent properties. We prove the efficacy of this approach by conducting two case studies on AATOM, an existing **A**gent-based **A**irport **T**erminal **O**perations **M**odel. The first case study examines the total expenditure on discretionary activities, such as shopping and dining. A combination of poor staffing strategies and high occupancy rates on certain flights was found to disrupt the terminal journey of passengers on subsequent flights. As a result of these knock-on phenomena, less free time is left for discretionary activities, which has a negative effect on the total expenditure. The second case study examines the throughput of security checkpoints. While throughput increases with passenger numbers, a clear point was observed where the checkpoint reaches its maximum capacity. This leads to longer queues and therefore higher waiting times. It even goes so far as to put passengers at risk of missing their flight, especially with poor staffing strategies. Altogether, we clearly observed the preservation of emergent phenomena in surrogate models, and conclude that their combination with interpretable machine learning is an effective way to explain the dynamics of complex sociotechnical systems.

Keywords: Agent-based Modeling, Surrogate Modeling, Interpretable Machine Learning, Airport Terminal

1 Introduction

In recent decades, the aviation sector has benefited from stable growth in air traffic demand. While long-term prospects have long been taken for granted, abrupt events such as a financial crisis or the outbreak of a disease have shown the vulnerability of this supposition [12]. Furthermore, a growing number of people are also becoming concerned about the environmental impact [22]. It proves that airlines should operate more agile and lean: they must react quickly to such events and adapt to the new status quo. Airports, in turn, are the infrastructural epicenter of the system, so their operations are directly affected by changes in passenger numbers. This demonstrates the need for reliable models of terminal operations. Such models would be useful to prevent chaotic events, like in the aftermath of the COVID-19 pandemic at European airports [e.g., 54, 68].

The modeling of airport terminal operations has been previously explored by numerous scholars. Pao-Yen Wu and Mengersen [53] summarized these efforts in a meta-study, wherein was concluded that agent-based simulation models are most commonly used for operational planning and design purposes. Indeed, such models are preeminent for high levels of detail without compromising the complexity and emergent properties of sociotechnical systems like an airport terminal [48]. Notwithstanding, their computational requirements are often substantial, which might become a limiting factor as the scale of the simulation increases. To address this limitation, a worthy alternative is the consideration of surrogate modeling, also known as meta-modeling. A surrogate mimics model responses through so-called black-box approximation functions [6]. Fundamentally, the principle is subject to a dichotomy between savings in computational requirements and fidelity to the original model [17]. It is presumed to be viable as long as the reduction in computation time justifies the associated lower level of accuracy [58].

*MSc Student, Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology

However, there are additional arguments. Namely, another disadvantage of agent-based modeling is that the emergent properties are only revealed a posteriori [41]. One is thus required to run several simulations to obtain some tangible information. As a result, it is rather challenging to thoroughly analyze the dynamics of a system based solely on its agent-based model. Surrogates allow for a wider range of interpretation possibilities, making them useful in this case too [17]. Traditional examples of such practices are the analysis of regression coefficients or the evaluation of feature sensitivities [e.g., 34, 8]. Yet, a promising direction is the rapidly emerging field of explainable artificial intelligence (XAI). Its aim is to interpret and explain the reasoning behind machine learning algorithms [4]. Hence, there is an overlap with the purposes of meta-modeling and so it makes sense to exploit the synergies between the two disciplines. Applying the methods of XAI to surrogates will identify which rules, variables and characteristics of airport terminal operations are most decisive, ultimately leading to a better understanding of emergence in the system.

To our knowledge, existing literature on the common ground between surrogate modeling and interpretable machine learning is rather scarce. We therefore contribute by demonstrating the further potential of meta-modeling in this direction, and in particular its application to agent-based airport terminal operations models. The ability to detect and elucidate emergent properties is of paramount importance to realize these ambitions. As such, the objective of the research can be summarized as to accurately abstract and explain the dynamics of airport terminal operations by means of computationally efficient and interpretable surrogate models, based on an existing agent-based simulation model. Apart from the scientific contributions, the outcome is mainly relevant for airport and airline managers. It leads to detailed insights into terminal processes, and provides them with efficient and effective decision-making tools. This enables them to act agile and adapt quickly to changing conditions, thereby maximizing service against available resources.

We intend to achieve the objective by proposing the following two-stage methodology. The starting point is AATOM, which is the abbreviation for agent-based airport terminal operations model. It was recently designed and calibrated by Janssen et al. [33], and has been further developed ever since. AATOM is known for its high-fidelity to the actual terminal system, although it suffers from large computational requirements. Hence, the first stage of our methodology relates to the creation of surrogate models. This includes generating a data set, training black-box functions, and validation. The process is not necessarily linear, as data collection can be combined with training surrogates — commonly known as active learning or adaptive sampling [8]. Once they reach a satisfactory level of accuracy, the second stage then uses them for the interpretation of the agent-based model. Both traditional and more advanced techniques from the field of XAI are considered. Several approaches are thus explored, although it is not the intention to implement as many as possible. Instead, we select those that ultimately yield the best insights into AATOM. The end product is therefore a concise synthesis of a multitude of results from different interpretation methods. If done right, the emergent properties of the sociotechnical system should come to light, demonstrating the relevance of this methodology. Altogether, its novelty is not situated in the individual components, but rather in their consolidation where we take advantage of the synergies.

The paper is organized as follows. It starts with compiling the theoretical background in section 2. This is done by reviewing three research dimensions, namely the modeling of airport terminal operations, surrogate modeling, and interpretable machine learning. After that, section 3 further elaborates on the methodology based on its two stages. The AATOM simulator is described next in section 4: the main principles behind the model are illustrated, along with the specific settings for the simulations. Examples include the terminal layout, airport strategies, airline time slots, and so on. Section 5 continues by presenting the results. First, the performance of the meta-models is examined, after which two examples show the strengths of synthesizing different interpretation methods. In particular, we analyze: 1) the discretionary spending behavior of passengers, and 2) how the throughput at security is affected by different scenarios in the airport terminal. Finally, the results and their implications are further discussed in section 6 and a conclusion is drawn up in section 7, along with recommendations for future work.

2 Theoretical Background

Three research fields are related to our research. This section provides the necessary groundwork by reviewing the state-of-the-art of each of these fields. Respectively, that is the modeling of airport terminal operations, surrogate modeling, and interpretable machine learning. We also discuss existing examples of their specific combination.

2.1 Modeling Airport Terminal Operations

Airport terminals are central to passenger handling. It is the place where departure, arrival, and transfer flows congregate, each of which has its own characteristics and goals [10]. In particular, we focus on the departure flow of passengers. Typical activities include the check-in, security check, border control for international destinations, and possibly some non-aeronautical activities such as shopping or dining [37]. Scholars have shown great interest in modeling these processes as they are subject to stochasticity and non-trivial complexity inherent in natural human behavior.

Tosic [69] is one of the earliest available review studies, yet it has not lost its relevance. The author identifies several ingredients of modeling airport terminal operations, the most pertinent of which are the following. First of all, the demand of air traffic is usually forecast with traditional statistics. This is important for planning purposes and thus forms the basis for rigorous decision-making. The more recent literature has further subdivided it into problems with strategic, tactical and operational horizons [e.g., 29, 46]. Secondly, one can also consider specific physical locations; examples are single check-in counters or border control. They are often modeled using queuing theory, where performance is measured by quality of service. The results can then be directly benchmarked against the International Air Transport Association expectations [28]. The models are either stochastic or deterministic, the former being closer to reality at the cost of greater complexity. Thirdly, terminal operations may be viewed from the perspective of the process itself, like security screening at the checkpoint. That allows to optimize it as a whole rather than the components individually. Processes are generally modeled in two ways: analytically or through simulation. Analytical approaches are quick, exact and not overly complicated. However, this affects their fidelity to the real world [47]. A simulation-based approach is therefore preferable if the system entails a certain degree of complexity. Lastly, the entire terminal building can be taken into account at once. In the end, individual processes influence one another and as such contribute to the overall emergent properties. Most examples in the literature are simulation-based, which makes sense as analytical approaches often fail to capture much of the complexity associated with the dynamics of sociotechnical systems. Depending on the level of detail, one can still distinguish between microscopic, mesoscopic and macroscopic models, although the former is rather the standard when considering operational flows [47].

Alternatively, the more recent meta-study of Pao-Yen Wu and Mengersen [53] differentiate existing airport terminal models according to their use case. They identified four purposes: capacity estimation, operational planning, security risk evaluation, and performance measurement. This becomes particularly interesting in combination with Tosic [69], as it reveals appropriate methods to realize our research ambitions. Notably, most models to represent the operations of an entire departure flow seem to be agent-based. That is a microscopic bottom-up approach capable of simulating the behavior of individual passengers, along with the interactions between them and the environment [53]. It became particularly relevant as computing power increased over the years, giving researchers the opportunity to create simulation models that are meticulously close to reality [45]. Hence, agent-based modeling is indeed very suitable if one requires detailed information about terminal processes, which is crucial for understanding emergent properties.

In line with the above observations and suggestions, Janssen et al. [33] have recently developed such an agent-based architecture and a simulator for airport terminal operations. We use this model to prove our methodology, so section 4 further elaborates on its working principle and usage. Nevertheless, it is rather known for its heavy computational requirements, making surrogate modeling a viable alternative.

2.2 Surrogate Modeling

Despite currently available technology, advances to understand complex systems in detail are often hampered by computational limitations. This has encouraged the development of surrogate modeling, which aims to fit black-box functions between the input and output of an expensive model in an attempt to accurately mimic its behavior [17, 6]. The concept is not new and knows several applications in engineering: Forrester et al. [17] mentions structural analysis, computational fluid dynamics, geostatistics, etc. More recently, it is also finding its way to emulate agent-based simulation models, as demonstrated in the meta-study by Pietzsch et al. [56]. The benefit is that, once a meta-model is trained, it can approximate the output at a fraction of the computational effort that would otherwise be required. Moreover, it facilitates the analysis of the underlying system. An example is Elshawi et al. [14], where they managed to gain detailed insights into relationships between several variables to assess people's risk of hypertension based on local and global surrogates.

Meta-modeling starts with a design of experiments (DOE), the purpose of which is the creation of a training sample [17]. One typically strives to maximize the amount of statistical information in the data against the number of observations. After all, the original model of the complex system usually has large computational requirements. Several strategies exist for the DOE, although literature generally distinguishes between one-shot, space-filling, and adaptive approaches [42, 21]. The latter in particular has been receiving a lot of academic attention lately, despite it being rather complicated. We describe the strategy as a process where an algorithm iteratively samples in regions from which a surrogate model benefits the most. Hence, it is also known as 'active learning' because an emulator is trained simultaneously [42]. The notion is that with the same or even a smaller sample size, adaptive approaches should outperform the other two because data points are selected in such a way as to maximize the accuracy of the meta-models.

The second degree of freedom is the architectural choice of the surrogate. In the early days, this was limited to mostly linear models. They are however still widely used, mainly because of their simplicity [72]. Later on, Gaussian processes and radial basis functions made their entrance as they proved to be more accurate, albeit at higher computational costs [72, 73]. Nowadays, research into the possibilities of machine learning is thriving. Its application to surrogate modeling has not been overlooked with support-vector machines, decision trees, ensembles, and neural networks as the most recent additions to the field [1, 58].

In line with the latest trends, we particularly focus on the possibilities of an adaptive DOE. Furthermore, multiple meta-model architectures are explored, with accuracy and interpretability as principal key drivers for the ultimate selection. To enhance the compatibility between these two drivers, techniques from the field of interpretable machine learning are also being looked at.

2.3 Interpretable Machine Learning

The traditional approach to explain the behavior of a model has been to conduct a sensitivity analysis (SA) [70]. It allows to investigate the effect of input parameters on the output. While its usefulness remains undisputed, further research in this area has been flourishing of late. This is fueled by the European Union's adaption of the General Data Protection Regulation, which requires transparency and a profound understanding of the rationale behind machine learning implementations [39, 51]. Seemingly simple questions, whether the outcome is trustworthy or not, or how the algorithms arrive at their solutions, are actually quite challenging to answer. The literature calls it explainable artificial intelligence (XAI), which even goes so far as questioning whether accuracy or explainability should be optimized [61].

The motivation for developing surrogates to understand a complex system is twofold. On the one hand, more extensive analyses can be performed as they are much faster than the original model. On the other hand, they also consist of internal functional relationships between input and output parameters, which may reveal emergent properties. Methods belonging to the former are commonly referred to as agnostic [50]. The SA is one of them, although state-of-the-art examples are dependency plots [19], examining the relevance of features [16], local interpretable model-agnostic explanations [60], and Shapley additive explanations [44]. Alternatively, methods belonging to the latter are associated with a particular architecture. That is, they are model-specific [50]. A well-known example is the interpretation of regression coefficients, for which widely accepted statistical significance tests are available to evaluate whether and which parameters truly influence a response [4]. Another possibility is to extract decision rules from tree-based ensembles, such as RuleFit [20]. This enables an IF-THEN type of reasoning and yields insights that are quite unique. Clearly, there are myriads of options which show that research into XAI is still emerging. For those new to the field, we refer to Barredo Arrieta et al. [4] for relevant definitions and taxonomies, while practical illustrations are found in Belle and Papantonis [5], Elshawi et al. [14], and Molnar [49].

Both traditional and more sophisticated interpretation methods have their strengths and weaknesses. There is no one-size-fits-all and the best approach often depends on the problem itself. Consequently, we investigate several alternatives to obtain the most relevant information from the surrogates about complex dynamics and emergence in airport terminals. The ultimate purpose is a concise synthesis with interesting additions from different perspectives.

2.4 Surrogate Modeling for the Interpretation of Agent-based Models

Research on the employment of meta-modeling for understanding agent-based models is rather scarce. An example is the work of Janssen et al. [34], in which linear regression was applied as one of the methods to examine causality between features and responses. In addition to causal graphs and Sobol sensitivity indices, regression weights proved useful in elucidating the strength and direction of relationships. Nevertheless, the authors emphasize that emergent properties are best observed when insights from different angles are combined. This also helps against spurious interpretations, as causalities may be inferred that do not exist in reality.

Another example are De Leeuw et al. [11], who implemented random forests and artificial neural networks to derive feature importances. The results clearly show which input parameters their surrogates are most reliant on. These are usually solid indicators of the drivers behind a response, although further conclusions are difficult to draw. Hence, it is a valuable addition to the overall analysis, but solely the importances will not reveal any dynamics nor emergence in agent-based models.

Lastly, ten Broeke et al. [66] proposed to split the feature space first. They use a classification algorithm to distinguish different types of model behavior from one another. Only thereafter, support-vector regression is deployed to mimic responses in each of the distinct regions. This enables them to differentiate between parts of the domain, which in turn leads to a more in-depth analysis. Specifically, sensitivity indices are calculated from the classification and regression algorithms. These results then provide insight into how an agent-based model is affected from a qualitative and quantitative perspective, respectively. The authors state that their methodology is especially powerful for responses with different behavioral modes, such as the density of a population that is either dying out or surviving. While they do indeed attain a better understanding of the heterogeneity and its drivers, it remains rather difficult to reach concrete conclusions about changing dynamics. This requires an approach that is allowed to be less discrete in its reasoning.

Despite the latest developments, it is clear that the full potential of surrogate modeling and XAI has not yet been exploited in the interpretation of agent-based models. We fill the gap by proposing a two-stage methodology in the next section. First, surrogate models are created, to which various techniques from the field of interpretable machine learning are then applied. This paves the way for detecting and visualizing emergent properties of complex systems behind agent-based models.

3 Methodology

As relevant dimensions of the research have been touched upon in the theoretical background, current section continues with the methodology. A high-level overview is depicted in Figure 1. The first step is to define and prepare the agent-based model of interest. This model is typically highly detailed and close to reality, but computationally intensive. Consequently, there are two reasons that make surrogate modeling an attractive alternative. On the one hand, it gives access to much faster models. On the other hand, they enable us to better understand the underlying system — recall that agent-based models reveal the emergent properties only a posteriori, thereby requiring numerous simulation runs [41]. These two reasons are reflected in stage I and stage II of the methodology. The former consists of sampling, fitting surrogate models, and validation. The latter is concerned with agnostic and specific analyses, after which their outcome is validated through triangulation. Finally, the results of the second stage are synthesized in order to interpret and understand the complex dynamics and emergence of the focal agent-based model. The purpose is not to make the analysis as elaborate as possible, but rather to select the results that ultimately lead to the best insight.

We now go into further detail per methodological step. That includes the theory, a selection of the most appropriate methods, and verification. The theory per individual method is only briefly discussed, as the scope of this research is mainly about their synergies. For more in-depth information, we refer the reader to the references in the tables comparing advantages and disadvantages of the respective methods. Furthermore, note that the numbers of each step in Figure 1 correspond to those of the following subsections, respectively.

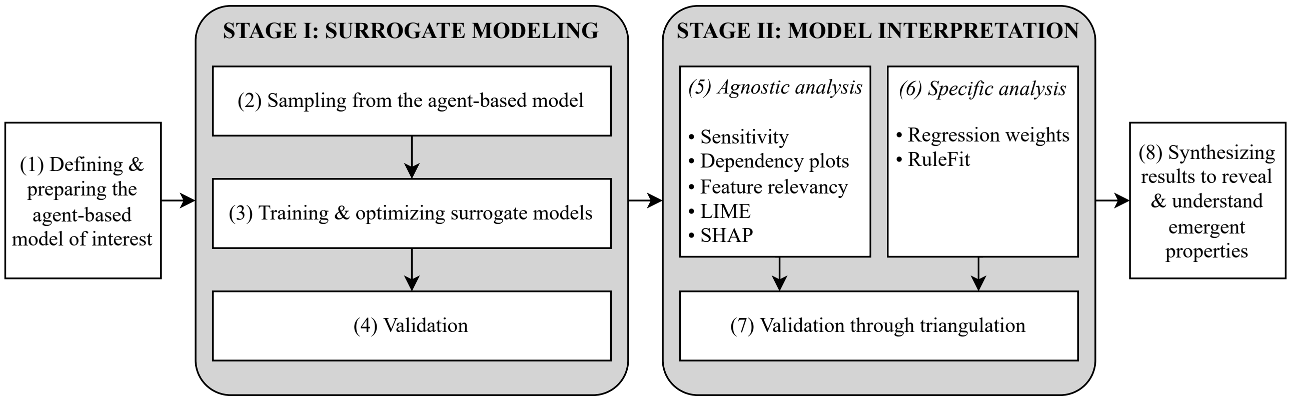

STAGE I: SURROGATE MODELING
STAGE II: MODEL INTERPRETATION
(1) Defining & preparing the agent-based model of interest
(2) Sampling from the agent-based model
(3) Training & optimizing surrogate models
(4) Validation
(5) Agnostic analysis
• Sensitivity
• Dependency plots
• Feature relevancy
• LIME
• SHAP
(6) Specific analysis
• Regression weights
• RuleFit
(7) Validation through triangulation
(8) Synthesizing results to reveal & understand emergent properties

Figure 1: High-level overview of the two-stage methodology.

3.1 Defining and Preparing Agent-based Models

Before diving into the possibilities of surrogate modeling, one must first have access to an agent-based model of the system under investigation. Our general presumption is that the model has been verified and validated, although it is strongly recommended to conduct some additional tests to see whether this is actually the case. Namely, surrogates cannot be expected to yield excellent performance if their training data is of poor quality. Hailpern and Santhanam [24] provide an overview of common practices for software testing and verification. More details can be found in Appendix A of the Supporting Work, which explains the procedure we have followed to ensure that our agent-based model functions as desired.

Next to that, sociotechnical systems are characterized by high degrees of stochasticity; a natural consequence of the human behavior to which they are subject. This results in variable model responses. Even if the input settings are exactly the same, output parameters will differ in another simulation run. It is usually considered beneficial and makes agent-based modeling a powerful paradigm to analyze such systems. Notwithstanding, surrogate models are different as they are generally deterministic in nature. In other words, they associate one input vector with one specific output vector, making it difficult for them to incorporate stochasticity. There is thus a need to stabilize responses of agent-based models before their data can be used to train surrogates. In accordance with the law of large numbers, a straightforward approach to achieve this is by averaging the results over many simulation runs without changing the input parameters [13]. The coefficient of variation can then be evaluated to determine exactly how many simulations are needed — a common approach to describe the statistical dispersion of a particular outcome [15]. We further elaborate on the concrete steps in Appendix B of the Supporting Work, together with the removal of outliers. Once the responses are deemed sufficiently stable, one can proceeded to stage I, starting with sampling from the agent-based model.

3.2 Sampling from Agent-based Models

The surrogate modeling process commences with the creation of a training data set. This is often referred to as the design of experiments (DOE), which aims to extract as much statistical information as possible from the focal agent-based model [17]. While several sampling strategies exist, it follows from the theoretical background in section 2 that adaptive designs are state-of-the-art. We therefore focus in particular on such approaches. Once an initial sample is available, the general procedure is to iteratively evaluate a meta-model and select a new point to sample until some stopping criterion is reached [42]. There are still a few degrees of freedom, so Algorithm 1 presents an overview of the proposed strategy, of which we now discuss the consecutive steps.

Algorithm 1 Proposed adaptive sampling strategy.

```
1: Use the Hammersley sequence to generate an initial set of feature values     ▷ Size = 10 × dimensionality
2: Sample the corresponding responses from the agent-based model
3: while the stopping criterion is not reached do                              ▷ Based on meta-model accuracy
4:     Train a Gaussian process regression model
5:     Identify the next data point to sample based on the EIGF acquisition function
6:     Sample the responses corresponding to the selected data point from the agent-based model
7:     Add the data point to the training set
8: end while
```

Since active learning requires the prediction uncertainty of a surrogate, one must first train the model to evaluate its output. This in turn can only be done if an initial sample is available. In the literature, scholars often use one of the following four sampling methods: Latin hypercube sampling, orthogonal arrays, Hammersley sequences, and uniform designs [72]. They all have pros and cons, though we find the Hammersley sequence best suited to serve especially the exploration purpose of an initial sample. Wong et al. [77] describe the technical details, but in essence it is based on a mathematical formulation to generate sequences with low discrepancy while maintaining a uniform distribution in high-dimensional spaces. That also makes the method space-filling and computationally efficient, albeit only quasirandom and rather poor at covering the boundaries of the domain [65, 77]. Regarding the size of the initial sample, Loeppky et al. [43] recommend 10 times the number of input features if a Gaussian process regression model is used as surrogate [42]. This turned out to be the case in the while loop of the active learning algorithm.

The surveys of Liu et al. [42] and Fuhg et al. [21] are excellent starting points to design such a while loop. Fundamentally, it consists of two key ingredients: 1) a way to obtain the uncertainty of a meta-model, and 2) an acquisition function to translate the uncertainty into a specific data point that is most interesting to sample next. Liu et al. [42] distinguish four strategies for the first ingredient. One can use either the variance, disagreement between a committee of surrogates, cross-validation prediction error, or derivatives. The former is by far the most effective and efficient way to infer about surrogate model uncertainty over an entire domain — especially the combination with Gaussian process regression is powerful, as it naturally yields the prediction variance [42, 3]. For the second ingredient, Fuhg et al. [21] rigorously reviewed 14 promising acquisition functions. Various drivers were evaluated and based on their assessment, we chose the expected improvement for global fit (EIGF), developed by Lam [40]. Namely, it yields a reliable and solid performance for meta-modeling purposes, without increasing the theoretical complexity or computational requirements too much. Mathematically, the EIGF acquisition functions is expressed as

$$\text{for } f \text{ in } D \in \mathbb{R}^n, \quad \boldsymbol{x}_{\text{next}} = \underset{\boldsymbol{x} \in D}{\arg\max} \left[\alpha \left(\hat{\sigma}^2(\boldsymbol{x}) \right) + (1-\alpha) \left(\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x}^*) \right)^2 \right] \tag{1}$$

where f denotes the target, $\hat{f}$ the surrogate approximation, D the feature space, n its dimensionality, $\hat{\sigma}^2$ the variance of the surrogate, $\boldsymbol{x}$ the input vector, and $\boldsymbol{x}^*$ the nearest already sampled data point. We later added a balancing factor α to control the trade-off between exploration and exploitation during the sampling process. The closer to one, the more preference to exploration, and vice versa. This may change throughout the loop, e.g., to allow for a decaying strategy that gradually shifts the emphasis from exploration to exploitation.

The while loop of the active learning algorithm continues until a stopping criterion is reached. That is a predefined condition which determines when the sample ought to be sufficiently informative. Usually it is based on computational or time constraints, though one can also look at the attained meta-model accuracy [42, 21]. We opt for the latter because the computational budget for current research allows this. More specifically, the active learning algorithm continues until accuracy no longer improves by increasing the sample size.

In addition to a training set, it is also desirable to have independent validation and test sets for the hyperparameter optimization and model assessment, respectively. Common approaches include splitting the training set beforehand or pursuing a cross-validation strategy [25]. However, this is not straightforward when the sample is gathered adaptively, as the benefits of active learning could be jeopardized. We therefore prefer to create two additional data sets. They are both randomly sampled from the agent-based model, but only those input parameter combinations that were not selected during the adaptive sampling process. This continues until the size of each equals about 20% of the entire sample; a typical proportion for validation and test sets [25, 23].

Finally, the implementation of the DOE must be verified. Explanatory testing is used to eliminate most of the bugs, although unit and integration tests are also necessary to ensure that the software behaves as expected at a deeper level [24]. After this, one can proceed with training and optimizing surrogate models.

3.3 Training and Optimizing Surrogate Models

When a data sample is available, the next step of stage I is to train and optimize surrogates. In essence, we aim to find an appropriate function $\hat{f}$ so that the prediction error e is as small as possible compared to the actual outcome of the focal agent-based model, denoted by f. This is mathematically described as

$$f(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}) + e \tag{2}$$

where $\boldsymbol{x}$ denotes an n-dimensional input vector [36]. Put differently, the purpose is to accurately mimic the response behavior of the original function f by fitting a machine learning algorithm on the sample [17].

A follow-up question is then what modeling method is most suitable to replicate the relationship between features and outcome. Various alternatives have been tried out over the years, a comparison of which is provided in Table 1. We summarized the principal advantages and disadvantages of common architectures. Expressions that linearly combine mathematical terms are referred to as linear models; specific examples are linear regression, polynomials, and multivariate adaptive regression splines [6, 79, 18]. They are widely popular, mainly because of their simplicity and interpretability [72]. Secondly, there are Gaussian processes and radial basis functions [57, 6]. Both use Euclidean distances to known observations, although the former builds surrogates as a stochastic process and the latter as a weighted sum of basis functions [63, 6]. They became prevalent surrogate models as scholars reported high accuracies, especially for non-linear relationships [72, 73]. Finally, more powerful machine learning algorithms are gradually gaining the upper hand, with support-vector machines, decision trees, and neural networks as the latest additions to the field [1, 58]. While there is some academic debate about the

Table 1: Comparison between common architectures for surrogate modeling.

Architecture	**Advantages**	**Disadvantages**	**Ref.**
Linear models	Simple, interpretable, can be extended to a higher-order polynomial or spline	Bounded to their functional form, less suitable for complicated problems	[72, 58, 73]
Gaussian processes	Yields the uncertainty of the prediction, high accuracy, integrated hyperparameter tuning	Computationally intensive, ill-conditioned covariance matrix, complicated mathematics	[58, 73, 79]
Radial basis functions	Captures non-linear relationships without having a too complicated construction	Difficult to interpret, basis functions and hyperparameters need to be defined, ill-conditioning	[58, 73, 17]
Support-vector machines	Flexible, robust, low risk of overfitting, performs well with small samples	Difficult to interpret, complex hyperparameters, computationally intensive	[6, 79, 25]
Decision trees	Powerful, handles different data types without much preprocessing, performs especially well in boosted/bagged ensembles	Deteriorating interpretability when used in an ensemble, prone to overfitting, responses are not smooth	[1, 23, 25]
Neural networks	Powerful, completely customizable, captures complex relationships and interactions in a high-dimensional environment	Prone to overfitting, computationally intensive, difficult to interpret, complex hyperparameters, requires a lot of data	[23, 25, 79]

accuracy of support-vector machines (e.g., a superior, comparable and inferior performance is mentioned by Wang and Shan [72], Bhosekar and Ierapetritou [6], Williams and Cremaschi [75], respectively), the dominance of tree ensembles and neural networks is not questioned [25]. The latter, however, suffers from severe constructional complexity, and also requires a large data sample and a substantial computational budget [23, 25].

Now that the salient characteristics of common architectures are clear, we move on to selecting appropriate alternatives in order to achieve the research objective. First and foremost, recall from section 3.2 that a Gaussian process regression model is central to the adaptive sampling strategy. It is therefore automatically part of our selection. Secondly, the highest accuracy is presumably attained by tree ensembles, making random forests and gradient boosting also promising candidates. Finally, we include polynomials because they are reasonably interpretable without losing the ability to capture curvilinear patterns [1]. The other methods are not taken into account; they are either too inaccurate, too complicated, or too difficult to interpret.

Next, the selected architectures should also be optimized, as the performance of machine learning algorithms can be severely affected by the choice of hyperparameters. Notwithstanding, finding the optimal combination is often a difficult and, above all, computationally demanding task [27]. Popular tuning methods are therefore a grid search, random search, Bayesian optimization, and evolutionary metaheuristics [2, 78, 80]. We prefer Bayesian optimization as it is an efficient algorithm with high chances of finding a combination close to the global optimum. Moreover, it does not require in-depth knowledge of promising candidates. Two drawbacks are its complexity and difficulty to multi-thread, although these are easily outweighed by the benefits [78, 80]. For an introduction to Bayesian optimization, we refer the reader to Shahriari et al. [64], but in essence it works as follows. An auxiliary model is fitted onto a prediction error metric of the surrogate, then the parameter combination is selected that leads to the smallest error, after which this combination is evaluated on the actual surrogate. The auxiliary model is updated with the new information and the process continues sequentially until a predefined number of iterations is arrived at [2, 78].

Lastly, the training and optimizing step must also be verified. This depends on exactly how it was implemented, but one typically employs existing packages, such as Python's scikit-learn [55] or scikit-optimize [26]. These packages are generally tested before being released, so there is no need to do this again. However, one should still perform integration tests to ensure that interfaces are properly embedded in the software [24]. Once everything is verified, the surrogate models can be validated to see how closely they resemble the behavior of the original agent-based model.

3.4 Surrogate Model Validation

Validation is important as it gives an idea about the generalization power of meta-models. Namely, it shows to what extent they remain close to the agent-based model under investigation. The out-of-sample performance is usually measured by evaluating validation metrics on an independent test set [23].

Undoubtedly, one of the most popular metrics is the coefficient of determination (R^2). It indicates the proportion of variation in a response of the agent-based model that is explained by input parameters in the surrogate [9, 15]. The R^2 is a very useful and informative metric, but it does not directly measure the prediction error. For that, one can resort to the root-mean-square error (RMSE), which yields the expected error of a surrogate [17]. Alternatively, there is also the mean absolute error (MAE). The RMSE and MAE are closely related, although an essential distinction is that the former squares the discrepancies between actual and predicted responses and takes the root of their average, instead of using the absolute value. This makes the RMSE naturally more sensitive to outliers [9]. Both the RMSE and MAE are expressed on the same scale as the evaluated output parameter. While this can be convenient, the performance of surrogate models for different responses cannot be compared when they are expressed differently. The mean absolute percentage error (MAPE) resolves the issue by calculating the prediction error relatively. One drawback, however, is that the indicator tends to exaggerate the error for responses close to zero [9].

All mentioned key performance indicators have their pros and cons. For that reason, we do not select just one metric for the overall validation, but rather compute all four. This provides a better insight into the bigger picture and ensures a stronger interpretation of the results. If the surrogates prove to be sufficiently faithful, one can proceed to the second stage of the methodology: model interpretation. It consists of two types of analyses, which could be done in parallel. We first discuss the agnostic options and then the specific ones.

3.5 Model-agnostic Analysis

When the concept of meta-modeling was introduced, we argued that it can play an important role in understanding an underlying system. Surrogates are not only much faster, but they also have internal functional relationships between the input and output parameters, unlike agent-based models [17, 41]. As a result, they enable: 1) more extensive analyses, and 2) explicit insight into their rationale. Interpretation methods that relate to the former are referred to as model-agnostic [50]. They are independent from machine learning algorithms and can therefore be applied to essentially any model. Only the effect of input parameters on the output

Table 2: Comparison between model-agnostic interpretation methods.

Method	Advantages	Disadvantages	Ref.
Sensitivity analysis	Simple and evident, gives solid indications, several implementations exist with different levels of sophistication (local and global)	Interactions are not always considered, can be computationally intensive, questions about the variance as a proxy for variability	[7, 62, 70]
Dependency plots	Intuitive and easily understood, no ambiguity, heterogeneity can be detected, straightforward implementation	Features are analyzed separately when in fact they may be correlated, difficult to consider more than two dimensions at the same time	[49, 14, 19]
Feature relevancy	Easily understood, incorporates feature interactions, provides one comprehensive overview	Perturbations can lead to unstable results, relevancy may be divided among correlated features	[49, 16]
LIME	Fully customizable, easily understood, straightforward implementation, yields an actual model rather than just insights	Unstable results: there may be large local differences, no theoretical basis for why this should work, many degrees of freedom	[49, 59]
SHAP	Based on valid theoretical arguments, provides a comprehensive overview, local and global interpretations are consistent	Computationally intensive, approximations are necessary, rather complicated, not evident to interpret the results correctly	[49, 5, 44]

is analyzed, thereby completely disregarding an algorithm's internal working principle [4]. This makes agnostic approaches flexible and gives them a large user base. Table 2 compares salient characteristics of the five most common methods [5, 49, 7]. We now briefly discuss them in more detail.

First and foremost, there is the sensitivity analysis (SA), the general purpose of which is to examine how model responses behave with respect to changes in their input parameters. Put differently, it attributes response variability to the respective features. Both local and global approaches exist, such as a one-at-a-time SA or variance-based Sobol sensitivity indices [7, 62]. While a SA shows to what extent the output of a machine learning model is affected by changing features, it says nothing about the actual form of the relationship. Such information can be obtained by analyzing the dependency of an outcome on its input parameters [5]. Typical examples are partial dependence plots by Friedman [19] and the closely related individual conditional expectation curves [14, 5]. These graphs visualize how responses behave as a function of a feature: for instance, whether the nature of a relation is (non-)monotonic, (non-)linear, and so on [49]. Thirdly, the outcome of a surrogate is expressed as a function of input variables, so one may wonder which ones are more determinative than others. In this sense, examining the relevancy of features is a useful approach to understanding them. Fisher et al. [16] proposed to calculate their importances based on perturbations. The more a meta-model relies on a certain parameter, the greater its importance, and vice versa [14, 50]. Next, there is LIME, which is short for local interpretable model-agnostic explanations. It originated from the research by Ribeiro et al. [60] and is currently one of the more popular approaches in the field of XAI. The core idea is to explain the reasoning behind black-box algorithms by means of fitting an auxiliary model that is locally faithful [5]. Logically, this model must be transparent and easy to interpret, such as linear regression or decision trees [49, 60]. The explanations at a particular point in the domain are then presumed to be in accordance with the local behavior of the machine learning algorithm. Finally, we discuss SHAP, which next to LIME is one of the more recent additions to XAI research. It stands for Shapley additive explanations and is considered an eminent interpretation technique [5]. Based on cooperative game theory, Shapley values are calculated to estimate the individual contribution of features to the prediction of a machine learning model. SHAP thus shows exactly how a response is affected by which input parameters and can be applied at both a local and global level [5, 44, 49].

While the aforementioned agnostic interpretation methods do provide useful insights into the relationships between features and responses, the rationale behind a surrogate model's outcome remains unclear [14]. For that, one should rather look at the internal reasoning, often referred to as model-specific analysis.

3.6 Model-specific Analysis

Unlike model-agnostic, model-specific approaches explain the behavior of machine learning algorithms based on their internal properties [14, 50]. Functional relationships between input and output parameters are used to interpret how responses come about. Consequently, the application of these methods is strongly limited to algorithms that provide access to such information. This is an evident drawback, but it should be borne in

Table 3: Comparison between model-specific interpretation methods.

Method	Advantages	Disadvantages	Ref.
Regression weights	Well-accepted and mature, does not require many calculations, transparent, accessible	Mediocre performance of the underlying model, multicollinearity can lead to fallacious conclusions	[7, 49, 4]
RuleFit	Combined strengths of linear regression and tree ensembles: powerful without compromising transparency, flexible, yields a model	Overlapping decision rules deteriorate the explanatory power, often results in mediocre performance as it remains a linear model	[49, 20]

mind that agnostic approaches do not explicitly reveal the rationale behind an outcome [14]. They do explain how responses are affected, albeit on the basis of intermediate steps where proxies are the rule rather than the exception. For direct interpretations, one should resort to model-specific approaches. Given our selected architectures in section 3.3, there are two methods that are deemed feasible: the analysis of polynomial regression weights and RuleFit for the tree-based ensembles. We now briefly elaborate on them, alongside a summary of salient characteristics in Table 3.

Linear regression, in addition to simplicity and accessibility, is often praised for its transparency. Indeed, one can easily understand how results are obtained and the linear structure enables decomposition, enhancing the overall interpretability [4, 49]. The analysis is based on its coefficients: these can be interpreted as the effect on the response of one unit change of the corresponding feature [49]. Regression weights are in that sense global measures of sensitivity. However, it is crucial to understand that the coefficients have dimensions, and therefore an order of magnitude. This can be solved by standardization, which eliminates the scaling issues [7]. Secondly, tree-based machine learning algorithms are among the popular alternatives, mainly because of their impressive performance [23]. Notwithstanding, this comes at the expense of interpretability. They are usually so complex that users can no longer understand their reasoning. Friedman and Popescu [20] therefore suggested RuleFit to analyze decision rules. In essence, rules are extracted from the ensemble and then a regularized linear regression model is fitted with both the decision rules and direct feature effects [49]. The parameter value of a decision rule equals one if the condition is met, and zero otherwise. The further interpretation is similar to ordinary linear models, namely regression coefficients are analyzed, revealing the impact and direction of the corresponding influences on a response [49].

Finally, both the agnostic and specific methods must again be verified. Since one typically uses existing packages that have already been tested, the same applies as for the training and optimization step in section 3.3. There is thus no need to retest them, although the integration itself still needs to be verified [24]. As soon as the external interfaces are properly incorporated, validation can be performed by triangulating the results of different interpretation methods. We are especially interested in consistencies and inconsistencies.

3.7 Triangulating Interpretation Methods

The second stage of the methodology concludes with validation, which aims to enhance the credibility of the overall model interpretation. Its main question is whether the results are in line with expectations and if not, how anomalies can be explained. However, unlike the surrogate model validation in section 3.4, quantitative performance indicators are not suitable for such an assessment. Numerous insights from distinct methods are involved, so triangulation would be a better approach. Scholars describe it as a means of validation where the combination of different data sources, investigators, theories, etc. leads to research findings that are more reliable and therefore more credible [52, 67]. It is based on the presumption that conclusions become stronger when a similar outcome is obtained from different perspectives [52].

In our case, one should qualitatively evaluate consistencies and inconsistencies between the results of considered meta-model interpretation techniques. Some specific examples could be: 1) comparing global sensitivity indices, feature importances, and regression weights, 2) assessing whether local methods such as LIME or a one-at-a-time SA match global patterns in dependency plots, 3) examining the extent to which decision rules are visible in SHAP, and so on. Especially SHAP is very helpful in our opinion, as it can provide one comprehensive overview of exactly how a response is impacted. This enables a rather straightforward juxtaposition with the other approaches. Note, however, that aforementioned examples are not carved in stone. It depends on the selected interpretation methods, along with the ultimate intentions of a research. We only emphasize the importance of carrying out some form of triangulation in order to validate the results. After the validation of stage II, one step of the methodology remains, namely the concluding synthesis.

3.8 Synthesizing Results to Understand Emergent Properties

The final step is to reveal and understand emergent properties of the focal agent-based model by synthesizing insights from stage II. In practice, this step is often combined with the previous one from section 3.7, although there is a crucial difference. That is, the purpose of triangulation is to assess consistencies and inconsistencies between the interpretation results, while a synthesis is to explain emergence in the system under investigation. Consequently, the latter yields a concise summary of perspectives from relevant model-agnostic and model-specific methods.

Nonetheless, different perspectives are usually necessary to detect such phenomena. This is one of the reasons why the AbACaD methodology proposed by Janssen et al. [34] was rather successful; the authors combined causal graphs with regression weights and sensitivity indices, as discussed in section 2.4. Such an approach enables one to read between the lines and use synergies to deeply understand what is actually going on in the agent-based model. A specific example with our selected interpretation methods could be as follows. First, several points in the feature space are examined with a local approach, such as LIME or a one-at-a-time SA. This gives an initial impression of how a response is affected and indicates whether there are mutual differences within the domain. Global key drivers are evaluated next, like feature importances or sensitivity indices, to observe general trends. Both the local and global results can then be juxtaposed with patterns in one and two-dimensional dependency plots, showing the effects of changing input parameters and how they are influenced by interdependencies. Once again, the above is just an example of one way to synthesize. In practice, it depends on the selected interpretation methods and the phenomena under investigation.

If done properly, it should elucidate the emergent properties of the focal system, thereby achieving the objective of the research. Two-stages were indeed necessary to: 1) create faithful surrogate models, and 2) use them for the employment of interpretation methods to understand and explain the dynamics of underlying system. With that, we continue with a description of AATOM in the next section; the agent-based model we use as a case study to prove that the methodology is efficacious.

4 Description of the Agent-based Model

To demonstrate the applicability of our two-stage methodology, we aim to detect and explain emergent phenomena in a complex sociotechnical system. It follows from the theoretical background in section 2 that a passenger terminal is the epitome of such a system: cognitive, social, technical, and organizational factors play a major role. With this in mind, Janssen et al. [33] recently developed an agent-based airport terminal operations model (AATOM) — existing alternatives were not accurate enough, too generic, too difficult to use, or the source code was not openly available. AATOM is designed with an object-oriented philosophy, allowing users to easily model the associated passenger flows. This makes it a very versatile tool, as it can be completely adapted to any set of requirements. One of its main features is that it contains prebuilt components, such as check-in desks or a security checkpoint [33]. Consequently, the layout of an entire terminal can be built with just a few lines of code. Since its introduction, the model has shown its capabilities in various studies. Some recent examples are Janssen et al. [32] on the relationship between checkpoint security and efficiency, Janssen et al. [31] on the management of airport security risks, and Mekic et al. [48] with an analysis on non-aeronautical activities and their impact on terminal operations. We focus in particular on the latter, as this is currently the most advanced version to simulate the operations of an entire terminal.

An agent-based model is known to consist of an environment, agents, and interactions between them [74]. These three components of AATOM are now briefly discussed in respective order. First of all, the environment contains all elements of an airport terminal. That includes various functional areas with physical objects, but also more abstract items such as flights [30]. The former is depicted in Figure 2, which resembles the terminal layout of Rotterdam The Hague Airport (RTHA). We specifically opted for RTHA due to the availability of data and associated insights from a previous study (see [35]). The layout was also available in the latest version by Mekic et al. [48], although some changes have been implemented as addressed in Appendix A of the Supporting Work. Regarding the flights, we consider a typical morning rush hour at RTHA. The schedule is summarized by Table E.1 in Appendix E. Secondly, cognitive agents are the key players in an environment. Three types can be defined in AATOM: passengers, operators, and orchestrators [30]. The former are trivial, operators are generally the employees in the terminal (e.g., security officers, check-in staff, cashiers, etc.), and orchestrators help with coordination and monitoring (e.g., employees who open or close check-in counters based on an airport's strategy). Agents have certain goals on which they act and interact accordingly. Behind their reasoning is a three-layered hierarchical architecture, allowing AATOM to realistically model human behavior. The structure is visualized by Figure E.1 in Appendix E. In essence, they operate as follows: observations are perceived and interpreted, allowing agents to reason so that their activities can be set. This eventually leads to the actuation of specific actions. The final component of AATOM is that agents can interact with the environment as well as with each other. The model reflects these two types of interaction in many different ways [30]. For example, check-in employees managing flights or security officers using sensors at the checkpoint are concrete cases of

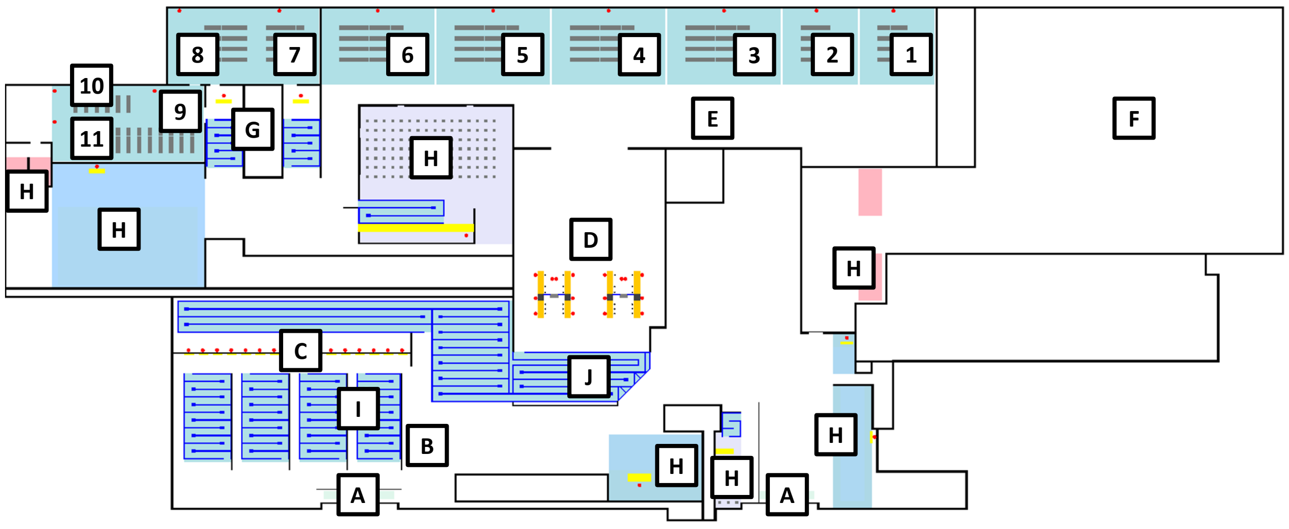


Figure 2: Terminal layout of RTHA represented in the model. Passengers arrive through entrances (A) in the public area (B). Those who have not checked-in online can do so at the counters (C) via designated queues (I). Thereafter, all passengers continue via queues (J) to the security checkpoint (D) to access the restricted area. This area is split up into a departure hall (E) and an arrival hall (F). The arrival hall is not further developed as our research focuses on solely the outbound passenger flow. The departure hall has gates 1 to 6 for flights with destinations in the Schengen area and gates 7 to 11 for flights outside the Schengen area (the numbers on the map correspond to the gate numbers). To access the latter gates, passengers should go through border control to have their passports checked (G). Along the journey, passengers are free to make use of the facility areas for non-aeronautical activities (H) [33].

interaction between agents and the environment. On the other hand, border control agents checking passports or an X-ray operator instructing another security officer to further examine some suspicious baggage are examples of agent-to-agent interaction. For more detailed information, the reader is referred to Janssen et al. [30] and Janssen et al. [33] as the key principles behind the architecture of AATOM have now been touched upon.

Finally, relevant input and output parameters are discussed. Knowing that AATOM was created as a versatile tool, we emphasize the fact that essentially everything can be customized and adapted to the requirements of a user. Nevertheless, several calibrated presets are available to restrain complexity. Mekic et al. [48, pp. 20–21] made a comprehensive overview, though two examples are the distribution of the time required for checking-in at airport counters and the distribution of passengers arriving at the terminal. We use the defaults for most of these settings. The remaining features that are considered variables for our case study are listed in Table 4. The call-to-gate strategy is bounded between 15 and 60 minutes before departure, while more information about flights, check-in strategies and security checkpoint strategies is provided by Tables E.1–E.3 in Appendix E, respectively. Furthermore, similar to the input, the output parameters are presented in Table 5. AATOM allows a user to define and extract any indicator, so again a selection has to be made. We believe that the listed

Table 4: Considered input parameters of AATOM. A remark regarding the number of passengers is that Pax_t is defined for every available time slot t. In other words, if RTHA has 7 available time slots during the simulated time frame, the model requires 10 input parameters (i.e., 7 parameters to define the number of passengers and 3 strategy parameters).

Input parameter	**Unit**	**Explanation**
Pax_t	[#]	An integer indicating how many passengers are traveling on the flight on time slot t. It is strictly positive, bounded by the maximum capacity of an aircraft. If the occupancy rate is below 50%, it becomes 0 because the flight is canceled.
$\text{CTG}_{\text{strategy}}$	[s]	A positive real number that determines the time when passengers are called to their gate prior to the departure time. It represents the airport's call-to-gate (CTG) strategy [48].
$\text{CI}_{\text{strategy}}$	[-]	An integer that determines the number of open check-in counters over time. It represents the airport's check-in (CI) strategy. An orchestrator agent couples the number with a predefined strategy [48].
$\text{SC}_{\text{strategy}}$	[-]	An integer that determines the number of open lanes at the security checkpoint over time. It represents the airport's security check (SC) strategy. An orchestrator agent couples the number with a predefined strategy [48].

metrics yield a solid indication of the airport's passenger handling performance, hence no other indicators are defined and these will form the basis of the analysis.

Now that all important aspects of AATOM have been described, the next section continues with applying our proposed methodology on the focal model. It includes the outcome of both stage I and stage II, with the ultimate purpose of explaining interesting dynamics and emergent properties of terminal activities related to the entire departure flow in RTHA.

Table 5: Relevant key performance indicators of AATOM.

Output parameter	**Unit**	**Explanation**
AvgQueueTime$_{CI}$	[s]	Indicates the average time that passengers wait in a queue until they can be served at an available check-in (CI) counter.
AvgQueueTime$_{SC}$	[s]	Indicates the average time that passengers wait in a queue until they can be served at a security checkpoint (SC) lane.
MaxPaxInQueue$_{CI}$	[#]	Indicates the maximum queue size at check-in during the simulated time frame.
MaxPaxInQueue$_{SC}$	[#]	Indicates the maximum queue size at security during the simulated time frame.
AvgTimeToGate	[s]	Indicates the average time it takes passengers to get to their gate. It is counted from the moment they arrive at the airport.
PaxCompleted$_{CI}$	[#]	Indicates the total number of passengers that have completed the check-in (CI) activity at the airport counters (i.e., the throughput at check-in).
PaxCompleted$_{SC}$	[#]	Indicates the total number of passengers that have completed the security check (SC) activity at the checkpoint (i.e., the throughput at security).
NumMissedFlights	[#]	Indicates the total number of passengers who could not reach their gate at the time of departure.
TotalExpenditure	[€]	Indicates the amount of money that all passengers together have spent during their non-aeronautical activities [48].

5 Results

This section showcases how the methodology can be deployed in practice. We apply it to the AATOM implementation of Mekic et al. [48] and start with evaluating the surrogates' performance; a critical step to ensure they generalize sufficiently well before being used for further analysis. Subsequently, the departure flow in RTHA is examined on the basis of two specific case studies. First, the total expenditure of passengers on discretionary activities in the terminal is analyzed, and then the saturation of throughput at security. Both cases are largely determined by emergence in airport terminals, so studying them will prove that our methodology is an effective way to explain such phenomena in complex sociotechnical systems — the objective of this research.

5.1 Surrogate Model Performance

Before assessing fidelity, one must first collect data and tune the surrogate model architectures. The former is elaborated upon in Appendix C of the Supporting Work. We visualize the distribution of selected data points, analyze summary statistics of the responses, and show how the stopping criterion of the active learning scheme is reached. It turns out that the training sample is sufficiently informative with 300 data points in total. This automatically leads to validation and test sets with both 100 additional observations, so that the proportion of each equals 20% of the entire sample. Secondly, hyperparameter tuning is discussed in Appendix D. There is an overview of considered model parameters and their search space, along with the results of the optimization. Convergence plots show that the algorithms can be deemed optimal after 50 initial trials and 200 subsequent Bayesian iterations. With that, the next step is to evaluate the surrogates' out-of-sample performance.

We perform validation in Table 6. Per output parameter of AATOM, the metrics are calculated for Gaussian process regression (GP), polynomial regression (LR), random forests (RF), and gradient boosting regression (GB) — the four selected meta-model architectures. The R^2 and MAPE are dimensionless, but the RMSE and MAE are expressed in the same unit as the corresponding response, given in Table 5. Finally, the surrogate model that yields the best performance for each response is indicated by an asterisk.

The first thing that immediately stands out is the disappointing generalization power of random forests. Their performance is clearly inferior to the other architectures, often with quite a large discrepancy. The initial hypothesis was that overfitting posed the issue, although their accuracy no longer improves near the stopping

Table 6: Validation of the surrogate model performance.

	PaxCompleted$_{SC}$				**AvgTimeToGate**				**PaxCompleted**$_{CI}$			
Metric	GP	LR	RF	GB*	GP	LR*	RF	GB	GP	LR	RF	GB*
R^2	0.90	0.90	0.79	0.93	0.91	0.92	0.55	0.89	0.93	0.94	0.86	0.98
RMSE	17.52	17.62	25.75	14.94	64.32	61.34	143.80	72.14	7.55	7.40	11.17	4.14
MAE	12.59	12.86	20.29	11.51	48.11	46.19	113.45	58.27	5.02	5.66	8.11	3.30
MAPE	0.02	0.02	0.03	0.02	0.04	0.04	0.09	0.05	0.01	0.02	0.02	0.01
	AvgQueueTime$_{SC}$				**NumMissedFlights**				**TotalExpenditure**			
	GP	LR*	RF	GB	GP	LR*	RF	GB	GP	LR*	RF	GB
R^2	0.90	0.92	0.57	0.86	0.70	0.80	0.53	0.43	0.97	0.98	0.94	0.97
RMSE	68.63	63.46	142.90	80.69	9.28	7.51	11.58	12.85	52.05	42.44	69.23	52.25
MAE	52.50	49.54	118.74	64.33	7.09	4.42	6.94	6.60	40.34	33.76	55.61	42.00
MAPE	0.08	0.08	0.18	0.10	N/A†	N/A†	N/A†	N/A†	0.03	0.03	0.04	0.03
	AvgQueueTime$_{CI}$				**MaxPaxInQueue**$_{SC}$				**MaxPaxInQueue**$_{CI}$			
	GP	LR*	RF	GB	GP	LR*	RF	GB	GP	LR*	RF	GB
R^2	0.91	0.95	0.87	0.95	0.91	0.92	0.65	0.91	0.90	0.95	0.78	0.92
RMSE	19.46	14.13	23.73	15.13	9.15	8.69	17.99	9.06	0.58	0.43	0.86	0.51
MAE	13.79	9.78	16.16	10.25	6.60	6.69	14.59	7.24	0.43	0.29	0.66	0.39
MAPE	0.05	0.04	0.06	0.04	0.07	0.06	0.14	0.07	0.04	0.02	0.05	0.03

*Best performing surrogate model architecture for the associated response
†Mathematically undefined because of division by zero

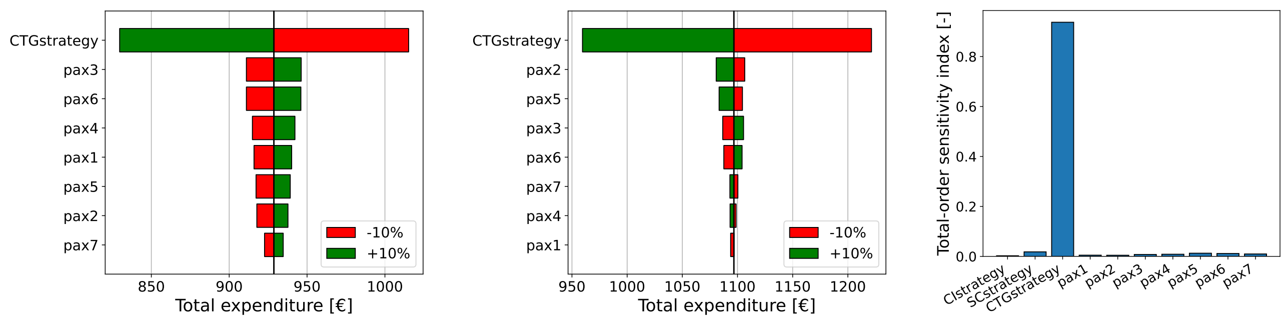
criterion of the active learning algorithm, nor did regularization help (see Appendix C and D, respectively). This is in stark contrast to LR, GP and GB, whose performance is actually rather impressive. While the validation metrics of these three architectures are generally comparable, regularized polynomials seem to mimic AATOM most accurately. Namely, they have been selected 7 out of 9 times, with only the throughput at check-in and security being better estimated by gradient boosting. This may be somewhat surprising, but a plausible explanation could be that the associated responses behave similarly according to the format of higher-order polynomials. Their combination then naturally produces superior results. For instance, the average queuing time at security is resembled with an expected error of about 1 minute and the total expenditure with an error of about 40 euros. Only the number of missed flights appears to be more challenging: the coefficient of determination decreases to 0.80. Yet, even that is still acceptable, because it is the response most influenced by higher-order knock-on effects and less directly by the features themselves. Consequently, it becomes inherently more difficult to predict (see also the conclusions of De Leeuw et al. [11], which are consistent with our results). Furthermore, note the relative difference between the RMSE and MAE — the number of missed flights has the highest of all, indicating the presence of outliers. This is confirmed by plotting the predicted against observed values, as Figure F.1 clearly shows in Appendix F. In spite of that, LR convincingly remains the best performing architecture for the response, while the tree-based ensembles are inadequate. Altogether, LR, GP and GB seem to mimic the output parameters of AATOM rather well, despite the fact that RTHA is a complex sociotechnical system. However, there is evidently "no free lunch", as multiple architectures must be considered and carefully optimized per individual parameter to achieve a high accuracy [76].

As there is now evidence that each response can be closely resembled by at least one surrogate, we continue with analyzing the total expenditure of passengers on discretionary activities. A precursory remark is that both case studies solely deploy a response's best performing meta-model for agnostic analysis.

5.2 Analysis of the Total Expenditure on Discretionary Activities

The first case study examines the spending behavior of passengers on non-aeronautical activities. In fact, this was the main topic of the analysis by Mekic et al. [48], though we go more in-depth to demonstrate the strengths of synthesizing interpretation techniques applied to surrogate models. The total expenditure represents the amount of money all passengers together spent on activities such as shopping, dining, etc., during the simulated time frame. These events are of course not mandatory to catch a flight and hence not a priority, so passengers will only consider them if they have enough time. Readily, it shows that the expenditure is an ideal starting point to analyze emergence; the indicator is affected by various interdependent phenomena in the airport terminal.

We commence the analysis in Figure 3 by exploring the sensitivity of features. First, two one-at-a-time assessments are performed in Figure 3a and 3b, which depict tornado diagrams of local sensitivities. An

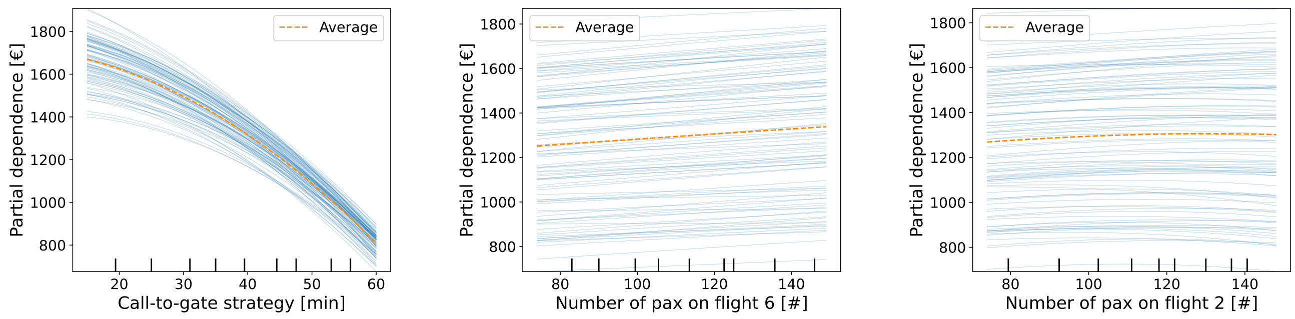


(a) Local sensitivity uncrowded terminal. (b) Local sensitivity crowded terminal. (c) Total-order global sensitivity.

Figure 3: Sensitivity analysis of the total expenditure.

uncrowded scenario is compared against a crowded one, both assuming poor airport staffing strategies (check-in and security check strategy 1) and an early call-to-gate (48 minutes before departure). The crowd is controlled by adopting a load factor of about 65% and 85% on all flights, respectively. Poor staffing strategies in combination with an early call-to-gate does not provide the ideal condition for discretionary activities — passengers have less spare time in the terminal. Nevertheless, the baseline values of the tornado diagrams suggest that busier scenarios lead to more spending. This makes sense, as larger crowds are naturally expected to have a higher expenditure. Both diagrams associate the greatest sensitivity to the call-to-gate strategy, though note that it has a negative direction. In other words, the sooner passengers are called to the gate, the less they spend along their journey and vice versa. While this is not surprising, a more striking difference is the influence of certain flights' load factor. They are all harmonious for the uncrowded terminal, but not when it gets busier. For flights 2 and 5 in particular, the total expenditure decreases as more passengers travel on those flights. This is rather counter-intuitive, so we resort to other methods to explain the negative effect and why it depends on the scenario. We conclude the sensitivity analysis by plotting total-order Sobol indices in Figure 3c. They attribute the variance of a response to the features in proportion to their contribution, so total expenditure appears to be most sensitive to the call-to-gate strategy. The global impression thus corresponds to the local impressions, although it is more pronounced.

In Figure 4, the analysis continues with partial dependence plots, which visualize the marginal effect of a feature on the total expenditure. On the left, Figure 4a shows the effect of the call-to-gate strategy. As expected, the curve has a downward slope, indicating a lower spending for higher strategies. More interestingly, however, is that the gradient increases the sooner passengers are called to their gate. This suggests that the total expenditure is less sensitive to the feature in the lower part of its domain. To illustrate with a specific example, changing the call-to-gate strategy from 20 to 30 minutes will reduce the spending less than changing it from 50 to 60 minutes. Secondly, Figure 4b and 4c depict the marginal effects of the number of passengers on flight 6 and 2, respectively. Both plots are consistent with the previous one-at-a-time sensitivity analysis. For flight 6, it holds that larger occupancy rates yield more expenditure in the terminal; the partial dependence rises almost linearly and there is a solid homogeneity among the individual conditional expectation curves. The same cannot be said of flight 2, where spending first rises, then levels off, and eventually falls back slightly. This indeed indicates that at some point, adding passengers may have a negative effect on the ability to engage in non-aeronautical activities. Note, however, the increased heterogeneity near the upper bound of the feature space. It does not seem to be case for all scenarios in the terminal. For example, if all the other flights have very few passengers, then it is probably the other way around. Nonetheless, some flights clearly hinder a number of passengers from enjoying leisure activities during their journey.



(a) Marginal effect of the call-to-gate. (b) Marginal effect of flight 6. (c) Marginal effect of flight 2.

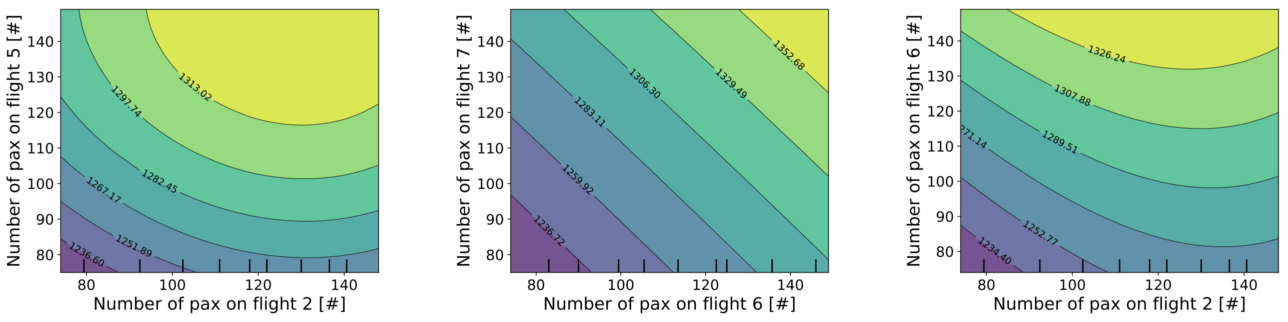
Figure 4: One-dimensional partial dependence and individual conditional expectation plots of the total expenditure.

To understand exactly why this happens to some and not all of them, we revisit the simulated schedule in Table E.1 in Appendix E. Certain flights, such as the second and sixth, are assigned to the same check-in counters, although that in itself is nothing unusual. Yet, it is remarkable that flight 2 and 5, which mostly affect expenditure in the negative direction according to Figure 3b, are both first in line. This could indicate the presence of emergent knock-on phenomena where passengers of one flight hinder those of another, but only if the terminal is sufficiently crowded. In turn, waiting times at check-in and security increase, ultimately leading to fewer opportunities for discretionary activities. That would be a logical explanation for the observed pattern in the partial dependence plot of flight 2, and also why it is not the case for others like flight 6. The third flight goes against this logic, although note the lower capacity on its successor — the knock-on effect is therefore presumably less powerful. To prove whether our reasoning is actually true, we further examine the partial dependence in Figure 5, but now from a two-dimensional perspective.

The marginal effect of the two flights that introduce the highest knock-on phenomena is depicted in Figure 5a. One can see that the total expenditure indeed levels off and may even decrease the more passengers they accommodate. The curvature in the contour plot increases near the top right corner, so the average effect becomes stronger at higher load factors on both flights; they seem to amplify one another. Also, the impact of the second flight is larger than the fifth, but that makes sense since it is earlier in the schedule. It has thus more passengers behind to impede at check-in and security. As a matter of fact, Figure 5b confirms that the phenomenon is caused by a sequence of consecutive flights. The graph shows the marginal effect of the latter two in the schedule, and clearly there are almost no signs left of a deteriorating expenditure. For both, higher load factors result in more participation in non-aeronautical activities, which was expected from Figure 4b. Ergo, flight 6 and 7 do not hinder other passengers, but are only hindered themselves by their predecessors. The interdependence between two flights that check-in at the same counters is visualized by Figure 5c. Despite the pattern looks entirely different, it is essentially a combination of the one-dimensional partial dependencies in Figure 4b and 4c. The plot reveals two interesting phenomena: 1) the expenditure increases with more passengers on flight 6, albeit at a slower rate as occupancy rises on flight 2, and 2) the point at which adding passengers on flight 2 starts to negatively affect the expenditure shifts to the right with a lower occupancy on flight 6. It demonstrates that there are indeed knock-on effects, providing explicit evidence of emergence. That is, passengers from earlier flights impede the terminal journey of those from subsequent flights, which limits their ability to engage in discretionary activities. This, in turn, lowers the total expenditure.

While we focused primarily on examples where travelers were delayed at both check-in and security, one should bear in mind that the phenomenon is not limited to solely these cases. For instance, the marginal effect of flights 1 and 4 in Figure F.2 in Appendix F still shows a reasonable degree of curvature, even though they are last to use their respective check-in counters. This means that they also affect subsequent flights, but only by increasing the waiting time at security and thus not at check-in.

The analysis is finalized by comparing the previous results with the outcome of SHAP and regression weights as a means of validation. SHAP's bee swarm summary plot is depicted by Figure F.3 in Appendix F. It does not reveal the aforementioned emergent phenomena, but the global impact and direction of features are clearly in accordance with the expectations. However, one salient detail is the influence of staffing strategies. We did not elaborately discuss them before, though the plot suggests that especially security check strategy 1 and 2 may have a negative impact on the opportunity to participate in non-aeronautical activities. This is indeed the case if one has a look at the marginal effect in Figure F.4. It will therefore come as no surprise that these two are among the worst possible alternatives, as shown by Table E.3 in Appendix E. Secondly, by analyzing weights of the regularized higher-order polynomial, we also include a model-specific method. The intercept equals 1162.4, while the call-to-gate-strategy squared has the highest absolute coefficient with a value of -506.6. Both the impact and direction are thus as expected, and it is also interesting to point out that the shape of $f(x) = 1162.4 - 506.6x^2$ heavily resembles the one in Figure 4a. As a final remark, the possibilities



(a) Marginal effect of flight 2 and 5. (b) Marginal effect of flight 6 and 7. (c) Marginal effect of flight 2 and 6.

Figure 5: Two-dimensional partial dependence plots of the total expenditure.

of RuleFit have been explored, but the results were too unstable to draw conclusions. Not only are seemingly important decision rules entirely different for successive trials, their overlap also introduces a lot of ambiguity. We therefore argue that the method is less suitable for interpretation and validation purposes.

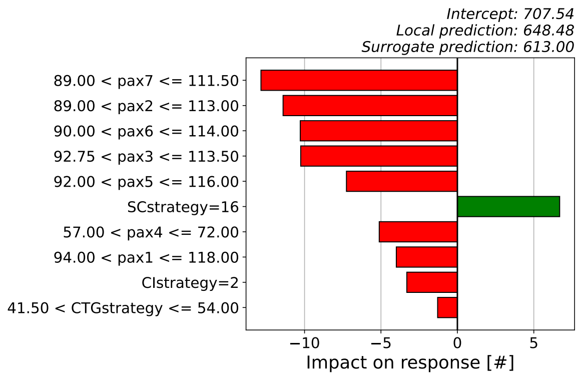
This concludes the analysis of total expenditure on discretionary activities. A combination of poor staffing strategies at security and high occupancy rates on certain flights was found to disrupt the terminal journey of passengers on subsequent flights. As a result of these knock-on phenomena, less free time is left for discretionary activities, which has a negative effect on the total expenditure. With that, the throughput at the security checkpoint is examined next, and how its saturation affects the journey of outbound passengers at RTHA.

5.3 Analysis of the Saturation at Security

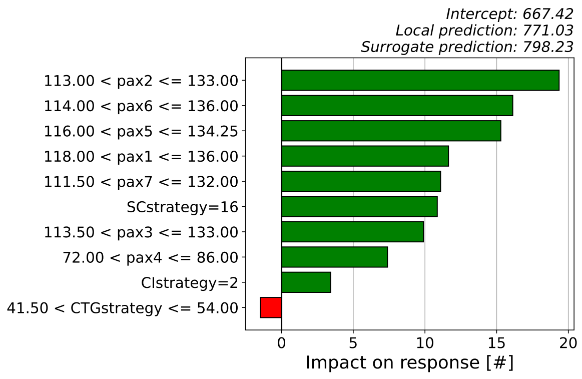
The second case study originates from the observation that throughput at security and the number of missed flights are not correlated at all, while correlation between the latter and average waiting time at security is 0.80 ($p < 0.01$). Section C.2 in Appendix C of the Supporting Work argued that these results make sense, namely the checkpoint is the system's main bottleneck. At some point, it reaches its maximum capacity as the crowd grows. Consequently, the throughput remains constant while waiting times add up quickly, causing passengers to miss their flight. Such phenomena are the result of various interdependent dynamics. The aim of current case study is therefore to provide more in-depth evidence of these emergent properties.

The analysis commences again with a local method, but this time with insights from LIME instead of a one-at-a-time sensitivity analysis. Figure 6 explains how the throughput at security is influenced. We compare an uncrowded terminal against a crowded one; both scenarios presume good staffing strategies and an early call-to-gate, the values of which are shown in the graphs. A trivial conclusion is that high load factors lead to positive contributions to the number of passengers passing through security, and vice versa. Furthermore, note that constantly operating the checkpoint at full capacity — security check strategy 16 — positively impacts the total flow. That is logical, as it delivers the best possible service, thereby maximizing throughput. Notwithstanding, one should remain vigilant about the discrepancy between predictions of LIME and the surrogate. The error is around 30 passengers for both scenarios, which is rather large compared to the size of the feature impacts. This calls for some caution with using LIME as the sole method of interpretation, even if it produces comprehensible insights. For example, if we triangulate it with the outcome of SHAP in Figure F.5 of Appendix F, there are clearly some inconsistencies. On the one hand, LIME appears to inflate the impact of security check strategy 16; it is not even visible on the bee swarm summary plot since its effect is negligible. On the other hand, SHAP never reports a negative impact for the second check-in strategy, despite it being almost negligible as well. Nevertheless, the other features are generally consistent and in line with expectations. We now focus on flights 2 and 6 for the remainder of the analysis. These two are among the most impactful, according to LIME and SHAP, and have an additional interconnection. Indeed, they are assigned to the same check-in counters, allowing us to examine whether there are again knock-on effects as in the previous case study.

Interdependent relationships are best understood by plotting the partial dependence, so Figure 7 visualizes three relevant cases. The marginal effect of flights 2 and 6 on the number of passengers who completed security is depicted in Figure 7a. First of all, note that decision boundaries are not as smooth as in previous dependence

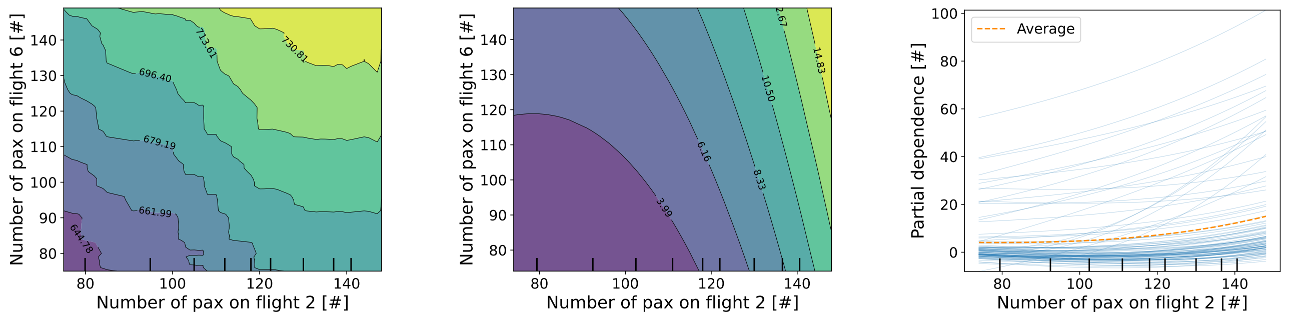


(a) Uncrowded terminal.



(b) Crowded terminal.

Figure 6: Local interpretable model-agnostic explanations of the throughput at security. The interpretation is as follows. The vertical axis shows all input parameters and their assumed values, while the corresponding contributions to the response are plotted horizontally. These contributions can be considered as the impact on a prediction relative to the intercept of LIME's approximation. Bars appear red if the effect is negative and green otherwise. To connect the dots, LIME arrives at its local prediction by adding the individual contributions of all features to the intercept, which should then be close to the actual outcome of the investigated surrogate model.



(a) Marginal effect of flight 2 and 6 on the throughput at security.

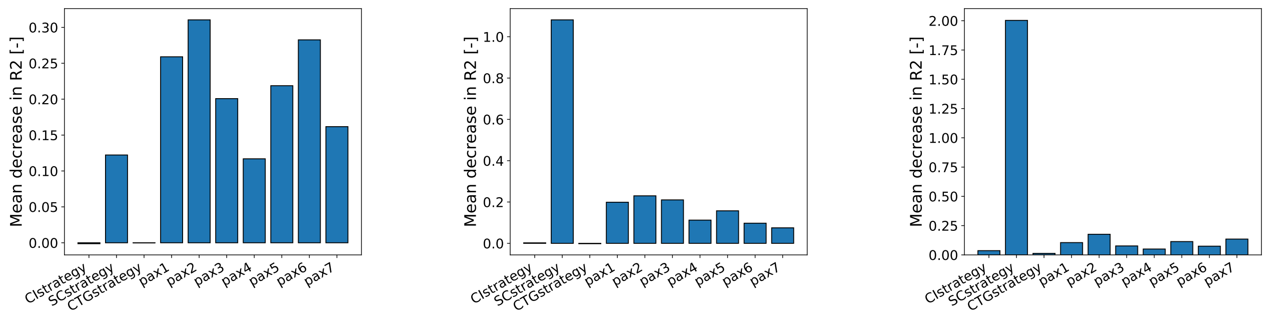
(b) Marginal effect of flight 2 and 6 on the number of missed flights.

(c) Marginal effect of flight 2 on the number of missed flights.

Figure 7: Partial dependence plots of the throughput at security and number of missed flights.

plots. This makes sense because it is one of the two responses that is mimicked by gradient boosting rather than by a polynomial; the outcome of the former is inherently more erratic. Secondly, at a given occupancy rate on flight 6, the response levels off when the number of passengers on the second flight exceeds 120–130. That indeed seems to confirm our earlier hypothesis: at some point, the security checkpoint becomes saturated, hence adding passengers no longer increases throughput. The phenomenon is not visible in flight 6, which is logical as it is much later in the schedule — obstructing the checkpoint is therefore less likely. In fact, its load factor and the throughput practically have a positive linear relationship. These findings are also apparent in the one-dimensional counterparts, depicted by Figures F.6a and F.6b in Appendix F. A natural follow-up question is then what happens to passengers who failed to complete the security check on time. Figure 7b shows the consequence by plotting the marginal effect of flights 2 and 6 on the total number of missed flights. The graph confirms our expectations, namely from about 120 passengers on flight 2, more and more people do not reach their gate on time. Flight 6 also has a slight impact, albeit rather limited because contour lines in the critical region are almost vertical. Finally, Figure 7c shows that the effect of flight 2 even accelerates as it approaches its maximum capacity, suggesting that there must be some degree of knock-on phenomena. The sixth flight does not appear to be affected, as otherwise it would have been visible in Figure 7b through more horizontal contours in the upper part. Nonetheless, flight 2 clearly has a considerable impact on whether or not the checkpoint may become obstructed. Note, however, the heterogeneity among the individual conditional expectation curves in Figure 7c. This proves that the conditions in the terminal remain important. For example, if it is not busy at all, then adding passengers to solely flight 2 will not necessarily increase the number of missed flights.

Next, we also analyze feature importances to see exactly which key drivers control the checkpoint's throughput, average waiting time, and the ensuing number of missed flights. The results are shown in Figure 8, respectively. It is immediately noticeable that the graphs of the latter two are similar; both are driven primarily by the staffing strategy at security and to a lesser extent by occupancy rates. The opposite holds for the checkpoint's throughput, although the difference is not as pronounced. One should interpret these results as follows. Under normal circumstances, more passengers lead to more passage through security, which is therefore mainly determined by the load factor on flights. However, if the airport opts for a bad strategy, waiting times may increase considerably. The extent also depends on how busy it is, but personnel strategy is more decisive. That is logical, as it directly dictates the number of lanes to be opened. Ceteris paribus, fewer lanes will always lead to longer waiting times, but not necessarily to a lower throughput. This explains the difference between Figures 8a and 8b. However, there is a risk that the waiting time, which we know is predominately driven



(a) Throughput at security.

(b) Average queuing time at security.

(c) Number of missed flights.

Figure 8: Visualization of permutation feature importances.

by strategy, will continue to rise so passengers are no longer able to reach their gate on time. At that point, the number of missed flights will increase rapidly, especially when it is busy. Queuing time and the number of missed flights thus have the same key drivers. These findings are confirmed by plotting the marginal effect of the security check strategy on the three respective responses in Figure F.7 of Appendix F. There is clearly a strong influence of certain strategies; especially those that close all but one lane after two hours appear to be really detrimental (see Table E.3 in Appendix E). While the relative effect on throughput may seem limited, note that Figure F.7a and F.7c are actually complementary. Hence, passengers who failed to pass security on time, will eventually miss their flight. Based on these insights, we conclude that there are indeed emergent phenomena which can lead to saturation of the security checkpoint. If so, waiting times may increase substantially, resulting in a high number of missed flights. This particularly happens in combination with poor staffing strategies.

Finally, the interpretations are again juxtaposed with SHAP as a means of validation. The bee swarm summary plot of the throughput was already shown in Figure F.5, but we also added those of the average queuing time and number of missed flights in Figures F.8 and F.9 of Appendix F, respectively. Aside from the previously discussed inconsistencies between LIME and SHAP, the results are actually rather consistent and in line with expectations. However, there is one interesting finding to point out. Security check strategy 4 makes the checkpoint operate at full capacity from an hour onward, so it is presumed to be a solid approach. According to SHAP, throughput is indeed higher and fewer passengers end up missing their flight. Yet, it appears that the strategy also prolongs the expected waiting time at security. This may conflict with what one would initially believe, although in fact it makes sense. By operating with too small a capacity, more passengers arrive than can be handled, causing the queue to grow. If suddenly all lanes are opened, then there is already a considerable queue while passengers are still arriving. Eventually, the queue is eliminated and a smooth passage is possible, although it took some time and increased the average wait. This again confirms that longer waiting times can, but not always, lead to a higher number of missed flights. Ergo, one ought to be careful about implying such causalities. Nonetheless, our previous arguments are in accordance with SHAP, which concludes the second case study. In the next section, we continue with the discussion where further implications are derived.

6 Discussion

From the results, it is evident that the proposed methodology is rather effective in explaining the dynamics of complex sociotechnical systems, based on an existing validated agent-based simulation model. The main presumption was that one already has access to such a faithful model, though we remind the reader not to take this for granted. In practice, it will not always be the case and alternatives such as obtaining real-word data are often infeasible. Moreover, one must also be aware of imperfections in the model. AATOM is known to be meticulously close to reality, but there are always certain assumptions. For example, Mekic et al. [48] mention passengers' perfect sense of time, the neglect of boarding delays, and so on. Surrogate models are directly subject to these limitations.

The surrogates were created in stage I of the methodology. While the resulting out-of-sample performance is rather impressive, it is in line with expectations. Namely, our validation metrics are comparable to those of De Leeuw et al. [11], who meta-modeled similar responses of AATOM. An important note, however, is that we investigated the departure flow in the terminal of RTHA; a much more complex system than theirs. Consequently, more sophistication does not automatically lead to a deterioration in the performance of surrogate models, though one should also not forget our advanced sampling approach — active learning is deemed rather powerful [42, 21]. An interesting discovery is that the above authors achieved high accuracy with random forests, thereby contradicting our findings. Indeed, the meta-modeling performance of random forests was found to be notably weak. Two probable causes are the increased system complexity or a different sampling strategy, as the other methodological steps were largely similar. Furthermore, the predictive strength of regularized higher-order polynomials should definitely not be underestimated. They turned out to be superior in 7 out of 9 cases. This may be somewhat surprising at first, but we emphasize that they are known to perform particularly well when responses behave like polynomials [58]. In addition, they provide analytical expressions of these responses as a function of the input parameters. Such transparency is considered a major strength, especially if the ultimate purpose is to gain insight into underlying relationships.

Yet, one cannot always rely on transparency to explain the rationale behind surrogates. Stage II of the methodology therefore introduced a more generic approach, combining both model-agnostic and model-specific interpretation methods. Emergent phenomena were clearly observed and the first case study is accordant with the conclusions of Mekic et al. [48]. For example, delaying the call-to-gate does indeed increase total expenditure, as does shortening waiting times at security. However, our methodology allowed us to go further in depth. We could even attribute certain phenomena down to load factors on specific flights. Stage II thus provides a unique insight into actual root causes, which is a considerable strength. For the synthesis, mainly model-agnostic methods were used. They are more accessible and easier to apply, while the model-specific methods were prone to the following weaknesses: 1) RuleFit proved to be very unstable, and 2) it was rather difficult to materialize polynomial regression weights into concrete implications. Regarding the latter, regularization may have drastically reduced the number of non-zero coefficients, the resulting expressions still consist of more than 100 terms each. This remains challenging to interpret, so transparency does not always promote explainability. The downside thereof is that we have not explicitly revealed how exactly a surrogate arrives at its outcome. Agnostic methods do clarify the behavior of responses, but not the internal reasoning of a model. Next, a weakness of stage II is finding the right illustrations to include in the analysis. By considering several approaches, numerous results are obtained, from which a careful selection must be made. We found a one-at-a-time sensitivity analysis, LIME, and correlation coefficients particularly useful to pinpoint interesting directions. From there, partial dependence plots turned out to be the preferred alternative when visualizing relevant dynamics, along with support from Sobol indices and feature importances to identify key drivers. Two final remarks are: 1) LIME raises questions about its reliability, despite providing comprehensible insights, and 2) SHAP's bee swarm summary plots are especially strong for validation purposes because of their holistic view. Altogether, we conclude that the methodology's second stage excels at explaining the dynamics of complex systems, even if it is sometimes challenging to find the right illustrations for the interpretation. Surrogate abstractions do preserve emergent properties, and hence they can be employed to enhance the understanding of a system that is otherwise difficult to analyze.

To generalize above findings, the scope of the two-stage methodology is not limited to solely agent-based models, nor to airport terminal operations. Firstly, it can be applied to essentially any type of model, although preferably one that entails a heavy computational burden. Otherwise, there are probably more efficient approaches, without the need to create surrogates as an intermediate step. That being said, it is not obligatory to use the methodology in its entirety. The two stages are in fact modular and may be deployed separately. For example, if accurate machine learning models are already available, one can go directly to the second stage. Conversely, if there is a mere desire to speed up the calculation process, then the first stage will suffice. The combination of both stages is thus only recommended if the ultimate purpose is to enhance the understanding of a particular model, for which conventional approaches fall short due to computational limitations. Secondly, we have applied the methodology to airport terminal operations, but its applicability certainly goes further. The only requirements are that responses should be numerical in nature and there must be a meaningful relationship between input and output parameters, direct or indirect. The former because promising interpretation methods for classification were not incorporated, such as in the case study of Belle and Papantonis [5]. The latter because otherwise it is rather difficult to visualize interdependencies and explain the overall dynamics of a system. A final consideration is whether it is desirable to have solely one surrogate model for a response across the entire feature space. For example, we found the number of missed flights more challenging to mimic. The predominant influence of higher-order interaction effects definitely plays a role, but the response is also zero for the vast majority of scenarios in the terminal. This makes sense, because missing a flight due to overcrowding is actually quite rare. It seems that surrogates cannot handle such patterns as well as agent-based models, so a better alternative may be to first distinguish between nominal and non-nominal scenarios, and then consider the respective regions of the feature space individually. A proof of concept is presented by ten Broeke et al. [66], where classification was applied first, followed by regression. To conclude, our proposed methodology is fairly generic, although there are boundaries to its applicability within which the best results are achieved.

Finally, we address two important limitations. On the one hand, a critical assumption is that responses are deemed deterministic — their stochasticity was averaged out by the law of large numbers. This is of course not true in reality and can even affect the emergence and knock-on effects in the system. On the other hand, the Hammersley sequence was selected as the initial sampling method. Based on our experience, we do not recommend this for future work. The reason is that in some exceptional cases, its beneficial properties only appear when the sample is large enough with respect to the number of features (see Appendix C of the Supporting Work). While related issues were largely resolved by the subsequent active learning algorithm, another method, such as Latin hypercube sampling, could have prevented them in the first place [71, 38]. Having discussed the implications of our methodology, the research is finalized with a conclusion and further recommendations.

7 Conclusions and Future Work

The motivation for our research originates from the observation that existing airport terminal operations models: 1) suffer from heavy computational requirements, and 2) reveal their emergent properties only a posteriori. These are typical challenges of agent-based modeling, the principle according to which they are usually built. Therefore, we introduced a two-stage methodology to analyze such systems in a more efficient way. The first stage involves the development of faithful surrogate models, whereafter the second stage applies techniques from the emerging field of explainable artificial intelligence to these abstractions. The novelty of our methodology lies thus in the amalgamation, rather than in the respective research fields themselves. Indeed, we have explored their common ground to take advantage of synergies. A successful application reveals the properties of the focal system, which in the case of a sociotechnical system mainly concerns emergent phenomena.

Proof of the methodology's efficacy is provided by conducting two case studies on AATOM; a validated agent-based airport terminal operations model. On the one hand, we looked at the total expenditure on non-compulsory activities, like shopping and dining. It was found that the journey of some passengers may be disturbed in such a way there is an effect on their spending behavior. Knock-on phenomena were observed, with travelers from earlier flights holding up those from later flights at check-in and security. Consequently, less free time is left to engage in discretionary activities. It happens especially when the terminal is busy in combination with poor airport staffing strategies. This is a clear example of emergence, the root causes of which could even be associated to specific strategies and the occupancy on certain flights. On the other hand, we also examined the throughput at security. More passengers means more passage, but there is an evident point where the checkpoint reaches its maximum capacity. As a result, throughput remains constant, while the queue and therefore the waiting time quickly increase. This even goes so far as to put passengers at risk of missing their flight. Again, unequivocal evidence of emergent properties, which are thus clearly preserved in surrogates. The key drivers of the phenomenon could also be traced back, along with the critical settings; it only occurs under certain conditions. Altogether, the case studies demonstrated that the proposed methodology is indeed able to accurately abstract and explain the dynamics of airport terminal operations through surrogate modeling an existing simulation model. This confirms the research objective and emphasizes the strengths of combining meta-modeling with interpretable machine learning. Unique and detailed insights were attained into properties and relationships that would otherwise have been very difficult to reveal due to computational limitations.

We conclude the research by recommending three promising directions for future work, based on our experience. First of all, it is believed that responses can be mimicked more accurately if an advance distinction is made between certain scenarios in the airport terminal. For example, one could distinguish between nominal and non-nominal situations, as mentioned in the discussion. Their associated dynamics are rather diverse, which raises the question of the best approach to meta-model disparate and rare events — a particularly relevant question for scholars interested in safety and security related issues. Secondly, we solely looked at averages or total values of responses, but it is evident that their behavior may change over the simulated time frame. Such changes can be quite considerable; think of the waiting time at security, which is strongly influenced by the flight schedule. Hence, including time as an extra dimension could reveal properties that were previously invisible. Doing so will certainly add complexity, yet it is crucial to enhance the understanding of a sociotechnical system even further. Our final recommendation relates to the interpretation of machine learning models. While the discipline is still emerging, most effort is clearly being put into model-agnostic methods. That makes sense because they have a broader applicability, though a model's explicit internal reasoning can only be elucidated through model-specific approaches. We therefore advocate further development of the latter.

References

[1] R. Alizadeh, J. K. Allen, and F. Mistree. Managing computational complexity using surrogate models: a critical review. *Research in Engineering Design*, 31(3):275–298, July 2020. ISSN 0934-9839. doi: 10.1007/s00163-020-00336-7. URL `http://link.springer.com/10.1007/s00163-020-00336-7`.

[2] R. Andonie. Hyperparameter optimization in learning systems. *Journal of Membrane Computing*, 1 (4):279–291, Dec. 2019. doi: 10.1007/s41965-019-00023-0. URL `http://link.springer.com/10.1007/s41965-019-00023-0`.

[3] F. Archetti and A. Candelieri. *Bayesian Optimization and Data Science*. SpringerBriefs in Optimization. Springer International Publishing, Cham, 2019. ISBN 978-3-030-24494-1. doi: 10.1007/978-3-030-24494-1. URL `http://link.springer.com/10.1007/978-3-030-24494-1`.

[4] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, June 2020. ISSN 1566-2535. doi: 10.1016/j.inffus.2019.12.012. URL `https://www.sciencedirect.com/science/article/pii/S1566253519308103`.

[5] V. Belle and I. Papantonis. Principles and Practice of Explainable Machine Learning. *Frontiers in Big Data*, 4:39, 2021. ISSN 2624-909X. doi: 10.3389/fdata.2021.688969. URL `https://www.frontiersin.org/article/10.3389/fdata.2021.688969`.

[6] A. Bhosekar and M. Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering*, 108:250–267, Sept. 2017. ISSN 0098-1354. doi: 10.1016/j.compchemeng.2017.09.017. URL `https://linkinghub.elsevier.com/retrieve/pii/S0098135417303228`.

[7] E. Borgonovo and E. Plischke. Sensitivity analysis: A review of recent advances. *European Journal of Operational Research*, 248(3):869–887, Feb. 2016. ISSN 0377-2217. doi: 10.1016/j.ejor.2015.06.032. URL `https://linkinghub.elsevier.com/retrieve/pii/S0377221715005469`.

[8] K. Cheng, Z. Lu, C. Ling, and S. Zhou. Surrogate-assisted global sensitivity analysis: an overview. *Structural and Multidisciplinary Optimization*, 61(3):1187–1213, Mar. 2020. ISSN 1615-147X. doi: 10.1007/s00158-019-02413-5. URL `http://link.springer.com/10.1007/s00158-019-02413-5`.

[9] D. Chicco, M. J. Warrens, and G. Jurman. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ Computer Science*, 7, July 2021. ISSN 2376-5992. doi: 10.7717/peerj-cs.623. URL `https://peerj.com/articles/cs-623`.

[10] D. Curcio, F. Longo, G. Mirabelli, and E. Pappoff. Passengers Flow Analysis And Security Issues In Airport Terminals Using Modeling & Simulation. In *ECMS 2007*, pages 374–379. ECMS, June 2007. ISBN 978-0-9553018-2-7. doi: 10.7148/2007-0374. URL `http://www.scs-europe.net/dlib/2007/2007-0374.htm`.

[11] B. De Leeuw, S. S. Mohammadi Ziabari, and A. Sharpanskykh. Surrogate Modeling of Agent-based Airport Terminal Operations. In *Multi-Agent-Based Simulation XXIII*, Auckland, New Zealand, Mar. 2022. URL `https://mabsworkshop.github.io/articles/MABS_2022_paper_9.pdf`.

[12] R. De Neufville, A. Odoni, P. Belobaba, and T. Reynolds. *Airport Systems: Planning, Design, and Management.* McGraw-Hill, New York, 2nd edition, 2013. ISBN 978-0-07-177058-3.

[13] F. Dekking, C. Kraaikamp, H. Lopuhaä, and L. Meester. *A modern introduction to probability and statistics: understanding why and how.* Springer texts in statistics. Springer, London, 2005. ISBN 978-1-85233-896-1.

[14] R. Elshawi, M. H. Al-Mallah, and S. Sakr. On the interpretability of machine learning-based model for predicting hypertension. *BMC Medical Informatics and Decision Making*, 19(1):146, Dec. 2019. ISSN 1472-6947. doi: 10.1186/s12911-019-0874-0. URL `https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-019-0874-0`.

[15] D. S. Fay. A biologist's guide to statistical thinking and analysis. *WormBook*, pages 1–54, July 2013. ISSN 15518507. doi: 10.1895/wormbook.1.159.1. URL `http://www.wormbook.org/chapters/www_statisticalanalysis/statisticalanalysis.html`.

[16] A. Fisher, C. Rudin, and F. Dominici. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019. ISSN 1533-7928. URL `http://jmlr.org/papers/v20/18-760.html`.

[17] A. I. J. Forrester, A. Sóbester, and A. J. Keane. *Engineering Design via Surrogate Modelling: A Practical Guide.* Wiley, 1 edition, July 2008. ISBN 978-0-470-06068-1. URL `https://onlinelibrary.wiley.com/doi/book/10.1002/9780470770801`.

[18] J. H. Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1–67, Mar. 1991. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1176347963. URL `http://projecteuclid.org/journals/annals-of-statistics/volume-19/issue-1/Multivariate-Adaptive-Regression-Splines/10.1214/aos/1176347963.full`.

[19] J. H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. ISSN 0090-5364. URL `http://www.jstor.org/stable/2699986`. Publisher: Institute of Mathematical Statistics.

[20] J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3), Sept. 2008. ISSN 1932-6157. doi: 10.1214/07-AOAS148. URL `https://projecteuclid.org/journals/annals-of-applied-statistics/volume-2/issue-3/Predictive-learning-via-rule-ensembles/10.1214/07-AOAS148.full`.

[21] J. N. Fuhg, A. Fau, and U. Nackenhorst. State-of-the-Art and Comparative Review of Adaptive Sampling Methods for Kriging. *Archives of Computational Methods in Engineering*, 28(4):2689–2747, June 2021. ISSN 1134-3060. doi: 10.1007/s11831-020-09474-6. URL `https://link.springer.com/10.1007/s11831-020-09474-6`.

[22] A. Graham. *Managing Airports: An International Perspective.* Routledge, New York, 4th edition, Sept. 2013. ISBN 978-0-415-52941-9.

[23] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems*, volume 2th. O'Reilly Media, Sebastopol, CA, 2019. ISBN 978-1-4920-3264-9.

[24] B. Hailpern and P. Santhanam. Software debugging, testing, and verification. *IBM Systems Journal*, 41(1): 4–12, 2002. ISSN 0018-8670. doi: 10.1147/sj.411.0004. URL `https://ieeexplore.ieee.org/document/5386906`.

[25] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer series in statistics. Springer, New York, NY, 2nd edition, 2009. ISBN 978-0-387-84857-0.

[26] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe, and I. Shcherbatyi. Scikit-Optimize: Sequential model-based optimization in Python, Oct. 2021. URL `https://zenodo.org/record/5565057`.

[27] F. Hutter, J. Lücke, and L. Schmidt-Thieme. Beyond Manual Tuning of Hyperparameters. *Künstliche Intelligenz*, 29(4):329–337, Nov. 2015. ISSN 0933-1875. doi: 10.1007/s13218-015-0381-0. URL `http://link.springer.com/10.1007/s13218-015-0381-0`.

[28] IATA. *Airport Development Reference Manual.* Montreal, 9th edition, 2004. ISBN 978-92-9195-086-7.

[29] K. C. James and M. Bhasi. Development of model categories for performance improvement studies related to airport terminal operations. *Journal of Simulation*, 4(2):98–108, June 2010. ISSN 1747-7778. doi: 10.1057/jos.2009.27. URL `https://www.tandfonline.com/doi/full/10.1057/jos.2009.27`.

[30] S. Janssen, A.-N. Blok, and A. Knol. *AATOM - An Agent-based Airport Terminal Operations Model.* Delft University of Technology, Apr. 2018. URL `https://research.tudelft.nl/en/publications/aatom-an-agent-based-airport-terminal-operations-model`.

[31] S. Janssen, A. Sharpanskykh, and R. Curran. AbSRiM: An Agent-Based Security Risk Management Approach for Airport Operations. *Risk Analysis*, 39(7):1582–1596, 2019. ISSN 1539-6924. doi: 10.1111/risa.13278. URL `http://onlinelibrary.wiley.com/doi/abs/10.1111/risa.13278`.

[32] S. Janssen, A. Sharpanskykh, and R. Curran. Agent-based modelling and analysis of security and efficiency in airport terminals. *Transportation Research Part C: Emerging Technologies*, 100:142–160, Mar. 2019. ISSN 0968-090X. doi: 10.1016/j.trc.2019.01.012. URL `https://linkinghub.elsevier.com/retrieve/pii/S0968090X1830809X`.

[33] S. Janssen, A. Sharpanskykh, R. Curran, and K. Langendoen. AATOM: An Agent-Based Airport Terminal Operations Model Simulator. In *Proceedings of the 2019 Summer Simulation Conference (SummerSim '19)*, page 12, Berlin, July 2019. Assoc Computing Machinery.

[34] S. Janssen, A. Sharpanskykh, R. Curran, and K. Langendoen. Using causal discovery to analyze emergence in agent-based models. *Simulation Modelling Practice and Theory*, 96:101940, Nov. 2019. ISSN 1569190X. doi: 10.1016/j.simpat.2019.101940. URL `https://linkinghub.elsevier.com/retrieve/pii/S1569190X19300735`.

[35] S. Janssen, R. van der Sommen, A. Dilweg, and A. Sharpanskykh. Data-Driven Analysis of Airport Security Checkpoint Operations. *Aerospace*, 7(6):69, May 2020. ISSN 2226-4310. doi: 10.3390/aerospace7060069. URL `https://www.mdpi.com/2226-4310/7/6/69`.

[36] L. Jia, R. Alizadeh, J. Hao, G. Wang, J. K. Allen, and F. Mistree. A rule-based method for automated surrogate model selection. *Advanced Engineering Informatics*, 45:101123, Aug. 2020. ISSN 1474-0346. doi: 10.1016/j.aei.2020.101123. URL `https://linkinghub.elsevier.com/retrieve/pii/S1474034620300926`.

[37] S. Kalakou and F. Moura. Analyzing passenger behavior in airport terminals based on activity preferences. *Journal of Air Transport Management*, 96, Sept. 2021. ISSN 0969-6997. doi: 10.1016/j.jairtraman.2021.102110. URL `https://linkinghub.elsevier.com/retrieve/pii/S0969699721000934`.

[38] L. Kocis and W. J. Whiten. Computational investigations of low-discrepancy sequences. *ACM Transactions on Mathematical Software*, 23(2):266–294, June 1997. ISSN 0098-3500. doi: 10.1145/264029.264064. URL `https://dl.acm.org/doi/10.1145/264029.264064`.

[39] D. P. Kuttichira, S. Gupta, C. Li, S. Rana, and S. Venkatesh. Explaining Black-Box Models Using Interpretable Surrogates. In *PRICAI 2019: Trends in Artificial Intelligence*, volume 11670, pages 3–15. Springer International Publishing, Cham, 2019. ISBN 978-3-030-29907-1. doi: 10.1007/978-3-030-29908-8_1. URL `http://link.springer.com/10.1007/978-3-030-29908-8_1`.

[40] C. Q. Lam. *Sequential Adaptive Designs In Computer Experiments For Response Surface Model Fit*. PhD thesis, Ohio State University, 2008. URL http://rave.ohiolink.edu/etdc/view?acc_num=osu1211911211.

[41] F. Lamperti, A. Roventini, and A. Sani. Agent-based model calibration using machine learning surrogates. *Journal of Economic Dynamics and Control*, 90:366–389, May 2018. ISSN 0165-1889. doi: 10.1016/j.jedc.2018.03.011. URL https://linkinghub.elsevier.com/retrieve/pii/S0165188918301088.

[42] H. Liu, Y.-S. Ong, and J. Cai. A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design. *Structural and Multidisciplinary Optimization*, 57(1):393–416, Jan. 2018. ISSN 1615-147X. doi: 10.1007/s00158-017-1739-8. URL http://link.springer.com/10.1007/s00158-017-1739-8.

[43] J. L. Loeppky, J. Sacks, and W. J. Welch. Choosing the Sample Size of a Computer Experiment: A Practical Guide. *Technometrics*, 51(4):366–376, Nov. 2009. ISSN 0040-1706. doi: 10.1198/TECH.2009.08040. URL http://www.tandfonline.com/doi/abs/10.1198/TECH.2009.08040.

[44] S. M. Lundberg and S.-I. Lee. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, volume 30, Long Beach, CA, USA, 2017. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

[45] C. Macal and M. North. Tutorial on agent-based modeling and simulation. In *Proceedings of the Winter Simulation Conference, 2005.*, pages 2–15, Dec. 2005. doi: 10.1109/WSC.2005.1574234. ISSN: 1558-4305.

[46] L. Magalhães, V. Reis, and R. Macário. A new methodological framework for evaluating flexible options at airport passenger terminals. *Case Studies on Transport Policy*, 8(1):76–84, Mar. 2020. ISSN 2213-624X. doi: 10.1016/j.cstp.2018.03.003. URL https://linkinghub.elsevier.com/retrieve/pii/S2213624X18300749.

[47] I. E. Manataki and K. G. Zografos. Development and Demonstration of a Modeling Framework for Airport Terminal Planning and Performance Evaluation. *Transportation Research Record: Journal of the Transportation Research Board*, 2106(1):66–75, Jan. 2009. ISSN 0361-1981. doi: 10.3141/2106-08. URL http://journals.sagepub.com/doi/10.3141/2106-08.

[48] A. Mekic, S. S. Mohammadi Ziabari, and A. Sharpanskykh. Systemic Agent-Based Modeling and Analysis of Passenger Discretionary Activities in Airport Terminals. *Aerospace*, 8(6):162, June 2021. doi: 10.3390/aerospace8060162. URL https://www.mdpi.com/2226-4310/8/6/162.

[49] C. Molnar. *Interpretable Machine Learning*. Lulu, 2nd edition, Feb. 2019. ISBN 978-0-244-76852-2. URL https://christophm.github.io/interpretable-ml-book/.

[50] M. Naser. An engineer's guide to eXplainable Artificial Intelligence and Interpretable Machine Learning: Navigating causality, forced goodness, and the false perception of inference. *Automation in Construction*, 129, Sept. 2021. ISSN 0926-5805. doi: 10.1016/j.autcon.2021.103821. URL https://linkinghub.elsevier.com/retrieve/pii/S0926580521002727.

[51] C. Nóbrega and L. Marinho. Towards explaining recommendations through local surrogate models. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1671–1678, Limassol Cyprus, Apr. 2019. ACM. ISBN 978-1-4503-5933-7. doi: 10.1145/3297280.3297443. URL https://dl.acm.org/doi/10.1145/3297280.3297443.

[52] H. Noble and R. Heale. Triangulation in research, with examples. *Evidence Based Nursing*, 22(3):67–68, July 2019. ISSN 1367-6539, 1468-9618. doi: 10.1136/ebnurs-2019-103145. URL https://ebn.bmj.com/lookup/doi/10.1136/ebnurs-2019-103145.

[53] P. Pao-Yen Wu and K. Mengersen. A review of models and model usage scenarios for an airport complex system. *Transportation Research Part A: Policy and Practice*, 47:124–140, Jan. 2013. ISSN 0965-8564. doi: 10.1016/j.tra.2012.10.015. URL https://linkinghub.elsevier.com/retrieve/pii/S0965856412001541.

[54] T. Patel and W. Wilkes. Strikes and Labor Shortages Leave European Airports in Chaos. *Bloomberg*, June 2022. URL https://www.bloomberg.com/news/articles/2022-06-09/the-travel-boom-has-caught-airlines-still-in-bust-mode-off-guard.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[56] B. Pietzsch, S. Fiedler, K. G. Mertens, M. Richter, C. Scherer, K. Widyastuti, M.-C. Wimmler, L. Zakharova, and U. Berger. Metamodels for Evaluating, Calibrating and Applying Agent-Based Models: A Review. *Journal of Artificial Societies and Social Simulation*, 23(2):9, 2020. ISSN 1460-7425. doi: 10.18564/jasss.4274. URL http://jasss.soc.surrey.ac.uk/23/2/9.html.

[57] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2006. ISBN 978-0-262-18253-9.

[58] S. Razavi, B. A. Tolson, and D. H. Burn. Review of surrogate modeling in water resources. *Water Resources Research*, 48(7), 2012. ISSN 1944-7973. doi: 10.1029/2011WR011527. URL http://onlinelibrary.wiley.com/doi/abs/10.1029/2011WR011527.

[59] M. T. Ribeiro, S. Singh, and C. Guestrin. Model-Agnostic Interpretability of Machine Learning. *arXiv:1606.05386 [cs, stat]*, June 2016. URL http://arxiv.org/abs/1606.05386.

[60] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA, Aug. 2016. Association for Computing Machinery. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939778. URL http://doi.org/10.1145/2939672.2939778.

[61] R. Roscher, B. Bohn, M. F. Duarte, and J. Garcke. Explainable Machine Learning for Scientific Insights and Discoveries. *IEEE Access*, 8:42200–42216, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2976199.

[62] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global Sensitivity Analysis: The Primer*. John Wiley, Chichester, England, 2008. ISBN 978-0-470-05997-5.

[63] E. Schulz, M. Speekenbrink, and A. Krause. A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, Aug. 2018. ISSN 0022-2496. doi: 10.1016/j.jmp.2018.03.001. URL https://linkinghub.elsevier.com/retrieve/pii/S0022249617302158.

[64] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, Jan. 2016. ISSN 1558-2256. doi: 10.1109/JPROC.2015.2494218.

[65] T. W. Simpson, D. K. J. Lin, and W. Chen. Sampling Strategies for Computer Experiments: Design and Analysis. *International Journal of Reliability and Applications*, 2(3):209–240, 2001. URL http://www.personal.psu.edu/users/j/x/jxz203/lin/Lin_pub/2001_IJRA.pdf.

[66] G. ten Broeke, G. van Voorn, A. Ligtenberg, and J. Molenaar. The Use of Surrogate Models to Analyse Agent-Based Models. *Journal of Artificial Societies and Social Simulation*, 24(2):3, 2021. ISSN 1460-7425. doi: 10.18564/jasss.4530. URL http://jasss.soc.surrey.ac.uk/24/2/3.html.

[67] V. A. Thurmond. The Point of Triangulation. *Journal of Nursing Scholarship*, 33(3):253–258, Sept. 2001. ISSN 1527-6546, 1547-5069. doi: 10.1111/j.1547-5069.2001.00253.x. URL https://onlinelibrary.wiley.com/doi/10.1111/j.1547-5069.2001.00253.x.

[68] B. Timmins and K. Austin. Heathrow flight cancellations cause queues and 'chaos'. *BBC News*, June 2022. URL https://www.bbc.com/news/business-61857008.

[69] V. Tosic. A review of airport passenger terminal operations analysis and modelling. *Transportation Research Part A: Policy and Practice*, 26(1):3–26, Jan. 1992. ISSN 0965-8564. doi: 10.1016/0965-8564(92)90041-5. URL https://linkinghub.elsevier.com/retrieve/pii/0965856492900415.

[70] T. Van Steenkiste, J. van der Herten, I. Couckuyt, and T. Dhaene. Data-Efficient Sensitivity Analysis with Surrogate Modeling. In *Uncertainty Modeling for Engineering Applications*, PoliTO Springer Series, pages 55–69. Springer International Publishing, Cham, 2019. ISBN 978-3-030-04870-9. URL https://doi.org/10.1007/978-3-030-04870-9_4.

[71] F. A. Viana. Things you wanted to know about the Latin hypercube design and were afraid to ask. In *10th World Congress on Structural and Multidisciplinary Optimization*, page 9, Orlando, USA, May 2013. URL `https://mae.ufl.edu/mdo/Papers/5176.pdf`.

[72] G. G. Wang and S. Shan. Review of Metamodeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical Design*, 129(4):370–380, Apr. 2007. ISSN 1050-0472. doi: 10.1115/1.2429697. URL `https://asmedigitalcollection.asme.org/mechanicaldesign/article/129/4/370/466824/Review-of-Metamodeling-Techniques-in-Support-of`.

[73] P. Westermann and R. Evins. Surrogate modelling for sustainable building design A review. *Energy and Buildings*, 198:170–186, Sept. 2019. ISSN 0378-7788. doi: 10.1016/j.enbuild.2019.05.057. URL `https://www.sciencedirect.com/science/article/pii/S0378778819302877`.

[74] U. Wilensky and W. Rand. *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo*. MIT Press, Cambridge, MA, USA, Apr. 2015. ISBN 978-0-262-73189-8.

[75] B. Williams and S. Cremaschi. Selection of surrogate modeling techniques for surface approximation and surrogate-based optimization. *Chemical Engineering Research and Design*, 170:76–89, June 2021. ISSN 0263-8762. doi: 10.1016/j.cherd.2021.03.028. URL `https://www.sciencedirect.com/science/article/pii/S0263876221001465`.

[76] D. H. Wolpert. What Is Important About the No Free Lunch Theorems? In P. M. Pardalos, V. Rasskazova, and M. N. Vrahatis, editors, *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, Springer Optimization and Its Applications, pages 373–388. Springer International Publishing, Cham, 2021. ISBN 978-3-030-66515-9. URL `https://doi.org/10.1007/978-3-030-66515-9_13`.

[77] T.-T. Wong, W.-S. Luk, and P.-A. Heng. Sampling with Hammersley and Halton Points. *Journal of Graphics Tools*, 2(2):9–24, Jan. 1997. ISSN 1086-7651. doi: 10.1080/10867651.1997.10487471. URL `http://www.tandfonline.com/doi/abs/10.1080/10867651.1997.10487471`.

[78] L. Yang and A. Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, Nov. 2020. ISSN 0925-2312. doi: 10.1016/j.neucom.2020.07.061. URL `https://linkinghub.elsevier.com/retrieve/pii/S0925231220311693`.

[79] R. Yondo, E. Andrés, and E. Valero. A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses. *Progress in Aerospace Sciences*, 96:23–61, Jan. 2018. ISSN 0376-0421. doi: 10.1016/j.paerosci.2017.11.003. URL `https://www.sciencedirect.com/science/article/pii/S0376042117300611`.

[80] T. Yu and H. Zhu. Hyper-Parameter Optimization: A Review of Algorithms and Applications. *arXiv:2003.05689 [cs, stat]*, Mar. 2020. URL `http://arxiv.org/abs/2003.05689`.

II

Literature Study

Previously graded under AE4020

1 Introduction

As air traffic is characterized by a rather stable global growth rate of approximately 4% per annum, the airline industry has long been known for its attractive long-term prospects [21]. Several scholars use it as a main argument why their studies in this area are relevant [e.g., 1, 6, 15]. While the argument is sound, abrupt events such as the outbreak of the COVID-19 pandemic also show its vulnerability — according to the International Civil Aviation Organization, in 2020, the total number of passengers dropped about 60% in comparison with the year before [37]. Despite the industry has shown great forces of resilience to past 'black swan' events like the 9/11 attacks or the 2008 financial crisis, a 60% decline is unprecedented. Now more than ever, it proves the need for the sector to make their operations more agile and lean. They have to react quickly to unforeseen events and adapt to the new circumstances.

Airports are a key element in the system: they enable air transport activities by providing the necessary infrastructure and services. While they generally have a rather inflexible spending pattern due to their relatively high fixed costs, Graham [33] has calculated that staffing and outsourcing of services account for more than 50% of their total expenditure. Airports should therefore strive to operate as efficiently as possible. That is, minimizing operating expenses while maximizing the level of service. This is not only to ensure an optimal usage of resources, but also to meet the ever-changing customer demand for non-aeronautical terminal activities, which are an important source of their revenue [54, 68].

A logical follow-up question is how airports can optimize their complex terminal operations. Namely, this is not as straightforward as it may sound. The issue has been the subject of many studies and several modeling approaches have already been explored. Especially in regard to operational planning and design, agent-based simulation models are most commonly used according to the meta-study by Pao-Yen Wu and Mengersen [74]. One of the main reasons for the popularity of such models is that they are highly detailed without compromising the complexity and emergent properties of the overall system [68]. Notwithstanding, they require a lot of computational power, which can be a limiting factor as the scale of the simulation increases. Therefore, in addition to developing such detailed models, researchers have considered the possibilities of surrogates. These models mimic the original ones by means of so-called black-box functions [11]. Fundamentally, they are subject to a dichotomy between savings in computational time and their level of accuracy [28]. Hence, as long as the reduced computational resources justify a certain lower level of accuracy, they can be a viable alternative to the original model [78].

With that knowledge, and based on the notion that there are several lacunae in the existing research, this thesis will further explore the possibilities of surrogate modeling. The focal model is AATOM, which is the abbreviation for agent-based airport terminal operations model. It was recently designed, built and validated by Janssen [46] at the Faculty of Aerospace Engineering at Delft University of Technology, and has been further developed ever since. AATOM is known for its high-fidelity to the actual terminal system, although it suffers from large computational requirements. Our research is therefore an extension to the work of De Leeuw [20], who was first to develop surrogates of the simulation model. Yet it goes beyond proving the concept, as in addition to improving the current accuracy, the aim is also to use the meta-models in gaining relevant insights into the dynamics of the underlying system. Such analyses can be rather challenging with original agent-based models, because on top of the computational burden, they reveal the statistics only a posteriori [60]. To achieve a thorough understanding of the model behavior, synergies with the rapidly emerging research on explainable artificial intelligence will be exploited, alongside more traditional approaches. This is possible since the distinct disciplines share an important common interest, which is to interpret and explain the reasoning behind machine learning algorithms [58, 72, 96]. Applying these methods to the surrogates will identify which rules, variables and characteristics of airport terminal operations are most decisive, leading to a better understanding of the system. Moreover, it will also demonstrate the further potential of meta-modeling in this direction. Altogether, the objective of the research can be summarized as to accurately abstract and explain the dynamics of complex airport terminal operations by means of creating high-performing and interpretable surrogate models based on a detailed and validated agent-based terminal simulation model. The outcome of the thesis is mainly relevant for airport and airline managers, as it leads to detailed insights into terminal processes and provides them with an efficient decision-making tool. Put differently, it enables them to act agile and adapt quickly to the flight schedule, thereby maximizing service against operating expenses. Furthermore, the research is also interesting for engineering applications, such as in the design of a new airport terminal. These are exercises with many degrees of freedom, requiring a lot of computations. Hence, fast calculation models that are both accurate and interpretable are never a superfluous luxury.

This report follows after the project plan and takes the first essential steps of the actual research. It presents the literature survey, in which the state-of-the-art academic knowledge on the multidisciplinary subject has been critically reviewed. Chapter 2 is first and discusses the key elements of modeling airport terminal operations. The main topics include a description of the general passenger terminal characteristics, a brief overview of existing models and an elaborate explanation of AATOM. Thereafter, chapter 3 continues by expounding the principles behind surrogate modeling. It sets the stage with a succinct introduction, after which it reviews common sampling strategies, promising black-box architectures for the meta-models, methods to optimize their hyperparameters and most importantly, validation metrics. The chapter concludes with an overview of the ultimately selected methodology for creating the surrogates. Having access to accurate meta-models is one thing, but their usage can be extended even further for the interpretation and understanding of the underlying system. The possibilities are explained and assessed in chapter 4, distinguishing between model-agnostic and model-specific approaches. Finally, the literature survey has naturally led to the identification of some knowledge gaps in the scientific literature. These are the lacunae where our research intends to contribute. The proposal is presented in chapter 5, which consists of the objective and research questions, a motivation on the relevance of the project, and an extensive planning to translate the intentions into actual work packages. The latter is done by means of a Gantt chart.

2
Modeling Airport Terminal Operations

With airports being a key element in the overall air transport system, this chapter takes a closer look at their activities. First, section 2.1 discusses the characteristics of an airport terminal, which is central in passenger handling. Due to the inherent complexity resulting from natural human behavior, there has been great interest in modeling the operations inside terminals. Therefore, section 2.2 follows with a brief overview of leading models in the scientific literature. Lastly, we elaborate in section 2.3 on one particular model that was recently developed by Janssen [47] at the Aerospace Engineering Faculty of the Delft University of Technology. The agent-based airport terminal operations model, henceforth abbreviated as AATOM, will be extensively used further in the research.

2.1. The Characteristics of an Airport Terminal

Air transport is a highly international business, crossing many national borders. In this context, substantial efforts have been made by several organizations to standardize the sector. For airports, the International Air Transport Association (IATA) drafted the Airport Development Reference Manual as the industry standard [see 44]. A direct result of its success is that most international airports have actually a very similar design. A simplified high-level visualization of a typical layout is depicted in Figure 2.1, based on information from De Neufville et al. [21], IATA [44], Sturdivant and Chong [93]. For clarity purposes, the legend splits up airport and terminal building elements in two categories. The former is more holistic and contains the most important parts of an airport. IATA generally distinguishes between airside and landside areas [44]. The airside area

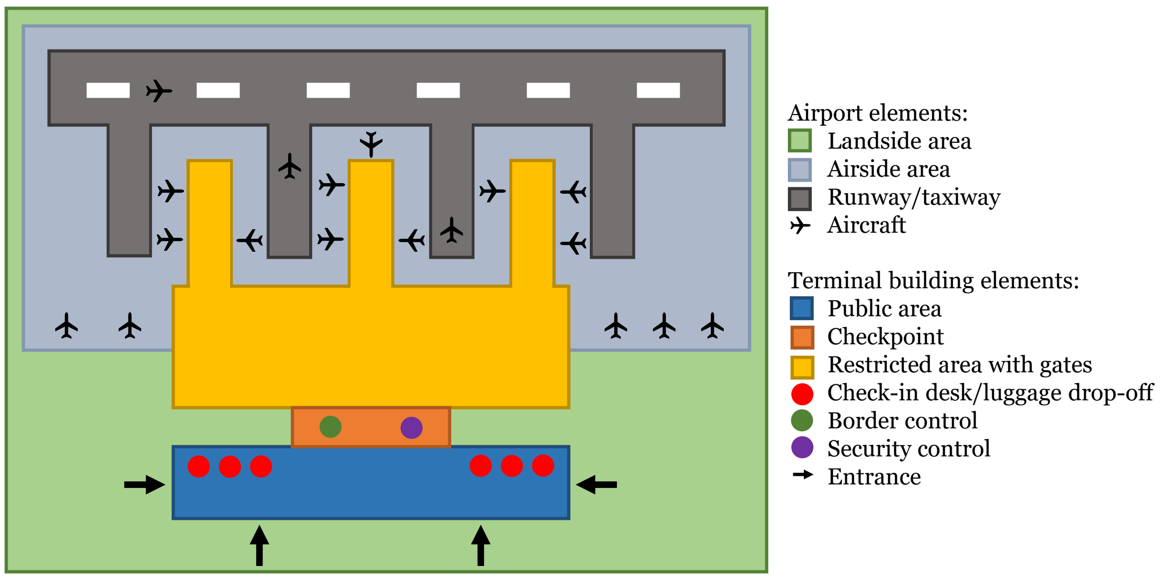


Figure 2.1: High-level visualization of a typical international airport [based on information from 21, 44, 93].

contains the infrastructure which directly relates to the operations of aircraft, so this includes runways, taxiways, parking areas, etc. It makes sense that the area is well secured and therefore inaccessible to the public. On the contrary, the landside area accommodates the infrastructure to facilitate passengers and cargo handling. This includes terminal buildings, but hotels, public and private transport facilities are also part of it. Thus, the landside area consists of the elements that connect the outside world with air transport. The second category in the legend of Figure 2.1 contains the elements of a passenger terminal building. It consists of two areas: a publicly accessible one and one with restrictions [93]. Typical elements of the public area are check-in counters, baggage drop-off points, shops, restaurants, etc. The restricted or sterile area can only be accessed after going through the checkpoint, where a security screening is performed and if necessary also an inspection of the passport [6]. Typical elements of this area are of course the gates, but there are also lounges, shops, restaurants, waiting places, etc.

The literature describes three types of flows in a terminal, namely departing, transferring and arriving passengers [19]. In current research, we focus on the departure flow to limit the scope. Passengers in this flow carry out a number of actions with the ultimate goal of catching a flight. Using information from Alodhaibi et al. [4], Andreatta et al. [6], Kalakou and Moura [54], a flowchart is drawn in Figure 2.2 with the general departure process. Note that the colors are consistent with previous Figure 2.1 to link the actions with a physical location at the airport. It starts with passengers arriving at the passenger terminal. Since most airlines offer online check-in services, they may have already checked-in before arrival. These passengers can directly proceed to the checkpoint if they do not need to drop-off luggage for the cargo hold. I.e., they are traveling without or only carry-on baggage for the cabin. Passengers who have not checked-in beforehand can do so at the counters in the terminal. This is generally also the place where luggage can be dropped-off for the cargo hold. Next, passengers must go through the checkpoint to access their gate in the restricted area. Regardless the destination, everyone is subject to a security screening to ensure no illegal items are taken on board an aircraft. The carry-on luggage is scanned with an X-ray sensor, while the passengers themselves have to walk through a metal detector [21]. Further tests can be done at random or if dangerous goods are suspected. After the security check, the next step depends on the destination. For domestic flights, passengers can go directly to their gate where they will embark the aircraft. However, for international flights, they must first pass through the border control unit where their passports are checked. Only thereafter they can proceed to their gate. Notice that there is a special situation in Europe according to the Schengen Treaty, which aims to abolish border controls between participating countries [44]. Consequently, seemingly international flights can in fact be treated as a domestic flight as no passport inspection is required. Finally, keep in mind that the flowchart in Figure 2.2 only shows the essential actions passengers need to do in order to catch a flight. More than ever, the so-called non-aernautical activities are getting more attention [54]. This includes retail shopping, dining, drinking, and other leisure activities. So in between the essential actions, passengers are free to spend their time as they wish. There are plenty of options, both in the public and the restricted area.

To conclude the section, special attention is paid to Rotterdam The Hague Airport, which is commonly abbreviated as RTHA. With approximately two million passengers per year, it is the third largest airport in the Netherlands [82]. They offer flights to about 50 destinations across Europe. As a regional airport, services are not only provided for scheduled and charter flights, but also for business and government-related air traffic [82]. To get an idea of the airport terminal, Figure 2.3 draws a simplified map of the recently renovated departure hall [83]. It shows the most relevant elements from the security check onward, which in the picture is called the 'central security filter'. Thereafter, gates 1 to 6 are for flights with destinations in the Schengen area, while gates 7 to 11 are for destinations outside the Schengen area; so a passport inspection is only required to access the latter gates. Furthermore, the non-aeronautical activities in the departure hall are also

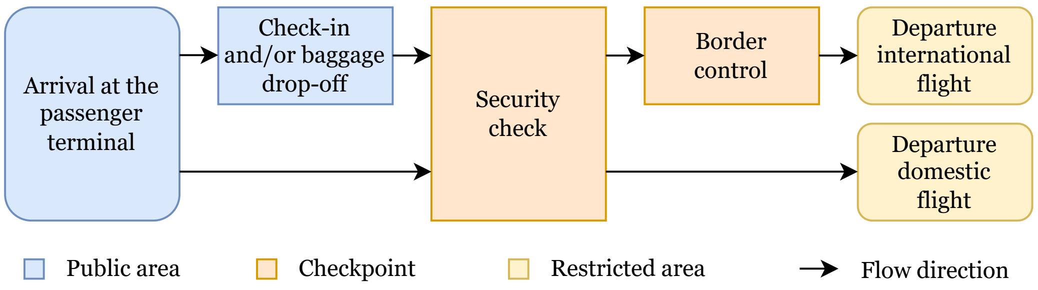

Arrival at the passenger terminal
Check-in and/or baggage drop-off
Security check
Border control
Departure international flight
Departure domestic flight
Public area
Checkpoint
Restricted area
Flow direction

Figure 2.2: Flowchart of outbound passengers [based on information from 4, 6, 54].

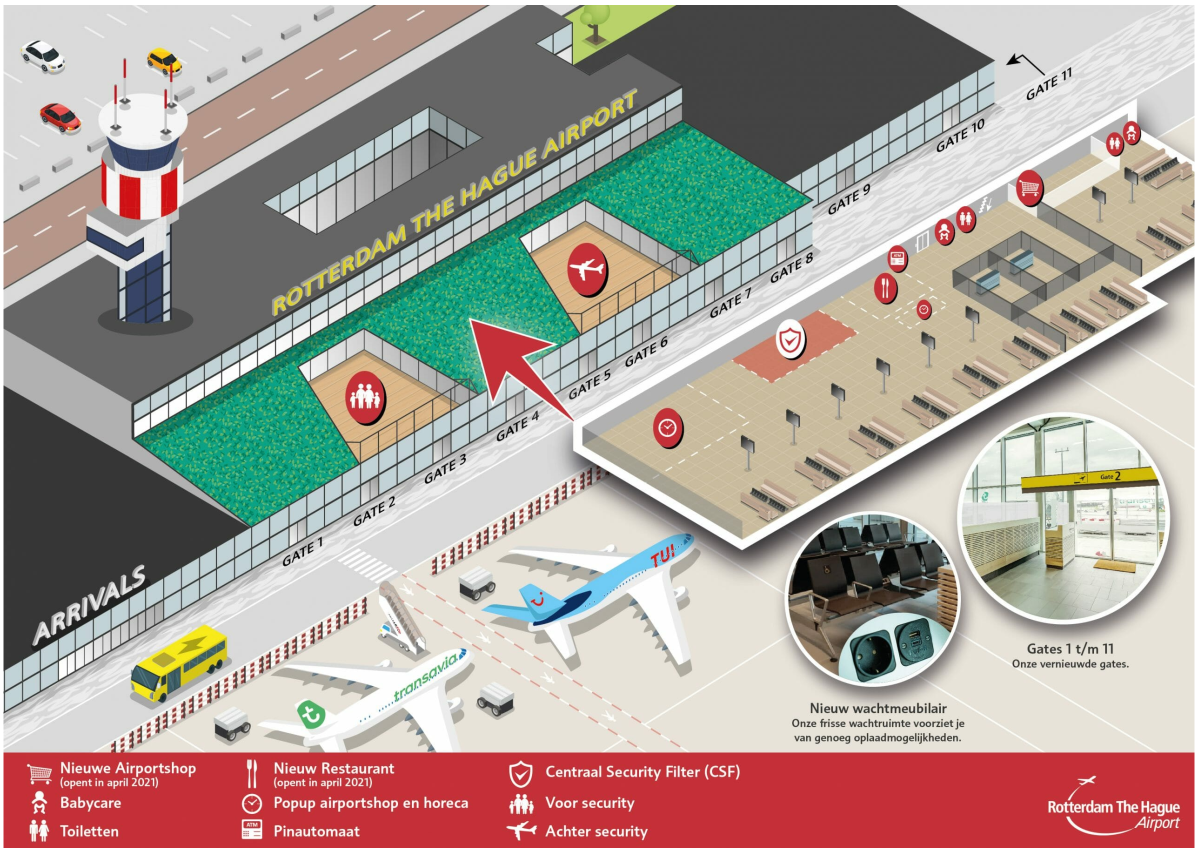

ROTTERDAM THE HAGUE AIRPORT
ARRIVALS
GATE 1
GATE 2
GATE 3
GATE 4
GATE 5
GATE 6
GATE 7
GATE 8
GATE 9
GATE 10
GATE 11
TUI
transavia
Gate 2
Gates 1 t/m 11
Onze vernieuwde gates.
Nieuw wachtmeubilair
Onze frisse wachtruimte voorziet je van genoeg oplaadmogelijkheden.
Nieuwe Airportshop (opent in april 2021)
Babycare
Toiletten
Nieuw Restaurant (opent in april 2021)
Popup airportshop en horeca
Pinautomaat
Centraal Security Filter (CSF)
Voor security
Achter security
Rotterdam The Hague Airport

Figure 2.3: Map of the Rotterdam The Hague Airport departure hall [83].

depicted, such as restaurants and shops. The reason for the special attention to RTHA is because the airport is selected as the experimental layout for current research. From a theoretical point of view, this choice is not relevant to the outcome of the study. After all, apart from a few calibration aspects, a different layout does not change the fundamental principles in an airport. It is thus a matter of choosing an application for the design of experiments. Notwithstanding, there are practical reasons at play. The first one is the availability of data and insights from an earlier study (see Janssen et al. [52]). This includes historical flight details, a detailed map of the terminal layout, insights into passenger types, etc. Such data is crucial to create a model that stays close to reality. In addition to that, previous study by De Leeuw [20] used a simple fictitious configuration for the proof of concept. To push the boundaries and see whether the findings still hold in more complex systems, we have the ambition to use a real airport layout. The challenge here will be to cleverly limit the increased computational burden, but this is not insurmountable. Altogether, the availability of information and its increased complexity makes RTHA a great choice for the experimental layout.

2.2. A Brief Overview of Existing Models and Related Work

Just as airports are central to the overall air transport system, terminals in turn form the heart of airport operations. They are known for their complexity resulting from many dynamic interactions with several stakeholders [19]. Not only does this lead to conflicting interests, it also involves a large degree of stochasticity through natural cognitive behavior and the random occurrence of unplanned events. In the literature, a system with such emergent characteristics is often called a 'sociotechnical system' [e.g., 68, 74]. The fact that they are difficult to understand makes them a challenge for decision-makers, both from a managerial and an engineering perspective. It proves the need for proper airport terminal activity modeling, as heuristics are no longer sufficient and it is simply not possible to test all scenarios that may arise in practice [66]. Terminal process modeling has long been a popular research topic, as its design directly affects human behavior. Especially with the rise of the air transport sector in the second half of last century, it received more scholarly attention. Therefore, this section examines key concepts in the literature to provide an overview of currently existing models and it additionally identifies interesting directions in academic research.

One of the earlier review studies about terminal operation modeling is Tosic [94]. At the end of last century, they argue that a lot of work has been done along with the emergence of the sector, but that major knowledge gaps still occur. These gaps mostly relate to planning issues, which become inherently more complex with growing traffic patterns. The study therefore presented a comprehensive overview of the state-of-the-art, so that academics could continue to build knowledge about airport terminals. Interestingly, the author divided the models based on 7 topics: one to forecast the demand, 5 topics that are tied to a specific physical location or process in the terminal building (e.g., service counters, luggage handling, assignment of gates, and so on) and one that models the whole terminal. Properly forecasting passenger demand is important for planning purposes, affecting short to long-term decisions. More recent literature categorizes this under strategic, tactical and operational decision-making [45, 65]. Strategic decisions are long-term design choices for an airport under development. For example, this can be to determine the size of the terminal building or the number of runways based on a prospected number of passengers. Furthermore, tactical decisions are generally made in the medium-term. Examples of such decisions are the number of check-in counters or the number of walk-through metal detectors at the security checkpoint. Finally, operational decisions are concerned with actually running an airport. They therefore have a short-term character (e.g., crew scheduling). According to Tosic [94], the passenger demand is commonly forecasted using traditional statistics. This involves finding statistical relations between parameters and analyzing demand patterns over time. Of course, several methods exist with varying degrees of difficulty. The findings are then extrapolated to future passenger numbers, which in turn is used for strategic, tactical and operational planning.

The second category in the meta-study by Tosic [94] are the models for a specific physical location or process in and around the airport terminal. While this is of course rather broad, most relevant to current research are the methods concerned with modeling the flow of passengers. Therefore, other applications such as gate assignment or the design of luggage transportation are not considered further. A notable finding is that the vast majority of the methods intend to infer about the service quality of the processes, which is commonly expressed in waiting times, lengths of queues, etc. Later research mentioned an ordinal level of service scale from A to F, which goes from excellent service to a complete breakdown of the process from a passenger point of view (e.g., see Table 1 in Andreatta et al. [6]). The scale can be applied to various terminal processes with expected standards defined by IATA in their Airport Development Reference Manual [44]. Regarding passenger flows, it is thus not surprising that queuing theory plays a big role in this category, as it allows for a direct calculation of the characteristics that define a queue. In the development of the theory, scholars make an important distinction between stochastic and deterministic approaches [74, 94]. The former incorporates the random factors and uncertainties where terminal processes are faced with, such as human behavior. For example, one never knows exactly when passengers arrive or how long a security check takes. Stochastic approaches are useful here because they can model processes with statistical distributions rather than specific values. The options are manifold, but the meta-study by Tosic [94] mentions for example a Poisson distribution for passenger arrivals. Up to date, this is still very relevant as can be seen in the study of Mekic et al. [68], where Table A.1 overviews the parameters and distributions of their model. To give an example, a Normal distribution is used for the time of the check-in process. Statistical distributions can be obtained by gathering actual data of a process at the airport (for RTHA, see the analysis by Janssen et al. [52]). On the other hand, deterministic approaches may also be used in queuing theory. These models do not contain any random factors, so under the same conditions, they will always have the same output values. Both approaches have their advantages and disadvantages; stochastic models are generally closer to reality and therefore more accurate, however, the downside is that they are more complicated compared to deterministic models. The purpose of modeling is often not only to achieve high levels of accuracy, but also to use them for understanding the underlying process [94]. Depending on the purpose, one can thus opt for one or the other approach.

Terminal processes are generally modeled in two ways: either analytically or through simulation — not to be confused with the former discussion on stochastic or deterministic approaches [66]. In analytical models, the relations between variables are expressed in an abstract and mathematical manner. Processes are thus formulated with one or more equations. As a result, it is often possible to obtain exact solutions. The benefit is that these models are very quick, precise and not overly complicated. However, this comes at the cost of accuracy, as assumptions are required to construct the expressions in the first place [66]. An example of analytical models are the methods from the field of operations research, such as an objective function which trades off different units of cost under certain constraints to ultimately arrive at an optimal solution. Tosic [94] mentions several applications in the design of passenger routes through a terminal building. Another example is mentioned by Pao-Yen Wu and Mengersen [74], where stochastic programming is applied to model the service time of a certain terminal process in order to estimate the optimal number of open desks (e.g., think of the check-in process). Alternatively, simulation-based models are different in the sense that they are not made up of mathematical expressions. Instead, solutions are obtained in a numerical manner. The benefit is that they are capable of modeling much more complex structures using less simplifying assumptions. Consequently, this leads to more realistic and faithful solutions with a better conservation of a system's emergent properties. The downside, however, is its greater computational requirements, as many simulation runs are required to cancel out numerical effects. In addition, they generally require more input data and information about the underlying system, making them more intense to use [66]. The literature again discusses several applications, like a Monte Carlo method for a simulation of the check-in process or an agent-based model to simulate congestions at different locations in an airport terminal [74, 94]. So, as with the previous discussion, scholars can choose between analytical or simulation-based approaches depending on their needs. When sufficient information about the system is available in combination with a computational budget, one will generally choose a simulation-based approach if the system entails a certain degree of complexity.

The last category in the study by Tosic [94] are models that aim to represent multiple processes simultaneously or even the entire terminal building. That is useful as it enables one to consider the system as a whole, instead of isolating separate processes. In the end, they all contribute to the emergent properties of an airport terminal. One thing which immediately stands out is that almost all examples are simulation-based models. However, this actually makes sense as it can be rather challenging for analytical models to capture much of the complexity associated with an entire sociotechnical system. The author discusses several applications in the literature: from models that consider a particular flow (e.g., departure or arrival) to those for the entire terminal. The benefit of the latter is that they take into account the interactions between different subsystems. Indeed, terminal operations are interrelated with flight operations, ground handling, etc. Notwithstanding, it is not always beneficial to consider as much from the system as possible. Scientists often make a distinction between microscopic, mesoscopic and macroscopic models [66, 74]. This is particularly relevant for holistic models, as it is a categorization based on the level of detail. Microscopic models entail high levels of detail, even to the extent that passengers and their interactions are considered individually [74]. Moreover, this also includes the interactions between passengers and the environment in which they find themselves. They are therefore extremely useful if one requires accurate information about certain operational challenges [66]. The opposite are macroscopic models. These models approach the issues from a higher level. In other words, they are more aggregate, leaving out unnecessary details. The advantage is that they allow for a straightforward but complete modeling approach, at the expense of accuracy [66]. Nonetheless, that is sufficient for many applications: e.g., Manataki and Zografos [66] mentioned its use for strategic decisions that have a longer horizon. The difference between microscopic and macroscopic models is however not black and white. Pao-Yen Wu and Mengersen [74] mentioned mesoscopic models, which are situated in between. These models still focus on higher level and more aggregate interactions, although they do so in more detail compared to macroscopic models. For example, mesoscopic models do not consider individual passengers, but rather focus on the characteristics of the passenger flow itself. Such models are useful because they combine the best of both worlds; i.e., they can be much more efficient than microscopic models without necessarily omitting all the details. It makes them ideal candidates to plan operational processes [74]. For example, an optimal number of available check-in counters can be determined by analyzing the characteristics of passenger flows; mesoscopic models can easily distinguish between peak or off-peak traffic patterns. For this purpose, there would be no need to consider passengers on an individual basis.

Besides a distinction at the level of detail, the more recent meta-study of Pao-Yen Wu and Mengersen [74] provides a unique overview of existing models that are differentiated according to their use case. They identified four purposes, viz. models to plan the capacity of terminal processes, models to design the operations behind it, models to evaluate security risks and models to assess the overall performance of an airport. Interestingly, this is where above discussions come together, as the categorization reveals quite clearly which approach scholars prefer per use case. The capacity of terminal processes is usually modeled using analytical approaches, either stochastic or deterministic. When looking further into the models themselves, this actually makes sense, because their purpose is e.g. to analyze the throughput of different layout options or to evaluate the capacity of a hypothetical terminal process. Clearly, the models are used to make decisions about longer-term planning issues — decisions with a strategic or tactical horizon. High-level analytical approaches are seemingly sufficient to address these challenges. This is definitely not the case for models that are used in operational design. Pao-Yen Wu and Mengersen [74] show that the vast majority of models in this domain are agent-based. That is a microscopic or mesoscopic simulation approach which is able to model the behavior of individual passengers with the interactions between them and the environment. Hence, agent-based modeling is a very suitable approach if one requires detailed information on a certain terminal process, especially if it is largely affected by human behavior. Third, stochastic approaches play a big role in evaluating security risks. Since these tasks inherently involve a high degree of uncertainty, probability distributions are useful to model vulnerabilities and threats to airport security. According to the meta-study, the models are applied in the evaluation of risk assessment tasks and in the development of airport security policies. The final purpose of use was to measure the performance of passenger terminals. Models in this category have not been discussed extensively before because they are rather different from the others. That is, subjective input data might be required to measure the satisfaction of passengers about an airport terminal. An elaborate review study on this topic was done by Zidarova and Zografos [110], who indeed made the important distinction between objective and subjective methods, but also between a combination of both (i.e., hybrid). Nevertheless, objective approaches in this domain are rather scarce. An example of such a method is extracting the aforementioned level of service standards from IATA [44]. They are objective because they measure the performance of terminal processes based on e.g. available space, occupancy, waiting times in queues, etc. These insights are certainly helpful, although they fail to capture the perception from the passengers' perspective. This problem is solved by using subjective or hybrid models — they do take into account data such as surveys with Likert scales, qualitative insights from interviews, etc. The models then translate the subjective data into meaningful performance metrics. For fully subjective models, this is usually done with traditional statistical tools (e.g., regression to assess key drivers) [110]. Hybrid approaches are more complicated because of a generally different objective for which the models are used. While subjective approaches infer about passengers satisfaction, hybrid ones aim to analyze passenger perception in an objective manner so that the results can be fed back to the prescribed levels of service. In that sense, their task is to identify the boundary values on subjective scales and convert them into objective service levels [110].

With that, we touched upon the main directions and approaches of airport terminal modeling in the literature. Summarizing this section, Table 2.1 lists the main dimensions from the discussion by which the models can be classified. Note that the overview is not necessarily exhaustive, although it contains the most important ones. It is clear that a lot of research has been done on the topic and that it is still ongoing. Especially the review by Pao-Yen Wu and Mengersen [74] has had a great influence on recent work. Their recommendations for future model development have been widely used ever since. For example, Nõmmik and Antov [73] used these insights to create a capacity model of a local airport in Estonia, while Alodhaibi et al. [4] designed a framework to simulate passenger flows of an international airport in Australia. More recently, albeit still in line with the trends of the meta-study, Janssen [47] have developed a versatile agent-based simulator that allows highly detailed modeling of airport terminal processes. Agent-based models became especially relevant as computing power is increasing, giving researchers the opportunity to create simulators that are meticulously close to reality. Since this model will form the backbone of current research, the following section 2.3 further elaborates on its details and working principles.

Table 2.1: Main dimensions to categorize airport terminal models.

	Process	**Nature**	**Horizon**	**Detail**	**Usage**	**Input**	**Uncertainty**
	Check-in	Simulation	Strategic	Macroscopic	Capacity	Subjective	Stochastic
	Gate assignment	Analytical	Tactical	Mesoscopic	Operations	Objective	Deterministic
	···		Operational	Microscopic	Security	Hybrid	
	Combination*				Performance		
Ref.	[94]	[66, 94]	[45, 65]	[66, 74]	[74]	[110]	[94]

*Combination refers to models that are made up of multiple terminal processes

2.3. The AATOM Simulator

At its core, AATOM is built on the philosophy and principles of agent-based modeling. Subsection 2.3.1 therefore introduces the general concepts of this modeling technique. Subsequently, the architecture and fundamentals behind the agent-based airport terminal operations model are explained in subsection 2.3.2. Lastly, subsection 2.3.3 provides an overview of the model's input and output parameters, as these will play a key role in the current research.

2.3.1. Agent-based Modeling

Fueled by increasing computational power, agent-based modeling — often abbreviated as ABM — has received quite a bit of academic attention in recent years. That is largely because it features some unique capabilities that cannot be achieved with more traditional approaches, some of which were touched upon in previous section 2.2. In essence, ABM is a microscopic bottom-up approach used in the discrete simulation of complex heterogeneous dynamical systems, which often involve a large degree of stochasticity [12]. Going into further detail, ABM is microscopic because actors are modeled on an individual basis, whereby they make decisions and perform actions independently of one another [64]. They come in different forms and internal states, making ABM a heterogeneous method. The actors operate in a certain environment where they interact not only with each other, but also with the environment [64]. The result of the interactions is that the emergent properties of the system naturally appear from the dynamics and this is rather unique. These properties can seldom be captured by traditional statistics and mathematical approaches [12, 17]. As ABM is one of the few approaches that can do so meticulously close to reality while remaining flexible, it is the epitome of modeling complex sociotechnical systems and its corresponding dynamics. Moreover, it will come as no surprise that the origin of the method lies in the simulation of human behavior and social interactions, which is known for its complexity, irrationality and stochasticity [17]. Note that the simulation part is an important aspect here; having the model is one thing, the statistics are only revealed a posteriori. The characteristics of the system are thus not known beforehand [60]. The main steps in the ABM process are therefore first to create the model, then to run the simulation and finally to obtain the outcome in order to draw conclusions about the underlying system. In practice, the simulation is performed along a discrete time dimension, although the difference between successive steps is generally so small that it differs not much from the continuous time [17]. Lastly, ABM was said to be a bottom-up approach. That is because it starts with modeling the individual actors in a certain environment. Eventually, their simulated actions and interactions lead to a sociotechnical system with emergent properties. So what actually happens is that the overall system is modeled and analyzed starting from its smallest element. To frame it in the words of Crooks and Heppenstall [17, p. 101], "local phenomena are understood and measured at a global level".

There are three main components in ABM: an environment, agents and their interactions [102]. The coherence between the three is depicted in Figure 2.4. In the environment, there are agents — technical jargon for the actors in a system — that have heterogeneous cognitive properties. This gives them the tools to make autonomous decisions, enabling them to perform actions based on a particular goal they want to achieve. This is facilitated by protocols so that they can interact and communicate during their journey, both with each other and with the environment [64]. Their intelligence also allows them to actively act and react to what is going on. In other words, they do not bluntly pursue their goals without considering the situation in the environment around them. These behavioral rules, typically in the form of mathematical expressions and if-then statements, are established based on the insights about the actual actors in the system, which may be obtained through the analysis of data, scientific publications, field experts, and so on [17]. Finally, the environment represents the space where the agents interact and try to achieve their goals [102]. This is generally the physical location of the underlying system. So in the case of an ABM for outbound passengers at an airport, the environment would be the terminal building with all of its furniture and equipment.

Lastly, the advantages of using ABM are compared against its disadvantages. The extent to which scholars are harmonious on this, is rather striking. They mention three main benefits, namely ABM's ability to catch the emergence of a complex system, its flexibility throughout the modeling process, and its natural bottom-up approach to build a larger whole out of a coherence between several small parts [12, 17, 64]. Conversely, the main drawbacks are the following. Firstly, it remains challenging to incorporate all psychological

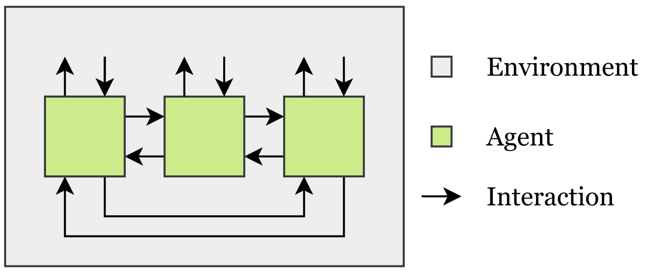


Figure 2.4: The three components of agent-based modeling [based on theory from 102].

elements of human behavior. Another challenge is to find the right balance in the level of detail which is relevant to consider. Finally, it needs high computational demands due to its microscopic nature [12, 17]. Nonetheless, according to the argumentation in Bonabeau [12] and Macal and North [64], ABM remains a very attractive modeling approach when a system has the following characteristics. Most importantly, it is trivial that agents and their interactions must play a key role in a complex environment. Especially if the interactions are difficult to understand and non-linear, ABM excels compared to other methods. Second, the same holds when the heterogeneity of agents is an important factor to take into account. Similarly, when the actions of agents are dynamic and can suddenly change based on certain events happening in the environment, scholars recommend to use ABM. Altogether, it proves to be an excellent method when one is interested in modeling a sociotechnical system. There are a myriad of examples in the literature, of which some are to model people's behavior in financial markets, crowd control, consumer behavior, the analysis of flows, and so on [12, 64]. Moving back to the scope of current research, modeling passenger flows in airport terminals would be a textbook example where the application of ABM is appropriate.

2.3.2. Model Architecture

It is now clear that airport terminals can be called sociotechnical systems which are rather challenging to model due to their inherent complexity. With this in mind, AATOM was developed from the notion that there is a lack of accessible simulation models. Existing options are not accurate enough, too generic, too difficult to use or their source code is not openly available [51]. Therefore, Janssen [47] created AATOM — a model specifically for the agent-based simulation of airport terminals. It is designed with an object-oriented philosophy, allowing scientists to easily model the associated passenger flows. This makes it a very versatile tool, as it can be completely adapted to any set of requirements. One of its main features is that it contains prebuilt components of the terminal, such as check-in desks or a security checkpoint [51]. Consequently, based on a few building blocks, the layout of an entire terminal can be built with just a few lines of code.

AATOM focuses mainly on the departing passenger flow. In accordance with Figure 2.2, the most important elements are the check-in, the security check, passport control and a set of discretionary activities [48]. The latter concerns the non-aeronautical activities that are not mandatory for passengers to catch a flight; so this includes shopping, dining, using restrooms, etc. As always, creating a model also means that a number of assumptions are necessary, as listed by Janssen et al. [48, p. 35]. The overview is self-explanatory so there will not be gone into detail, but one example is that passengers are always traveling alone. The assumptions and their respective effects should be kept in mind when using the model. Furthermore, a big advantage is that AATOM and its components are calibrated and validated with real data gathered at airports [46, 51]. This is important because it ensures that the outcomes of a simulation are close to those in reality and it gives the model validity. Since its development, the model has shown its capabilities in various studies. Some recent examples include Janssen et al. [50] on the relationship between checkpoint security and efficiency, Janssen et al. [49] on the management of airport security risks and Mekic et al. [68] with an analysis on the non-aeronautical activities and their impact on terminal operations.

The architecture of AATOM is now further explained with information from the technical report by Janssen et al. [48][1]. The description consists of three parts according to the three main components of agent-based modeling. Namely, the environment, the construction of the agents themselves and their interactions. Finally, as the model has been further developed since its introduction, the recent updates are also discussed.

The Environment

As one would expect, the environment contains all elements of an airport terminal. AATOM defines three objects in this environment, viz. a flight, an area and a physical object. We readily stress on the fact that these objects are not necessarily tangible parts of the terminal. For example, a flight does not exist physically, but is rather an abstract element in the system. The three objects are now discussed in respective order.

Traffic at airports is provided by flights. While AATOM allows for both departing and arriving flights, current research focuses on the outbound flow. Each flight leaves at a certain time with a specific number of passengers from a predefined gate at the airport. Flights are managed by the operators behind allocated check-in desks in the public area of the terminal. Which ones exactly and the number of desks is also predetermined. Note that AATOM sees the flight details and planning as is — it is thus an input of the model.

The second object of the environment is an area. As in reality, an airport terminal consists of several functional areas. These are modeled as two-dimensional bounded surfaces and can be seen as containers

[1]For the sake of readability of the literature review, it is not constantly cited, but all information about the architecture of AATOM is obtained from the technical report, unless stated otherwise.

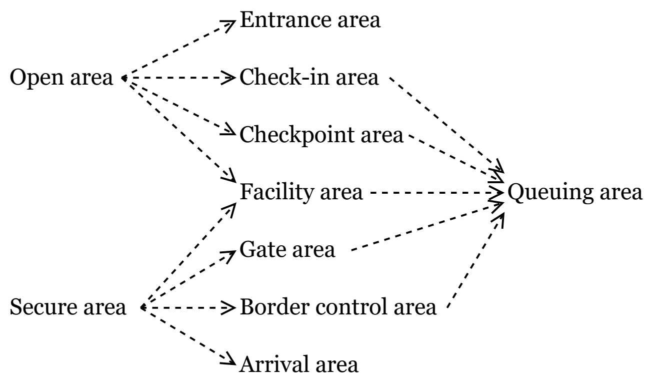


Figure 2.5: Functional areas in AATOM and their mutual relations [based on information from 48, 68].

for designated subareas or physical objects. Essentially, the areas represent the ’world’ of the agents in which they interact and perform actions. AATOM knows 10 different areas which are are depicted in Figure 2.5 together with their mutual relations — they can overlap. The visualization should be read from left to right, starting with a distinction between the open and the secure area. Figure 2.1 introduced them as respectively the public and the restricted area. The difference between the two is explained in section 2.1, so they will not be discussed further here. The open area must consist of an entrance, a check-in and a checkpoint area. The entrance is the place in the terminal where outgoing passengers arrive and incoming passengers leave. Hence from the perspective of AATOM, it defines the agent generation and removal place of the system. The check-in area is explained with Figure 2.2: passengers who have not checked-in online before arriving at the airport should do so at the counters in the open area. Passengers can access the secure area by going through the checkpoint for a security check. This process is also elaborately described in section 2.1. Facility areas can be present in the open area, but that is optional and depends on the layout of the airport. These are the places where discretionary activities take place (e.g., shopping, dining, restrooms). Furthermore, the secure area must consist of a gate and an arrival area. They are in fact the inverse of the entrance. Namely, these are the respective places in the terminal where outbound passengers are removed when their flight departs and inbound passengers are generated when their flight arrives. If an airport facilitates international (or non-Schengen) flights, there is also a border control area to check passengers’ passports when entering or leaving the Schengen area. Again, facility areas may be present in the secure area, although it is optional and depends on the layout of the airport. The last functional area in Figure 2.5 is where passengers queue up. They await in queues until they can be assisted at the checkpoint, check-in counters, border control, facilities or the gate.

The final object of the environment are physical objects. This is a broad term which represents all physically existing elements in the airport terminal. Every object has at least three properties: whether it is a sensor, whether it is transparent and whether it is blocking. Sensors enable security officers to detect dangerous goods or illegal items at the security checkpoint. AATOM knows three different sensors, viz. an X-ray sensor, a walk-through metal detector (WTMD) and an explosive trace detector (ETD). Respectively, they observe the threat of passengers’ carry-on baggage, the presence of metal and whether there are particles of explosives. Furthermore, the transparency property determines whether agents or sensors can observe through a certain object. For example, this is not possible for a wall in the building, while it is possible for seats. Third, the blocking property of an object determines whether agents or other physical objects that block can have the same position within the passenger terminal. Apart from the earlier discussed sensors, the technical report mentions belts, queue separators, seats, walls, desks and luggage as existing physical objects in AATOM. The latter is somewhat special; it has three additional properties, namely a level of threat, its complexity and whether it is carry-on or checked baggage for the aircraft’s cargo hold.

Now that all elements of the environment have been explained, Figure 2.6 shows what the terminal layout of Rotterdam The Hague Airport looks like in AATOM. When passengers arrive at the airport through the entrance (A), they are in the public area (B). Those who have not checked-in online, can do so at the check-in counters (C) after waiting in the designated queues (I). Thereafter, all passengers can continue via other queues (J) to the security checkpoint (D) to access the restricted area. This area is split up into a departure hall (E) and an arrival hall (F). The arrival hall is not further developed as the research focuses on solely the outbound passenger flow. The departure hall has gates 1 to 6 for flights with destinations in the Schengen area and gates 7 to 11 for flights outside the Schengen area (the numbers on the map correspond to the gate numbers). All gates have seats where passengers can wait. To access the latter gates, passengers should go through border control to have their passports checked (G). Along the journey, passengers are free to make use of the facility areas for non-aeronautical activities (H). Note that the markings are not exhaustive, but rather are illustrative to clarify the main features of the map. Finally, we emphasize the similarity of the departure hall with the actual one of RTHA in Figure 2.3. Only the length of the security check queue (J) is exaggerated, but that is to accommodate all passengers in the queuing area as they could otherwise cause congestion. This would lead to spurious results and is therefore avoided by lengthening queues.

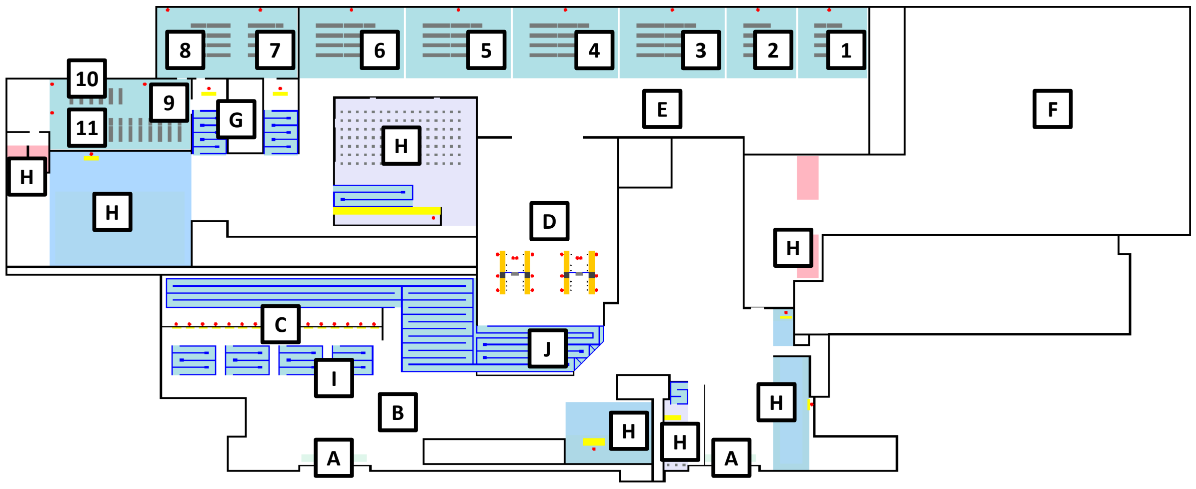


Figure 2.6: Modeled terminal layout of RTHA in AATOM [partly based on information from 51].

Agents

Agents are the key players in the environment. Three types can be defined in AATOM: passengers, operators and orchestrators. The former is trivial, operators are generally the employees in the terminal (e.g., security officers, check-in staff, cashiers, etc.) and orchestrators help with coordinating and monitoring (e.g., employees who open or close check-in counters). Agents have certain goals on which they act and interact accordingly. Behind their reasoning is a three-layer hierarchical architecture, allowing AATOM to realistically model human behavior. The architecture is visualized in Figure 2.7, which is now discussed in more detail with the operational, tactical and strategic layers[2] respectively.

The operational layer is at the bottom of the structure and is directly involved with fellow agents and the environment. Two modules are part of the layer: viz. the perception and the actuation module. The first is occupied with the perception of information. Essentially, the observations of an agent are collected and form the input of the model. The module then feeds this information to the tactical layer above. Secondly, the actions of an agent are executed by the actuation module. That is the model's output. For this, the tactical layer gives specific instructions. The type of actions can be broad, such as walking[3] or communicating. An important note is that not all agents can do the same observations or actions. For example, passengers cannot check-in themselves at the airport counters, only an operator agent who is responsible for the check-in process can do so. This is, inter alia, what introduces the heterogeneity between different agents in the system — just like in reality.

The tactical layer is in between the other two. It consists of four modules mainly concerned with interpreting perceptions, having a belief, navigating in the environment and setting out a sequence of activities. The information from the operational layer goes to the interpretation module, which interprets an agent's perceptions in collaboration with the belief module. Meanwhile, the belief module aligns itself with the interpretations if inconsistencies are noticed. So, in addition to feeding information to the strategic layer, the belief module is essentially a memory. It remembers an agent's characteristics, interpretations, activity states, action states and a plan from the strategic layer, which together make up the belief. Furthermore, the remaining two modules of the tactical layer use this information. On the one hand, the activity module combines

[2]As operators execute single activities rather than sequences (like passengers), a strategic layer is redundant and thus omitted for them.
[3]It is not explained further due to the scope of current research, but the walking of agents and how it is affected by others around them has been modeled on the basis of social forces using the research of Helbing et al. [40]

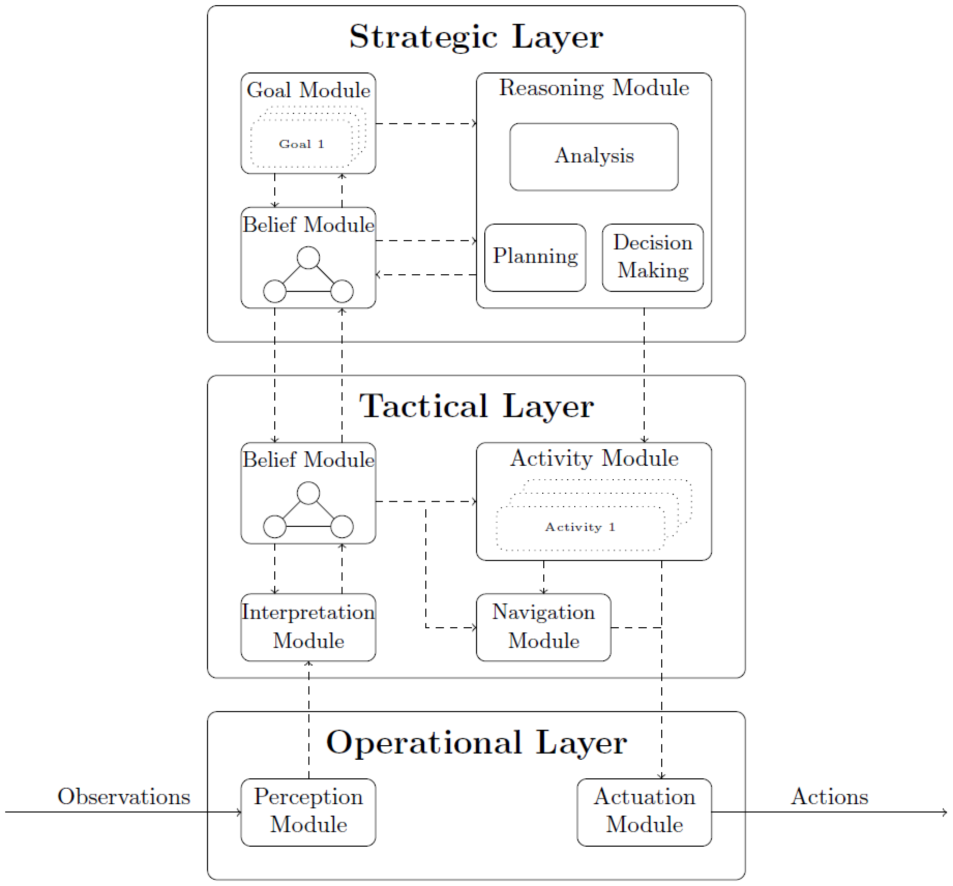


Figure 2.7: Architectural layout of agents in AATOM [48]. They operate as follows. Observations are perceived and interpreted, allowing agents to reason so that their activities can be set. Ultimately, this leads to the actuation of specific actions.

the belief with input from the strategic layer to determine which activities an agent should perform. These are then passed to the operational layer for actuation. The technical report lists various activities that operator and passenger agents can carry out. They are not covered here, but some examples are the check-in activity for outbound passengers, shopping activity for incoming and outgoing passengers, passport control for security officers, etc. On the other hand, the navigation module combines the belief with input from the activity module to navigate through the airport terminal. In line with the assumptions of the model, the navigation module provides the shortest path to an agent's target. Of course, this path must be free of any blocking objects in the environment. AATOM realizes this with the pathfinding algorithms of Cui and Shi [18] and Harabor and Grastien [36]. In addition, it is interesting to mention that agents can detect when they get stuck. If so, they will try an alternate route to their target.

The strategic layer is at the top of the architectural layout. Its main tasks are to define the goal of an agent, have a belief and do the necessary reasoning. The goal module contains the set of activities an agent wishes to accomplish. They are sequenced (e.g., outbound passengers must complete check-in before proceeding to security) and may be time-bound (e.g., outbound passengers aim to finish the gate activity before their flight's departure time). As in the tactical one, the strategic layer also has a belief module, albeit at a higher level. It only memorizes and updates an agent's interpretations, activity states and plan. There is mutual communication between the belief module and the goal module, the tactical layer and the reasoning module. In turn, the latter consists of three sub-modules: the analysis, planning and decision-making modules. The first one examines an agent's current state. Based on that, the module decides whether a new plan or a decision is required. If so, this is communicated accordingly to the respective modules. A plan is essentially a list of activities that corresponds as closely as possible to the activities in the goal module. If any, the remaining degrees of freedom are covered by the decision-making module. For example, that is to select a queue for the check-in counters if there are multiple options. Such choices are not part of the planning module.

Agent Interactions

The final component of the agent-based model is that agents can interact with the environment and with one another. Accordingly, two types of interactions are defined in AATOM. First, operator agents are responsible for handling and guiding the passengers through specific activities in the terminal. In a sense, they are the connection between a passenger and the environment. Hence, only the operators interact with the environment; passengers do not. The technical report mentions two instances, namely check-in agents managing flights and security officers using sensors. The second type are interactions between agents themselves. Mostly, that are operators interacting with passengers during the activities in the terminal. There are several examples, including border control checking passports, check-in agents checking-in passengers, security officers carrying out additional baggage checks, etc. Nevertheless, operators can also interact with colleagues. A good example is at the security checkpoint: if the X-ray operator finds a certain baggage suspicious, it orders another security officer to further examine the baggage for any illegal items.

The interactions of operators and passengers have been explained, but the discussion about agents in AATOM also mentioned a third type for coordinating and monitoring. Orchestrators are special because, depending on their goals, they can interact with just about everything. It really comes down to their tasks. To give an example, if an orchestrator manages the security checkpoint, it interacts with the environment to open or close lanes.

Further Development of the Model Architecture

After the introduction of AATOM by Janssen [47], it has been further developed ever since. In the meantime, the model is extended in two main directions. On the one hand, there is the version of Köstler [57] and van der Horst [95], which primarily focused on the profound improvement of the airport's checkpoint. Moreover, they added multiple configurations of alternative security screening setups to enhance the versatility of AATOM. On the other hand, Mekic et al. [68] expanded the model more holistically. They particularly improved the possibilities to integrate non-aeronautical activities in the simulations, although the cognitive behavior of agents was also further developed. Namely, they added basic traits and emotions to the architecture in Figure 2.7. To briefly explain the update, it leads to the reasoning module in the strategic layer being affected by both the belief module and an agent's emotions, which in turn are steered by the traits [68]. Logically, this has an impact on the other layers, and therefore on the overall cognitive behavior. In line with the arguments in section 2.1 why we opted for RTHA as the experimental layout, it makes sense to select the latter alternative by Mekic et al. [68] for our research. Up to date, it is the most advanced version of AATOM to simulate the operations of an entire passenger terminal.

2.3.3. Input and Output Parameters

Now that the architecture of the model is clear, the input and output parameters of the model are discussed as they will play an important role. First of all, we emphasize the fact that AATOM was created as a versatile tool. This means that essentially everything can be customized and adapted to the requirements of the user. Consequently, a full list of all parameters would be too comprehensive and complicated. AATOM solves this cleverly by defining calibrated presets (e.g., the distribution of the time needed for checking-in at the airport counters) [51]. For most of these settings, the standards will be used. That is justified as the purpose of current research is more to look at it from the airport managing perspective. It involves, for example, examining the relation between the number of open lanes at the security checkpoint and the average waiting time for passengers in the queue, and so on. An overview of the relevant input parameters is listed in Table 2.2. They are retrieved from the source code of the AATOM version that was developed by Mekic et al. [68]. Generally, there are two types of input parameters; those related to the flight schedule and those related to the airport strategy. Gate_t and Pax_t belong to the former type, and essentially determine how many passengers depart from which gate at what particular time. The other three belong to the latter, which defines how airport managers intend to operate the terminal. It includes the strategic deployment of staff and the settings of the call-to-gate system. More elaborate explanations are given in the table, though there is a small remark regarding the parameters of the flight schedule. While it seems that Gate_t and Pax_t are the only two inputs, notice that they are defined for every available time slot t regardless of whether a flight is actually planned. In other words, if an airport has 6 available time slots in a given time frame, AATOM requires 15 input parameters — 6 times 2 parameters for the flight schedule and 3 parameters to determine the aforementioned strategy.

Finally, similar to the input, the output parameters are listed in Table 2.3 along with an explanation. They are also retrieved from the source code of the AATOM version by Mekic et al. [68]. In principle, these parameters are key performance indicators (KPIs) because they provide insights into how the airport terminal handles its passengers. The KPIs can be divided into three categories: those for the check-in process, those for the security checkpoint and the more generic ones. On the one hand, the first two categories have indicators for the average queuing time and the throughput. On the other hand, the generic ones are the average time to reach a gate, the number of missed flights and the total expenditure of passengers during non-aeronautical activities. A last note is that AATOM allows to define and extract any KPI that a user needs. Also in this regard, it is a versatile and flexible tool. Nonetheless, the standard indicators from Table 2.3 give a good overview of the terminal performance. No other KPIs will be defined as they are considered sufficient for the current research.

Table 2.2: Relevant input parameters to define simulations in AATOM.

Input parameter	**Unit**	**Explanation**
Gate_t	[-]	An integer indicating from which gate the flight on time slot t is scheduled to depart. It can range from 0 to 11, in accordance with the layout of RTHA from Figure 2.3 and Figure 2.6. Recall that gates 1 to 6 correspond to flights with a destination inside the Schengen area and vice versa for the other gates. If it is 0, no flight is scheduled on the time slot.
Pax_t	[#]	An integer indicating how many passengers are traveling with the flight on time slot t. It is strictly positive, bound by the maximum capacity of the aircraft. If it is 0, no flight is scheduled on the time slot.
$\text{CTG}_{\text{strategy}}$	[s]	A positive real number that determines the time when passengers are called to their gate prior to the departure time. It represents the airport's call-to-gate (CTG) strategy [68].
$\text{CI}_{\text{strategy}}$	[-]	An integer that determines the number of open check-in counters over time. It represents the airport's check-in (CI) strategy. An orchestrator agent couples the number with a predefined strategy [68].
$\text{SC}_{\text{strategy}}$	[-]	An integer that determines the number of open lanes at the security checkpoint over time. It represents the airport's security check (SC) strategy. An orchestrator agent couples the number with a predefined strategy [68].

Table 2.3: Relevant output parameters from the simulations in AATOM.

Output parameter	**Unit**	**Explanation**
AvgQueueTime_{CI}	[s]	Indicates the average time that passengers wait in a queue until they can be served at an available check-in (CI) counter.
AvgQueueTime_{SC}	[s]	Indicates the average time that passengers wait in a queue until they can be served at a security checkpoint (SC) lane.
AvgTimeToGate	[s]	Indicates the average time it takes passengers to get to their gate. It is counted from the moment they arrive at the airport.
PaxCompleted_{CI}	[#]	Indicates the total number of passengers that have completed the check-in (CI) activity at the airport counters (i.e., the throughput).
PaxCompleted_{SC}	[#]	Indicates the total number of passengers that have completed the security check (SC) activity at the checkpoint (i.e., the throughput).
NumMissedFlights	[#]	Indicates the total number of passengers who could not reach their gate at the time of departure.
TotalExpenditure	[€]	Indicates the amount of money that all passengers together have spent during their non-aeronautical activities [68].

3
Surrogate Modeling

Now that the state-of-the-art models for airport terminals have been explained, this chapter examines the possibilities for surrogate modeling. Section 3.1 introduces what exactly it is and why it is useful. Moreover, a theoretical framework is also provided with the typical steps. Each of them is then further explained in a separate section. Respectively, section 3.2 discusses several methods to generate samples from AATOM, section 3.3 compares commonly used black-box algorithms for surrogate modeling, section 3.4 reviews various strategies for tuning the hyperparameters and section 3.5 elaborates on validating the surrogates. Finally, all theory is synthesized in section 3.6. This section presents the ultimately selected methodology for current research.

3.1. An Introduction to Surrogate Modeling

The previous chapter pointed out that the simulation of airport terminals using agent-based models has advantages that cannot be achieved by other approaches. The microscopic approach allows a detailed modeling of the complex sociotechnical system while preserving its emergent properties. While this is generally considered a good argument for using agent-based modeling, it is also its greatest weakness — scholars continue to push the boundaries despite currently available technology [76]. Indeed, the higher the level of detail, the greater the computational cost. This problem becomes especially apparent when the model is used for analysis, optimization or design tasks. They are inherently computationally expensive because a broad input parameter space easily leads to the so-called 'curse of dimensionality' [60]. To illustrate this with a straightforward example, if one were to test 4 parameters with 10 possibilities each, a serial process would take more than 27 hours when a single simulation run lasts 10 seconds. It proves the need for a different approach: what if scholars are willing to give up some of the accuracy in turn for a much faster algorithm? Going back to previous example, if a single simulation run now takes only 0.5 seconds instead of 10, the total process would reduce to a mere 80 minutes.

Above reasoning is one of the main arguments for surrogate modeling, which is the art of fitting black-box functions between the input and output of an original model in an attempt to accurately mimic its behavior [11, 28]. The benefit is that, once a meta-model is trained, it can approximate the output at a fraction of the computational effort that would otherwise be required. Hence, surrogates are fundamentally subject to a dichotomy between savings in computational time and their level of accuracy. In other words, as long as the reduced computational time justifies a certain lower level of accuracy, they can be a viable alternative to the original approach [28, 78]. The concept of surrogate modeling is not new and knows several applications in engineering: e.g., Forrester et al. [28] mentioned structural analysis, computational fluid dynamics, geostatistics, etc. More recently, it is also finding its way to emulate agent-based simulation models, as demonstrated in the meta-study by Pietzsch et al. [76]. De Leeuw [20] followed the trend and was the first to apply meta-modeling specifically to the AATOM simulator. The proof of concept showed that it was possible to create surrogates with an accuracy of 93% on average. While these results are certainly promising, there were also challenges. For example, the model encountered difficulties predicting the number of missed flights where the accuracy dropped to 83%. This leaves room for improvement and uncovers a first gap in the existing academic knowledge: to what extent can the current accuracy of the surrogate models be improved? Possible research opportunities include testing different black-box functions, using different sampling methods or revising the performance metrics. If successful, this will advance the capability of emulating airport terminal processes, which results in remarkably faster algorithms without compromising too much of the accuracy.

Surrogate models are mainly used in two research areas. On the one hand they provide a better understanding of the underlying system, on the other hand they ensure a more efficient process optimization and design [55]. The former is useful as it helps to explain specific model behavior and why certain outcomes are the result of predefined scenario settings in a simulation. The ultimate goal is to fully comprehend the dynamics of a system. This can be realized in various ways. For example, as surrogates are much faster than the original model, one can perform extensive analyses to gain insights into the sensitivity, feasibility, uncertainty, robustness, if-then rules, and so on [11, 14, 23, 55]. Moreover, a thorough comprehension makes it easier to visualize properties and relations between parameters. An example from the literature is the study by Elshawi et al. [25] who reviewed several methods to interpret and visualize machine learning models. Using both global and local techniques, they managed to gain detailed insights into the relationships between several variables to assess people's risk of hypertension. Next to this, another important reason is the creation of less complex models. While it is true that sophisticated algorithms can generally capture trends and patterns that are often too difficult for simpler methods, a thorny issue for scientists remains the poor understanding of how exactly they arrive at their solution. For example, Belle and Papantonis [10] openly question if we can really trust them or could the results perhaps be spurious? It was the main motivation of Kuttichira et al. [58] and Nóbrega and Marinho [72] to create surrogates that are less complex than the original models. They demonstrated the usefulness of comprehensible decision trees and linear regression, respectively. With that, a second gap in the literature is the application of system understanding techniques on the AATOM simulator. That is especially relevant as the research of De Leeuw [20] mainly focused on the proof of concept. Consequently, the system understanding research area remained largely untouched. The author did a preliminary analysis into the variable importances of a random forest regressor. However, given the possibilities of surrogate modeling in other applications, it quickly becomes clear that there are many more techniques that can be applied. Therefore, the main backbone of current thesis will be to attain a better understanding of the complex operations in airport terminals. Chapter 4 is devoted to this research area and reviews the most promising methods from the academic literature.

The second research area is the usage of surrogate models for optimization and design tasks. Evidently, such tasks often involve large parameter spaces, thereby requiring a lot of resources and time which may not always be available. Access to efficient and faster models provides engineers and managers, among other stakeholders, with more powerful tools. That is, more calculations can be done in less time. Some examples of this area in the literature are Song et al. [91] with the structural design for crashworthiness and Wang et al. [99] who optimized the power density of fuel cells. Despite the great potential and practical relevance of this direction, the scope of current research is limited to the system understanding area.

To conclude the introduction, Figure 3.1 depicts the general steps of the surrogate modeling process [14, 28, 78]. The first step is to determine the design of experiments (DOE). One should choose a sampling strategy that extracts as much information as possible from the computationally expensive model with as few data points as possible. Promising strategies are further discussed in section 3.2. Then, when the samples are available, one can train and optimize the meta-models. Although in practice, training and optimizing algorithms often go hand in hand, their theory is explained separately for the sake of clarity. Section 3.3 reviews commonly used abstraction algorithms with their advantages and disadvantages, while the hyperparameter optimization approaches are discussed in section 3.4. An important note is that previous steps are not necessarily sequential, it depends on the DOE — hence the feedback loop in Figure 3.1. For example, active learning strategies exist that iteratively sample, train and optimize surrogates until a certain stopping criterion is reached. Possible criterions are the available computational budget or the accuracy of the emulator. Finally, it is crucial to properly validate the meta-model. The details are explained in section 3.5, but this mainly involves comparing the output of AATOM and the surrogate using appropriate validation metrics, such as for example the coefficient of determination or the root-mean-square error. Logically, it is important to do this with an untouched test sample to avoid fallacious conclusions.

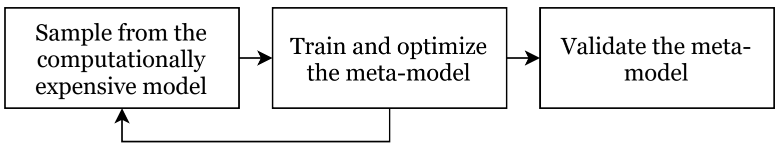

Sample from the computationally expensive model
Train and optimize the meta-model
Validate the meta-model

Figure 3.1: High-level framework of a typical surrogate modeling process [based on information from 14, 28, 78].

3.2. Design of Experiments

The design of experiments (DOE) is the first important step of surrogate modeling. Essentially, the process is concerned with creating a training sample for the meta-models. This is achieved by collecting data across the domain of the expensive model [61]. It makes sense that the sample should contain as much statistical information as possible — a surrogate cannot be expected to have more knowledge than the information it has been trained with. Therefore, this section explores promising sampling methods. The goal is to select a feasible method that extracts the maximum amount of information with the smallest possible sample of data. After all, it remains intensive to run simulations on AATOM.

Quite a bit of research has been done in this area because of its importance. As a result, there are numerous approaches to design the experiments, each with their pros and cons. A general overview of how the methods are usually categorized is shown in Figure 3.2. The theory is retrieved from the recent meta-studies by Liu et al. [61] and Fuhg et al. [32]. In essence, scholars distinguish between one-shot and sequential approaches. The former is the simplest: the sample is collected all at once and hence the name 'one-shot' [61]. It does not take into account any available knowledge. Figure 3.3a visualizes how the method could look like in practice. A number of input values are determined across the domain (indicated by the red dots), which are then fed to the expensive model to calculate the corresponding output. That way, a sample can be obtained. The benefits are that it is relatively easy to implement and the input values can be chosen in such a way as to avoid errors dedicated to randomness [100]. However, the one-shot sampling approach tends to focus on the boundaries of the domain. Depending on the goal, this may not necessarily be a problem, although it can lead to missing regions of interest on the inside [61]. The phenomenon is clearly visible in Figure 3.3a, where the so-called region of interest is the dark area located in the upper left corner.

On the other hand, there are the sequential approaches. As the name implies, data is collected sequentially, as opposed to the one-shot methods. This involves an algorithm that interprets the already gathered knowledge in order to grow the sample in a smart way [61]. There are multiple approaches to achieve this; the literature divides them into two categories. The first one are space-filling methods. They have the property of selecting data points as such to avoid gaps in the domain or in other words, they provide a better spread [32]. The working principle of space-filling DOEs is visualized in Figure 3.3b. What typically happens is that they enrich the samples by iteratively collecting new data using a space-filling criterion. For example, the red dots in the picture indicate an initial sample, to which the purple triangles are added in a sequential manner. Notice that this method fills the domain better, allowing it to capture more of the region of interest. Space-filling approaches are widely adopted for surrogate modeling purposes [11]. Despite being slightly more complicated than one-shot approaches, they are valued for their dispersal ability, leading to the efficient and robust creation of high-quality samples [61]. Nevertheless, there exist even more efficient approaches. One-shot and space-filling DOEs fail to utilize information from the surrogates themselves. While one can have the most dispersed sample, this does not necessarily guarantee the best possible performance of the meta-models. That is why adaptive sampling has received a lot of academic attention in recent years [32]. We describe it as the process where an algorithm iteratively samples in the regions from which a surrogate model benefits the most. Hence, it is also known as 'active learning' because an emulator is trained simultaneously [61]. The notion is that with the same or even a smaller sample size, adaptive approaches should outperform the other two because data points are selected in such a way as to maximize the accuracy of the meta-models. In addition, it might even lead to a reduction in computational requirements for the sampling process [61]. There are

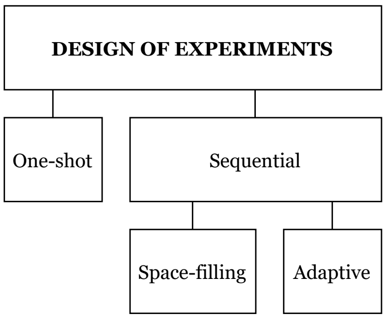
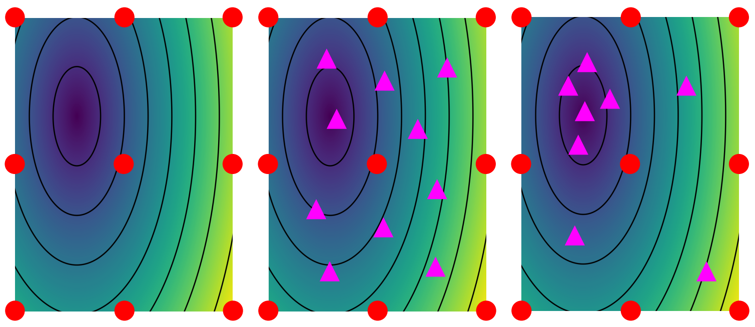


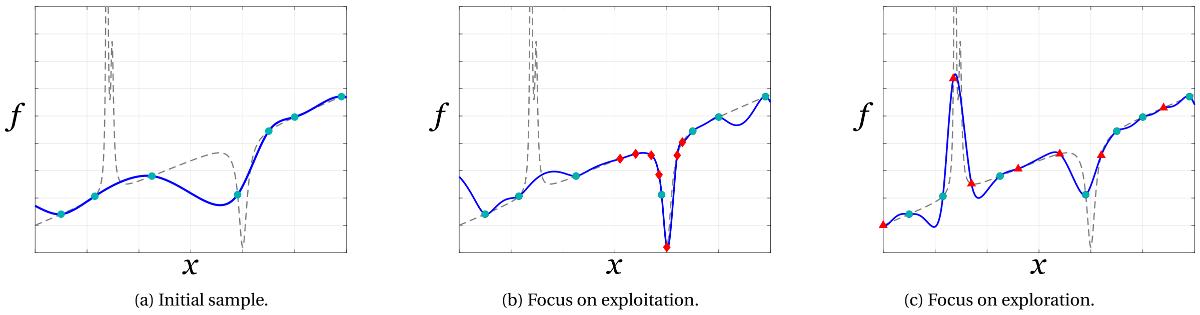
Figure 3.2: Categorization of the most common sampling approaches for the DOE [adapted from 32, 61].



(a) One-shot. (b) Space-filling. (c) Adaptive.

Figure 3.3: Visualization of the three sampling approaches [inspired by 32, 61].

several ways to identify these regions of interest, an example might be to select the point where the model has the greatest uncertainty. Figure 3.3c visualizes how adaptive sampling could look like in practice. As for the space-filling approach, an initial sample is collected first (see the red dots). However, this time the surrogate model is already trained with the initial sample. Then, the region with the largest uncertainty is evaluated by means of an acquisition function [8]. Using this information, the active learning algorithm selects a new data point after which the process is repeated until a stopping criterion is reached (see the purple triangles). Clearly, the adaptive sampling approach focuses more on the darker area — the overall region of interest. The advantage of this method proves itself easily; with similar intensity to generate the sample, higher-quality information can be obtained in comparison with the more traditional alternatives. This is especially useful when the emulated model is very expensive to evaluate [61]. Notwithstanding, applying adaptive sampling is not as easy as it sounds. It is not only a technical challenge, one must carefully weigh the exploration of the entire domain against the exploitation of interesting regions. This is a dichotomy which is often encountered in optimization exercises — one should avoid getting stuck in local optima [11]. A simplified representation of the two concepts is given in Figure 3.4 [32]. Clearly, an exploitation-focused approach (Figure 3.4b) gives more attention to local extrema, while an exploration-focused approach (Figure 3.4c) covers more of the domain. The art is to find an optimal balance between the two.



(a) Initial sample. (b) Focus on exploitation. (c) Focus on exploration.

Figure 3.4: Difference between exploration and exploitation [32]. The dashed function is the target, the blue dots make up the initial sample, the red dots are the adaptively sampled data points, and the blue function is the fitted surrogate using the entire sample.

Back to adaptive sampling; its general principle can be expressed according to Equation 3.1, as given in the study of Liu et al. [61]. AATOM is represented by the target function f. The purpose is then to sample a new input vector $\boldsymbol{x}$ so that the acquisition function AF is maximized. As can be seen, both the exploration and exploitation interests are taken into account. The process recurs until the stopping criterion is reached.

$$\text{for } f \text{ in } D \in \mathbb{R}^n, \quad \boldsymbol{x}_{\text{next}} = \underset{\boldsymbol{x}\in D}{\arg\max}\, \text{AF}\left(\text{exploration}(\boldsymbol{x}), \text{exploitation}(\boldsymbol{x})\right) \tag{3.1}$$

Despite the adaptive sampling strategy is more complicated than the other approaches, it comes with great advantages. It will help to extract the most relevant information from AATOM with a limited computational budget. This is particularly useful since the simulator is rather intensive. Moreover, it was also one of the urging recommendations of De Leeuw [20] following the proof of concept. We therefore opt for the active learning approach in the current research. It is further elaborated in subsection 3.2.1 and subsection 3.2.2, where the former discusses the methods to select an initial sample and the latter the details about the adaptive sampling algorithm.

3.2.1. Initial Sampling

Since adaptive sampling requires the uncertainty of a surrogate, one must first train the model to evaluate its output. This in turn can only be done if an initial sample is available. In the literature, scholars seem to use four sampling methods more than others: Latin hypercube sampling (LHS), orthogonal arrays, Hammersley sequences and uniform designs [100]. They are now briefly explained, while their advantages and disadvantages are summarized in Table 3.1.

Latin hypercube sampling comes first. Introduced a while ago by McKay et al. [67], it is by far the most popular approach in the design of experiments, even today [61, 97]. Assume that one aspires to sample from a two-dimensional input space. The two parameters x_1 and x_2 range both from 1 to 3, resulting in a total of 9 possibilities if only the integer values are taken into account. The LHS algorithm would then choose the options so that only one sample is taken from each row and column [97]. This is what they call a Latin square,

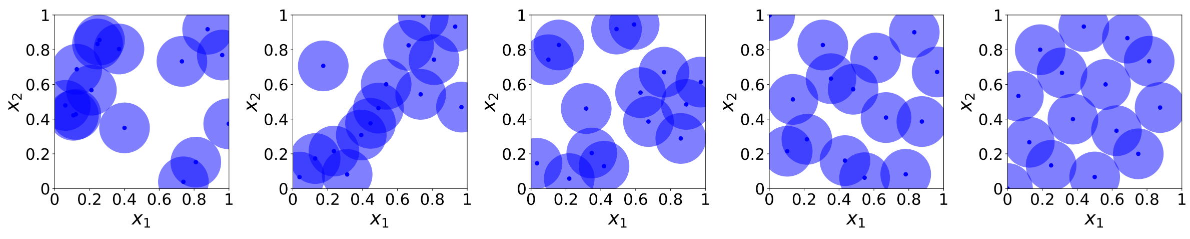
Table 3.1: Advantages and disadvantages of the considered initial sampling methods.

Method	**Advantages**	**Disadvantages**	**Ref.**
Latin hypercube sampling	Flexible, space-filling, stratified, non-collapsing	Correlation in the samples, suffers from dimensionality	[88, 97]
Orthogonal arrays	Similar to LHS, reduces the correlation in the samples, covers the boundaries well	Suffers from dimensionality	[88, 100]
Hammersley sequences	Low discrepancy, maintains uniformity in higher dimensions, space-filling, computationally efficient	Poor coverage of the boundaries, quasirandom	[88, 105]
Uniform designs	Similar to LHS, uniformity, low discrepancy	Poor coverage of the boundaries, suffers from dimensionality	[88, 100]

which becomes a Latin hypercube in a multi-dimensional space. A remarkable drawback from Table 3.1 is that LHS may suffer from correlation in the samples. Using orthogonal arrays instead resolves this problem: it is very similar to LHS, although the data points are chosen so that the correlation is minimal without compromising the Latin hypercube principle [97]. While this is a first improvement compared to the original LHS, another issue remains that the discrepancy is not accounted for. These are the 'gaps' in the domain which may arise when an algorithm samples too much around the same locations. Specific attention is paid to this by the uniform designs. Again, the method is similar to the original LHS, but now the algorithm is forced to select data points in the middle of the subspaces. In comparison, the regular LHS does this randomly [97]. As a result, more uniformity should lead to a lower discrepancy in the input space.

The last sampling method is entirely different from the other three, and neither is it an extension to LHS. Hammersley sequences sample according to the Hammersley set, which is a mathematical formulation to generate sequences with a low discrepancy while maintaining a uniform distribution [105]. As a result of using a sequence, they are technically not entirely random, even if they appear to be. This is what mathematicians call 'quasirandom'. It leads to samples that are well uniformly distributed, even in higher dimensions, without requiring much computational effort [105]. The full details behind the mathematics have been omitted to limit the scope of the literature study, but a simple example to get the idea is as follows. Suppose that one wants to sample between 0 and 1, an arbitrary mathematical sequence could be as 0.5, 0.25, 0.75, 0.125, 0.625, 0.375, 0.875, etc. Note that the discrepancies in the one-dimensional domain are gradually reduced without compromising the uniformity. The actual Hammersley sequence in a multi-dimensional design space is of course more complex than this example, but the general principle remains the same.

To see what the differences between the considered methods look like in practice, a simple experiment is set up to make an initial sample of 15 data points along two-dimensions x_1 and x_2. They both range from 0 to 1. The results are shown in Figure 3.5. The first Figure 3.5a is a benchmark, because in this sample the data has been randomly selected. The LHS in Figure 3.5b provides a better space-filling, although it indeed suffers from correlation — the plot shows a positive relation between x_1 and x_2. Sampling with orthogonal



(a) Random sampling. (b) Latin hypercube. (c) Orthogonal arrays. (d) Uniform designs. (e) Hammersley sequences.

Figure 3.5: Comparison between different space-filling methods to generate an initial sample. The results are created on the basis of a tutorial by Head et al. [39], which is in the documentation of the scikit-optimize package in Python.

arrays solve this problem; in Figure 3.5c there is no clear evidence anymore of a correlation between the two parameters. However, one can still find a presence of gaps as the data points are not well distributed across the input space. This can be solved by using an approach which entails the uniformity property, such as uniform designs or Hammersley sequences. Their samples are shown in Figure 3.5d and Figure 3.5e respectively. The uniform design method is definitely an improvement compared to the others, however the Hammersley sequence yields the lowest discrepancy and best uniformity of the considered methods. In addition, it does so at a low computational cost. As the initial sample is used to train the surrogate model for the first time, it is important to have a well-distributed collection of data points. That is because no information on interesting regions is available. Hence, the meta-models should first explore as much of the domain as possible. Only thereafter, when the adaptive sampling algorithm takes over, the exploitation aspect can be considered as well. Therefore, we choose the Hammersley sequence approach to generate the initial sample; it serves the exploration interests particularly well. The fact that it is only quasirandom is not an issue for this purpose, nor is the poor coverage at the boundaries. Moreover, to validate this choice, Simpson et al. [88] performed two experiments to compare, among other things, the four sampling methods. The study found that uniform designs and Hammersley sequences consistently led to meta-models with the lowest root-mean-square error. Ergo, the selected method will lay a good foundation for the adaptive sampling algorithm.

3.2.2. Adaptive Sampling

When a meta-model has been trained on the initial sample, the adaptive sampling algorithm continues the work. The first step is to assess the surrogate's uncertainty over the domain. This information is required for an acquisition function, which indicates the region of interest to sample a next data point. The new data is then added to the initial sample and the surrogate model is retrained. This process is repeated until it reaches a stopping criterion. The three steps are now discussed further, because there are again numerous approaches to realize the adaptive strategy. Respectively, these are first the methods to obtain the uncertainty of meta-models, then the possible acquisitions functions and lastly the stopping criteria.

Obtaining the Uncertainty of Surrogate Models

The uncertainty of surrogate models is usually described by the errors of their predictions. While this may seem trivial at first, calculating the prediction errors across the entire input space is not an easy task. In most cases, an intermediate step is required to evaluate a proxy for the prediction error. According to the meta-study by Liu et al. [61], four types of methods can be distinguished to obtain (in)directly the prediction error of surrogate models. Either they use the variance, a query-by-committee (QBC), error metrics under a cross-validation (CV) strategy or the derivatives. These are now explained in respective order.

A straightforward way to assess the uncertainty of a surrogate model is to evaluate its prediction variance, which is more commonly known as the MSE — the mean squared error. The notion is that regions with the highest variance are likely to match the regions with the greatest prediction error [61]. It is an effective and widely used approach, often associated with Gaussian processes (GP). That makes sense, since GPs naturally yield the prediction variance [8]. Based on prior knowledge, a posterior distribution is fitted according to Equation 3.2 [from 61], where f is the target, $\hat{f}$ the surrogate, $\hat{\sigma}^2$ the variance and $\boldsymbol{x}$ the input vector.

$$f(\boldsymbol{x}) \sim \mathrm{GP}\left(\hat{f}(\boldsymbol{x}), \hat{\sigma}^2(\boldsymbol{x})\right) \tag{3.2}$$

This is the basic principle of Bayesian inference: additional information is used to improve the prior belief of the GP model [8]. More details about GP regression are discussed in section 3.3. The advantages of a variance-based approach are that it is the most natural, powerful and efficient way to infer about the prediction error over the entire domain [8, 61]. In addition, the surrogate is trained in close cooperation with the adaptive sampling algorithm. Notwithstanding, this can also be considered a weakness, as only a few meta-models have the ability of providing the prediction variance. Consequently, one is limited in advance to those methods that can [61]. Further disadvantages of using specifically Gaussian processes are that only real values can be assumed, the cubic increase in computational effort with the sample size, and scaling issues with high dimensional input spaces [8]. These drawbacks can be solved by using other approaches such as random forests or neural networks instead of GP regression, as they provide the variance as well [8]. They are also explained in section 3.3.

The second strategy to locate regions of interest is to establish a committee of several surrogate models. Scholars describe the ensemble method as a query-by-committee or QBC in short [32, 61]. The key idea is expressed by Equation 3.3 [from 61] and is actually rather simple. A t number of diverse meta-models are fitted on the initial sample. Next, all members i of the committee $\mathscr{C}$ calculate their output $\hat{f}_i(\boldsymbol{x})$ at a potential location $\boldsymbol{x}$. The most interesting region is then determined at the location in the domain where the members have the least agreement [61]. One would expect that this is also where the prediction errors are the greatest. QBC thus assumes that the committee disagreement is a proxy for the prediction error of the actual emulator. Furthermore, the committee members must be sufficiently diverse, because comparable meta-models simply would not lead to disagreements [32].

$$\mathscr{C} = \{\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_t\} \tag{3.3}$$

The benefit of QBC is that it is more generic and does not rely on one specific surrogate model. This gives a lot of flexibility. In addition, the committee's diversity makes the approach generally less susceptible to overfitting on the training sample [61]. However, QBC also comes with some disadvantages. An ensemble of meta-models inherently leads to more models that need to be trained and optimized, with all the associated consequences. Moreover, the members should be carefully selected so that they are sufficiently diverse. The committee needs disagreement to be meaningful [32]. Finally, the independence from a surrogate also has a drawback. Namely, the prediction error of the model is not directly reduced by sampling at the most uncertain regions in the domain, but only via an intermediate committee. A risk is therefore the existence of a discrepancy between the committee disagreement and the actual prediction error.

The third method according to the review by Liu et al. [61] is to evaluate the prediction error under a cross-validation strategy. The details about CV are explained in section 3.4, but the main principle is that a surrogate is trained on all data points of the sample with size m except one. The prediction error e is then measured at that single point i according to Equation 3.4 [from 32, 61]. This is repeated for all data points in the sample. The location with the largest error is then assumed to be the most uncertain region.

$$\forall i \in [1, m]: \; e(\boldsymbol{x}_i) = \left| f(\boldsymbol{x}_i) - \hat{f}_{-i}(\boldsymbol{x}_i) \right| \tag{3.4}$$

The CV strategy is unique as it directly measures the prediction error. Notice, however, that the calculation uses the sample minus one point $\boldsymbol{x}_i$. There is thus still a discrepancy with the true error of the surrogate model that is trained on all available data [32]. Moreover, it is also important to understand that the CV error leads to a data point that has already been sampled. While this indicates the general region of interest, an inevitable follow-up question is where exactly the next point should be sampled. So the neighborhood is known, but an exact location cannot be deduced directly [61]. Furthermore, this strategy is also independent of the model. It can essentially be implemented with any surrogate, allowing for a lot of flexibility. Nevertheless, the main disadvantage remains that the CV algorithm only evaluates already observed data. Not only does this limit the knowledge to certain parts of the domain, it might also result in oversampling. That is the phenomenon when data is collected which is actually redundant [61]. Finally, assessing the prediction error for all data points in the sample can be computationally expensive [32]. This requires the surrogate model to be retrained for every single point: the prediction error cannot be evaluated at a location that was included for training the model. That would give a spurious impression of the surrogate's performance.

The final strategy is to make use of a surrogate model's derivatives. More specifically, the gradient at a point $\boldsymbol{x}$ can be obtained through Equation 3.5 [from 61, 92]. Again, $\hat{f}$ denotes the surrogate model and $\boldsymbol{x}$ the n-dimensional input vector at the point in the domain where the gradient is calculated. This method is based on the notion that meta-models have the least accuracy in regions with large gradients [61].

$$\text{for } \hat{f}: \mathbb{R}^n \to \mathbb{R}, \;\; \nabla \hat{f}(\boldsymbol{x}) = \left[\frac{\partial}{\partial x_1} \hat{f}(\boldsymbol{x}), \frac{\partial}{\partial x_2} \hat{f}(\boldsymbol{x}), \cdots, \frac{\partial}{\partial x_n} \hat{f}(\boldsymbol{x}) \right]^{\mathsf{T}} \tag{3.5}$$

The literature is rather scarce when it comes to using derivatives for adaptive sampling purposes [32]. One reason could be that it suffers from some persistent issues. For example, the approach is strictly limited to surrogate models that are differentiable, which is not always evident [61]. Another challenge is that the meta-model should be sufficiently accurate before any meaningful insights can be obtained [61]. An underfitted model does not contain much information, while an overfitted model can send the algorithm in the wrong direction. Finally, identifying regions with large gradients is one thing, translating them into locations for the next data point to sample is not always straightforward [61].

Now that they all have been discussed, the logical next step would be to choose a particular approach. However, this choice goes beyond comparing pros and cons. Recall that in Figure 3.4 an important distinction was made between exploration and exploitation. Indeed, it is vital that these two concepts are well balanced to create an adaptive sampling algorithm that is both effective and efficient. For example, a method based purely on the variance of a surrogate model would favor exploration over exploitation, and vice versa for a method based solely on the cross-validation prediction error [32]. This is where acquisition functions come into place. These are techniques which compromise the two concepts, as explained further below [8].

Acquisition Function

Meanwhile, it is clear that choosing the right acquisition function can make or break the adaptive sampling algorithm. Although the techniques have been used extensively for optimization tasks [8], research into their application to global surrogate modeling is still emerging. Nevertheless, a rigorous review study has been carried out very recently by Fuhg et al. [32]. Essentially, they made an overview of the state-of-the-art by evaluating 14 methods along various drivers. Their analysis is particularly interesting because the entire spectrum of meta-modeling was considered, rather than just the technical performance. As would be expected from the commonly used "No Free Lunch" theorem [104], no approach is considerably better than others across the board [32]. It therefore remains important to carefully trade-off the characteristics of methods against their task. Nevertheless, Fuhg et al. [32] narrow the scope based on experimental results by proposing 3 promising acquisition functions for surrogate modeling. These are MEPE, EIGF and MIPT, which stand for "maximizing expected prediction error", "expected improvement for global fit" and "Monte Carlo-intersite-proj-th", respectively. They are now briefly explained.

First the MEPE approach, which pursues a cross-validation strategy in combination with the variance of the surrogate model. In essence, the acquisition function selects the new data point in such a way as to maximize a weighted sum of the CV prediction error and the variance [32]. The weights add up to one and balances both terms for exploration and exploitation. To relief the computational burden, the authors suggest to approximate the prediction error. Secondly, the EIGF is the brainchild of Lam [59], who extended the more commonly known "expected improvement". The acquisition function also uses the variance, but now combined with the squared difference between the surrogate model's prediction and the nearest true outcome [32, 61]. That way, it also manages to include both exploration and exploitation interests. The last suggestion was to consider MIPT. That is a more naive approach based on the Monte Carlo principle. After randomly selecting a set of candidates, the acquisition function selects the next data point that has the maximum distance from previously sampled points [32]. Notice that this method does not consider any exploitation interests.

Next, the three considered acquisition functions are compared in Table 3.2 based on various criteria. The results were obtained from Table 3 in Fuhg et al. [32, p. 2728]. A plus sign represents a positive evaluation and vice versa for the minus sign. Outstanding performance — positive or negative — is indicated by two signs. The MIPT seems to outperform the two other. That is because it is a rather simple approach which focuses solely on exploring the domain as much as possible. Hence, not a lot of complexity is involved which is often necessary for the exploitation purpose. Notwithstanding, we have the ambition to consider both aspects in the sampling process. That is because adding an exploitation term should faster lead to a meta-model with a smaller prediction error. Since AATOM is intensive to evaluate, a strategy purely based on exploration can be rather computationally expensive. This limits the choice between MEPE and EIGF. According to Fuhg et al. [32], MEPE would be the preferred option, although they also immediately point out that it is not the easiest approach. MEPE entails quite some complexity. A good compromise would then be EIGF. The authors praise its reliable and solid performance for meta-modeling purposes, without increasing the theoretical complexity or computational requirements too much. For this reason, we choose the EIGF acquisition function.

To connect the dots, the discussions about obtaining a surrogate model's uncertainty and acquisition functions can be summarized as follows. By selecting the EIGF approach, there is opted for a variance-based strategy that is enriched with a geometric measure for the prediction error [32]. This makes it possible to

Table 3.2: Qualitative comparison of the considered acquisition functions [based on Table 3 in 32, p. 2728].

Criterion	**MEPE**	**EIGF**	**MIPT**
Suitability for surrogate modeling	++	+	++
Capturing regular responses	+	+	+
Capturing irregular responses	++	+	++
Performance with small initial samples	++	+	++
Ability to cope with dimensionality	+	+	+
Tendency to keep sampling around the same location	+	+	++
Independency of the surrogate model	-	-	+
Required computational effort	+	++	+
Programming complexity	-	++	++
Complexity to identify the next data point	-	+	+

consider both exploration and exploitation during the adaptive sampling process. Ultimately, this will lead to an efficient but effective surrogate modeling process. When applying the theory in practice, the general Equation 3.1 can now be written as Equation 3.6 [from 32, 61], where $\boldsymbol{x}^*$ denotes the closest input vector that has already been sampled to an arbitrary point $\boldsymbol{x}$ in the domain. We added α as a balancing factor to control the trade-off. The hyperparameter is allowed to change throughout the sampling process, albeit strictly between zero and one. The closer to one, the more preference is given to exploration and vice versa to exploitation.

$$\text{for } f \text{ in } D \in \mathbb{R}^n, \quad \boldsymbol{x}_{\text{next}} = \underset{\boldsymbol{x}\in D}{\arg\max} \left[\alpha \left(\hat{\sigma}^2(\boldsymbol{x})\right) + (1-\alpha)\left(\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x}^*)\right)^2 \right] \tag{3.6}$$

Stopping Criterion

A side effect of adaptive sampling is that it does not end naturally. However, this can be easily solved by implementing a stopping criterion. That is a predefined condition which determines when to stop the sampling algorithm [32]. Thus, data is continuously collected until the criterion is met, after which the sample is considered sufficiently informative. The literature generally distinguishes between two types of criteria: either based on practical limitations or based on the accuracy of the meta-model [32, 61]. Usually, scholars opt for the former, because that is the main concern in many cases. Practically, this is realized by imposing a time limit or computational constraints [32]. The other option is to use the meta-model's accuracy, which is an interesting approach because it collects so many data points until the surrogate behaves as desired. This is achieved by evaluating a validation metric[1] — possibly under a cross-validation strategy [61]. The sampling algorithm continues until a specific goal is achieved or until the successive difference is so small that the surrogate model no longer benefits from additional data. Ideally, we would choose a criterion based the surrogate model accuracy because of its efficiency. However, this is rather challenging in practice as it gives no control over the computation time. Since the sampling process is expected to be a computationally expensive task, the servers of the Air Transport Operations research group at the Delft University of Technology will be used as a supercomputer. Access is granted during predetermined time slots, so it is important to keep track of the time. We therefore opt for a stopping criterion that is primarily based on a time limit, although performance metrics will be monitored in the process to see whether additional sampling slots are needed.

3.3. Methods Used in Surrogate Modeling

Now that the sampling approach has been determined, the next step is to identify black-box functions that could be promising surrogate models. Specifically, this means finding an appropriate function $\hat{f}$ so that the prediction error e is as small as possible compared to the actual outcome of AATOM, denoted by f. The mathematical description is given in Equation 3.7 [from 53], where $\boldsymbol{x}$ is an n-dimensional input vector.

$$f(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}) + e \tag{3.7}$$

We mentioned earlier that meta-modeling has received a lot of academic attention over the years. Hence, it should come as no surprise that a multitude of algorithms have been tried out for this purpose. In the early days, scholars limited themselves to mostly linear models. These models are however still widely used up to this day, mainly because of their simplicity [100]. Later on, Gaussian processes and radial basis functions made their entrance as they proved to be more accurate, albeit at a higher computational cost [100, 101]. Nowadays, research into the possibilities of machine learning is thriving. Its application to surrogate modeling has not been overlooked with support-vector machines, decision trees and neural networks as the most recent additions to the field [3, 78]. The theory behind these six main model categories is now further discussed in subsection 3.3.1 to subsection 3.3.6 respectively. Finally, we compare them in subsection 3.3.7 and then select the ones that will be used in the current research.

3.3.1. Linear Models

Expressions that linearly combine mathematical terms are referred to as linear models. This is a broad category that comes with many possibilities. The simplest and best known method is linear regression, abbreviated as LR. It is formulated according to Equation 3.8 [from 11], in which the regression coefficients are represented by w_i. In essence, LR expresses the surrogate model as a weighted sum of the input variables.

$$\hat{f}(\boldsymbol{x}) = w_0 + \sum_{i=1}^{n} w_i x_i \tag{3.8}$$

[1] Interesting validation metrics are discussed in section 3.5.

The question is then how one can obtain the coefficients to fit the regression model. This is usually done by minimizing the sum of squared errors between the surrogate $\hat{f}$ and the target f over the available sample data — a method known as ordinary least squares (OLS) [11]. LR is popular because it is straightforward, very interpretable and not computationally demanding [100]. However, this also makes it less suitable for more complicated problems. The accuracy rapidly drops when dealing with relations that are non-linear [78]. Therefore, higher-order polynomials[2] are often preferred because they are better able to capture curvilinear patterns [3]. The most commonly applied polynomials for surrogate modeling are those of the second-order [78]. Generally, they are formulated according to Equation 3.9 [from 107]. Note that the interactions between the variables are also taken into account.

$$\hat{f}(\boldsymbol{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n}\sum_{j=i}^{n} w_{ij} x_i x_j \tag{3.9}$$

It is readily clear that polynomials have advantages over LR. Without adding too much complexity, they can model relations that are non-linear and even non-monotonic. Their performance is especially strong when responses are known in advance to behave like polynomials [78]. Conversely, the same also makes them rather inadequate if this were not the case. They remain bound to the polynomial form. Another issue is that these surrogates struggle with responses that have multiple optima, a phenomenon mathematicians call multimodality [107]. Furthermore, the order can be increased even further, although it should be borne in mind that this results in an explosive increase in terms. Many terms lead to many weights, which in turn require large samples to be sufficiently accurate [78]. On top of that, they can also cause instability [107].

An important drawback of both LR and higher-order polynomials is that their functional form is determined over the entire domain. For example, the above methods would have difficulties to emulate a function that is first linear and then becomes quadratic. Friedman [29] resolved this issue by introducing multivariate adaptive regression splines, henceforth abbreviated as MARS. The approach is regarded as an extension to linear models since they are made up of a weighted sum of so-called basis functions [101]. Their general formulation is given in Equation 3.10 [from 53]. The basis functions and their total number are denoted by B and M respectively.

$$\hat{f}(\boldsymbol{x}) = w_0 + \sum_{i=1}^{M} w_i B_i(\boldsymbol{x}) \tag{3.10}$$

A basis function can take three forms, either it is a constant, or it is a spline, or it is a multiplication of multiple splines [103]. In its simplest form, a spline — in this application usually referred to as a "hinge function" — is defined as $\max(0, x - t)$ or $\max(0, t - x)$ [53]. Hereby, the "knot value" t is a constant which is determined when the model is fitted. MARS is more complicated than the first two approaches, because in addition to the weights, the basis functions are also to be decided. The surrogates are trained in two steps: a forward and a backward pass [103]. In short, the forward pass considers several potential basis functions and calculates their corresponding weights based on the OLS method. The functions that ultimately reduce the error the most are then added to the model. This will likely result in overfitting on the sample, but therefore a backward pass is performed as well [38]. The basis functions that yield the least reduction in error are again omitted. This fosters the surrogate's ability to generalize on inputs it has not been trained with. The advantages of MARS are as follows [38, 107]. First of all, it is of course better at modeling responses with different functional forms across the domain than LR or polynomials. Second, since it selects the basis functions in such a way as to reduce the error the most, a natural way of regularization is carried out by choosing the more important variables. Lastly, they are efficient and can cope with large samples. On the other hand, they are not always as accurate and tend to overfit easily despite the backward pass [38, 107]. They are also not as interpretable as compared to the other linear models.

3.3.2. Gaussian Processes

The second category are surrogates based on a Gaussian process (GP). Rasmussen and Williams [77] define this as a group of random variables that has the property of being multivariate normally distributed for each finite combination of its constituents. This is rather abstract to understand, but it essentially means that a distribution is placed over functions. GPs have thus the benefit that the functional space is used directly to infer about the underlying system [77]. The method originates from what geostaticians introduced as Kriging, which expresses responses as a function of nearby observations and their correlations [28]. Over the years,

[2]While at first, one might argue that higher-order polynomials are no longer linear models, it is important to understand that the surrogate remains a weighted sum of mathematical terms. It is thus not required that the terms themselves be linear.

the approach has become a popular basis for surrogate modeling frameworks because of its favorable properties. Namely, it can predict not only responses at unsampled locations, but also the associated uncertainty [78]. That is the reason why it is a very suitable method in combination with adaptive sampling. Indeed, subsection 3.2.2 discussed that one aspires to sample in the most uncertain regions.

Surrogate models constructed from GPs are described in their general format according to Equation 3.11 [from 77, 87]. The mean function is denoted by $m(\boldsymbol{x})$ and the covariance function by $k(\boldsymbol{x},\boldsymbol{x}')$. The former is the expectation of the surrogate at a certain location $\boldsymbol{x}$ (see Equation 3.12). Here is also where several types of Kriging originate; some assume the prior mean to be a constant, others assume it to be unknown or even of some functional form [107]. Its modern application as a machine learning technique usually sets it to be zero. That way, the computational requirements are less intensive and the covariance function will completely take over the inference [77, 87]. The role of the covariance function, which is oftentimes also referred to as a kernel, is to express a relation between the responses at locations $\boldsymbol{x}$ and $\boldsymbol{x}'$ [87]. The general form is given in Equation 3.13, but note that there are several ways to model the correlation. Some examples are constant, linear, (squared) exponential, spherical, Matérn kernels, and so on [11, 107]. Nevertheless, their premise remains the same: the further away $\boldsymbol{x}$ and $\boldsymbol{x}'$ are from each other, the less likely they are correlated [87]. In other words, if the Euclidian distance is small, their responses are presumed to behave akin. One of the most commonly used kernel functions is the squared exponential, as given in Equation 3.14 [from 77, 87]. In the equation, one can clearly see that the value of the kernel function increases with a decreasing distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$, with l a hyperparameter scaling the length.

$$\hat{f}(\boldsymbol{x}) = \mathrm{GP}\left(m(\boldsymbol{x}), k(\boldsymbol{x},\boldsymbol{x}')\right) \tag{3.11}$$

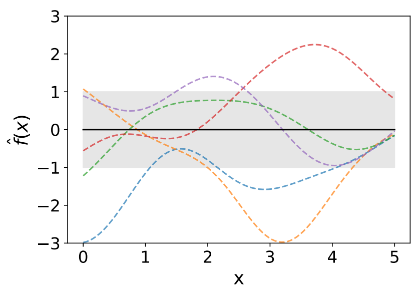
$$m(\boldsymbol{x}) = \mathbb{E}\left[\hat{f}(\boldsymbol{x})\right] \tag{3.12}$$

$$k(\boldsymbol{x},\boldsymbol{x}') = \mathbb{E}\left[\left(\hat{f}(\boldsymbol{x}) - m(\boldsymbol{x})\right)\left(\hat{f}(\boldsymbol{x}') - m(\boldsymbol{x}')\right)\right] \tag{3.13}$$

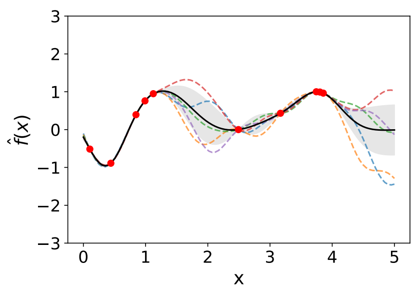
$$k(\boldsymbol{x},\boldsymbol{x}') = e^{\left(-\frac{||\boldsymbol{x}-\boldsymbol{x}'||^2}{2l^2}\right)} \tag{3.14}$$

Moving from theory to practice, a simple one-dimensional example of GP regression is illustrated in Figure 3.6 [based on 75]. In Figure 3.6a, functions are drawn from the GP before it has been fit on known observations. One can clearly observe that the mean value is equal to zero over the entire domain, which makes sense since the prior mean was assumed to be zero. However, this changes when functions are drawn from the posterior distribution, shown by Figure 3.6b. There is still some uncertainty, but as the functions interpolate the observed points, it is greatly reduced in the neighborhood of the points. In practice, there is no need to draw individual functions from the GP as they altogether lead to a mean value and a variance (or standard deviation) over the domain. Those are the two output values that are eventually used. In summary, the mean value of a GP surrogate is more reliable the closer it is near the response of an input combination that has actually been sampled in AATOM, where reliability is represented by the variance (or standard deviation).

It is now clear that one of the greatest advantages of GP regression is that it offers a measure of uncertainty while making a prediction [78]. It is a popular active learning approach for this reason alone, with a proven track-record in the field. On top of that, GPs are also praised for the high accuracy they can generally achieve [101]. Lastly, their hyperparameters can be determined in an objective manner with a maximum likelihood calculation [78, 107]. However, despite being an attractive method for surrogate modeling, they also have their drawbacks. GPs tend to become computationally intensive with larger sample sizes. The reason is



(a) Sampling from the prior distribution.



(b) Sampling from the posterior distribution.

Figure 3.6: A simple one-dimensional example of GP regression. The programming code of the example is obtained from a tutorial by Metzen and Lemaitre in the documentation of the scikit-learn package [75]. The solid dark line is the mean of the GP, the gray area represents the standard deviation, the dashed lines are functions drawn from the GP and the red dots represent the known observations.

because the posterior is obtained by inverting the covariance matrix, which becomes more challenging with an increasing number of observations [11, 107]. Another point of attention should be that the sampling algorithm does not take data points that are too close to one another. Otherwise, there is a risk that the covariance matrix becomes ill-conditioned, resulting in severe response instability with respect to small changes in the input vector [78]. Finally, GPs involve some mathematical complexity. This makes them black-boxes and quite difficult for post hoc interpretation [101].

3.3.3. Radial Basis Functions

Another popular meta-modeling method is the usage of radial basis functions (RBFs). These functions model responses based on distances from known observations [11]. This seems somewhat similar to the principle of GPs because they also make use of Euclidian distances. However, the main difference is that RBFs do not build the surrogate as a stochastic process, but rather as a weighted sum of basis functions. With m the size of the sample and B a basis function, the general notation of RBFs is shown in Equation 3.15 [from 3, 11]. Similar to the linear models, w denotes the allocated weights to the basis functions.

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{m} w_i B_i \left(||\boldsymbol{x} - \boldsymbol{x}_i||\right) \tag{3.15}$$

There are again multiple ways to define basis functions. They can be linear, cubic, thin plate, Gaussian, multi-quadratic, and so on [11, 28]. For example, a cubic basic function is given in Equation 3.16 [from 11].

$$B\left(||\boldsymbol{x} - \boldsymbol{x}_i||\right) = ||\boldsymbol{x} - \boldsymbol{x}_i||^3 \tag{3.16}$$

The choice of the basis function depends on the data. There is no single format that generally stands out over the others, although Bhosekar and Ierapetritou [11] mentioned the success of the cubic and Forrester et al. [28] mentioned the popularity of the Gaussian (see Equation 3.14). It is usually determined by trying out different functions. The associated weights are calculated through the ordinary least squares method [78].

The advantages and disadvantages of RBFs are to some extent comparable with GPs. Their construction is not as complex, yet they still manage to capture non-linear responses [28]. Notwithstanding, they do not give insights into the uncertainty of their predictions, which makes them even more difficult to interpret [101]. Furthermore, it is not always as straightforward to determine the basis functions and their possible hyperparameters. This often involves some iterations on a trial and error basis [78]. Finally, one must again be careful with clustering in the sample. Ill-conditioning can induce undesired instability in the response of the surrogate model [28].

3.3.4. Support-vector Machines

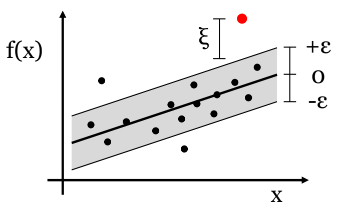
Along with the emergence of machine learning, support-vector machines (SVM) are one of the more recent additions to surrogate modeling architectures [100]. They are more common in classification tasks, although they also find their way into regression, better known as support-vector regression (SVR) [107]. Quite similar to RBFs, the model is made up of a weighted sum of basis functions in addition to a constant μ. The general format is given in Equation 3.17 [from 11, 28].

$$\hat{f}(\boldsymbol{x}) = \mu + \sum_{i=1}^{m} w_i B_i(\boldsymbol{x}, \boldsymbol{x}_i) \tag{3.17}$$

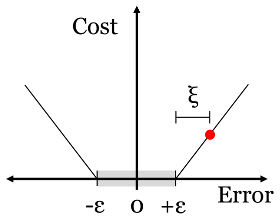
Now, one might be wondering how SVR and RBFs differ. That is in the calculation of the base term and the weights. Instead of the OLS method, SVR makes use of an optimization problem. A simple (linear) example is formulated according Equation 3.18 [from 11, 28], where ξ is a slack variable, ε determines the unpenalized error range, C is a tolerance hyperparameter and f is the output from AATOM.

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2}||\boldsymbol{w}||^2 + \frac{C}{n}\sum_{i=1}^{m}\left(\xi_i^- + \xi_i^+\right) \\ \text{Subject to} \quad & \boldsymbol{w}\cdot\boldsymbol{x}_i + \mu - f_i \leq \varepsilon + \xi_i^- \\ & f_i - \boldsymbol{w}\cdot\boldsymbol{x}_i - \mu \leq \varepsilon + \xi_i^+ \\ & \xi_i^-, \xi_i^+ \geq 0 \end{aligned} \tag{3.18}$$

This requires some additional explanation, which will be done in conjunction with Figure 3.7 [89]. In essence, SVR tries to find a fit to the data that explains as much of its variation as possible, while restraining the complexity of the surrogate [28]. The variation is said to be perfectly explained if all data points of the sample



(a) The tube within errors remain unpenalized.



(b) The effect of outliers on the cost function.

Figure 3.7: Main principle of support-vector regression [89]. The red dot indicates an outlier at a distance ξ from the tube.

would be within an accepted range $\pm\varepsilon$ from the surrogate. This range is called a "tube", and is represented by the first two constraints in Equation 3.18 [28]. However, one should be careful not to force this, otherwise the model would easily overfit. That results in a surrogate that generalizes poorly to unexplored regions, which is of course undesired. The optimization problem does this by choosing the weights so that the norm of the weight vector is as small as possible (see the first term in the cost function of the optimization problem) [28]. Graphically, the tube is shown in Figure 3.7a; minimizing $\boldsymbol{w}$ tries to flatten it as much as possible. So far the theory makes sense, but a question raises how outliers are handled. Indeed, Figure 3.7a shows a fitted surrogate model, even though there are three data points outside the accepted $\pm\varepsilon$-range. That is allowed because of the slack variables ξ in Equation 3.18. Without them, the optimization problem would not be feasible [89]. Notice that the slack variables are also included in the cost function. Their effect is depicted in Figure 3.7b. It illustrates that only outside the tube, the associated penalty increases linearly with the error [28]. The rate at which this happens is managed by the tolerance hyperparameter C. Hence, tuning the parameter balances the complexity of the surrogate against the extent to which outliers are accepted [89]. In summary, support-vector regression aims to fit a function to the sample so that the data points are within a tube of predefined width, without making the function too complex. To some degree, outliers can be outside this range, although this is kept at a minimum because it entails a penalty in an optimization problem.

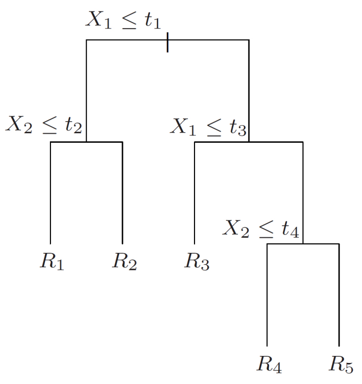
Up to now, it was assumed that the basis functions were linear. However in practice, the responses of AATOM are likely to be more complex. SVR can cope with that by adopting different basis functions to map the input into a so-called feature space [107]. The principle is the same as with GPs and RBFs. Common kernels are linear, polynomial, Gaussian, sigmoid, and so on [107]. The optimal choice again depends on the data and is usually determined by trying and comparing the different functions with one another.

It is quite remarkable that scholars are not entirely harmonious about the accuracy of SVR. Some argue that they outperform fellow surrogate modeling techniques [e.g., 100], others mention a comparable performance [e.g., 11], while Williams and Cremaschi [103] even said that they are generally not as accurate. A reason for the disagreement might be the need for a thorough hyperparameter optimization. This includes not only choosing the right kernel function, but also the width of the tube (determined by ε) and the tolerance for outliers (determined by C). The algorithm is thus rather flexible, which can be considered both an advantage and a disadvantage. Nevertheless, properly determining the optimal parameters is not always a straightforward task and requires quite some attention from the user [78, 107]. Another disadvantage of SVR is that the weights are determined through an optimization problem. Especially with an increasing dimensionality, this process could become computationally intensive [11, 103]. The method is also rather difficult to interpret [38]. On the contrary, Yondo et al. [107] praise its robustness, low risk of overfitting and ability to perform well with small samples. Razavi et al. [78] argue that SVR handles well noisy data because of the error insensitivity in the tube, and mentions the advantage of incorporated regularization. By that, they mean the tolerance C in the acceptance of outliers.

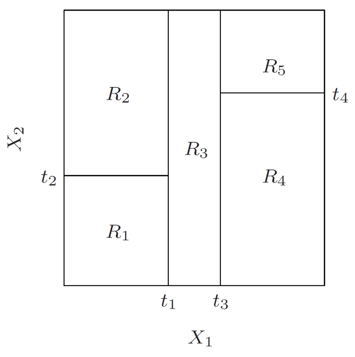
3.3.5. Decision Trees

Apart from support-vector machines, machine learning models on the basis of decision trees are also gaining momentum. The reason why is not far-fetched; they are considered to be one of the most powerful algorithms that are currently available [34]. Usually they are applied in an ensemble consisting of multiple trees, but to understand the general working principle, a single tree is explained first. The explanation is based on the theory from Hastie et al. [38].

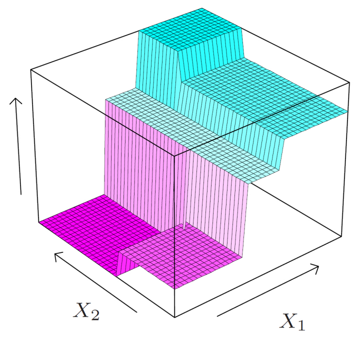
Imagine that AATOM has one response Y and two inputs X_1 and X_2. A sample is available to train the algorithm. The regression tree — a decision tree for regression tasks — is then constructed as follows. Without any additional knowledge, the best estimate for the response would simply be the average of the sample.



(a) A two-dimensional regression tree.



(b) The partitioned input space.



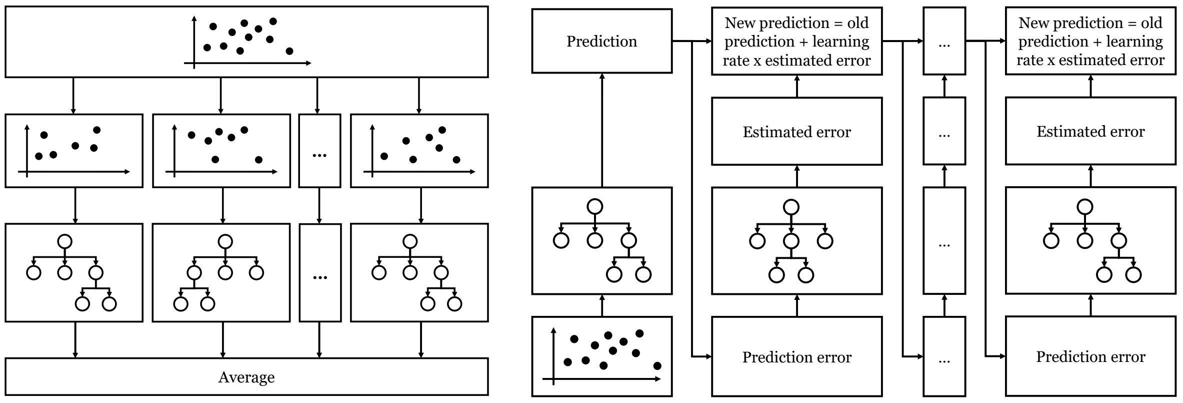
(c) The tree's response.

Figure 3.8: The principle behind regression trees [38].

This is of course rather naive and probably not very accurate, especially since we know that the response changes over the domain. The regression tree solves this by splitting the sample in two groups, based on a certain condition. All data points that have an X_1 smaller than or equal to a value t_1 are part of one group and the others make up the second group. The estimated response of data points with an X_1 smaller than t_1 is now the average of the first group, while the same holds for the other group. There are now thus two possible responses. The question is then which value of t_1 results in the best split. Cleverly, this is determined based on squared errors: t_1 is chosen as such that the sum of squared errors between the predictions of the regression tree and the actual outcome of AATOM is minimized. In the current example, both X_1 and X_2 are evaluated and eventually the best option is selected. However, for a real case with n dimensions, a hyperparameter can limit the considered options by randomly selecting a subset of the features at every split. While the two distinct responses are already a great improvement compared to the naive average of the entire sample from the beginning, the regression tree can do even better. Indeed, the two groups can be split up further. That happens until a leaf reaches a predefined number of observations or until the regression tree reaches a predefined depth. These are again two hyperparameters of the algorithm. It is important to note that trees are built in a recursive manner. That is, the splits are determined after one another. In the example, the value of t_1 is set before there is any knowledge about later splits t_2 to t_4. The algorithm is thus said to be greedy. The fact that t_1 is the best option at that time does not guarantee that this was the overall best choice when taking into account later splits. Yet, such approach is required to limit the computational burden of the algorithm [38]. Altogether, the end result of the regression tree is illustrated in Figure 3.8 [38]. At the left, Figure 3.8a shows the actual tree with four splits. This leads to five possible responses R_1 to R_5. How exactly the input space has been partitioned is shown by Figure 3.8b. We can see that there are five groups, each leading to one of the responses as depicted in Figure 3.8c at the right.

The main benefit of decision trees is undisputed: they are very interpretable [3, 34, 38]. This was readily clear from the example. They are not difficult to understand and the partitioned input space leads to accessible rules that can be used to explain the behavior of the algorithm. On top of that, they are powerful and can handle almost any type of data [34]. It does not matter whether an input variable is e.g. continuous or discrete. However, decision trees are prone to some disadvantages. First of all, a problem is that they are not very robust. Minor changes in the training sample could result in entirely different trees [34]. They can thus be rather unstable, increasing the variance [38]. Furthermore, they are not smooth. The response will be the same as long as one stays within a partition, but once the boundary is crossed into another, the response suddenly changes. This phenomenon is clearly visible in Figure 3.8c [38]. Finally, decision trees tend to quickly overfit on the training sample. Consequently, they struggle to properly generalize, making them rather inaccurate for regions that have not been explored [3, 34].

Despite the drawbacks mentioned above, decision trees are actually used a lot in machine learning tasks. The key is that they are rarely used alone, but rather in an ensemble of multiple trees. Analogously, scholars refer to these structures as "forests". The notion is that an ensemble of many trees can make better predictions than one tree on its own [34, 38]. We will consider two approaches to build the forest. On the one hand, random forests are created through bootstrap aggregation, better known abbreviated as bagging. One the other hand, trees can be concatenated, which is known as boosting. The fundamental difference between the two approaches is the way they aim to improve their performance. The former reduces the variance by


...
...
Average
Prediction
New prediction = old prediction + learning rate x estimated error
Estimated error
Prediction error
...
New prediction = old prediction + learning rate x estimated error
Estimated error
Prediction error

(a) Random forests.

(b) Gradient boosting.

Figure 3.9: The principle behind ensembles of regression trees [based on materials from 34].

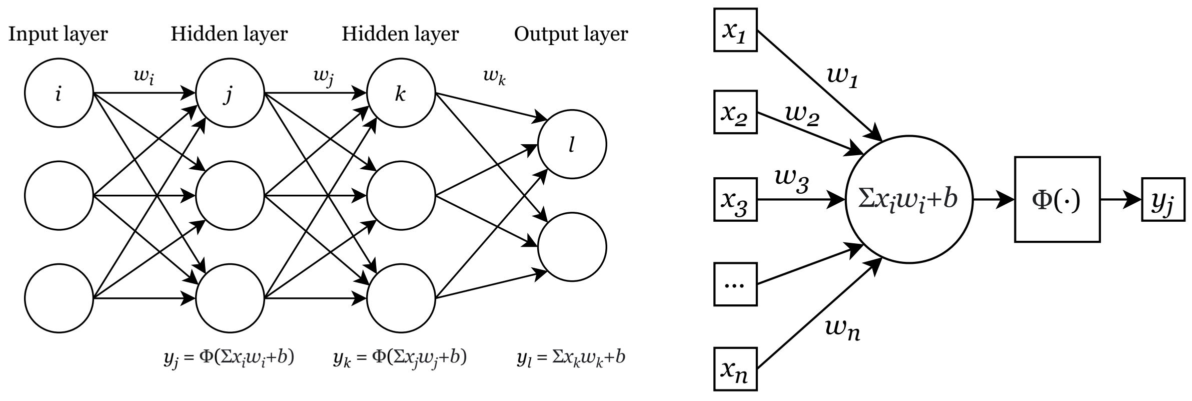
averaging many independent trees in accordance with the law of large numbers [34]. Conversely, the latter focuses on reducing the bias, as it grows trees specifically for that purpose [38]. Thus, they both reduce the prediction errors of single decision trees, albeit in a different way. Hastie et al. [38] describe this with the bias-variance decomposition of the error. The principles behind random forests (RF) and gradient boosting regression (GB) are now explained in respective order, along with their advantages and disadvantages.

The architecture of random forests is depicted in Figure 3.9a [based on 34]. From top to bottom, the first step is to create a subsample. This is done by randomly drawing data points from the overall sample, where the same point can be selected more than once — a bootstrap sample according to the terminology [38]. Then, as explained before, a single regression tree is trained on the subsample. This process is repeated until the forest reaches a predefined size, which is an additional hyperparameter of the model. Finally, the ultimate response $\hat{f}(\boldsymbol{x})$ of the random forest is determined by taking the average of all individual trees [34, 38]. Random forests are one of the most popularly used algorithms in machine learning. That is because they are extremely powerful, do not require a lot of preprocessing and are generally very accurate on a wide variety of data sets [34, 38]. Moreover, even with the slightest tuning, they already yield a solid performance. They do not require too much effort in optimizing the hyperparameters [38]. The ensemble structure makes random forests inherently less interpretable and more computationally intensive in comparison with single regression trees. However, a large benefit is that they give insights into the relative importance of variables. This measure shows which inputs contribute the most in making accurate predictions and is therefore a very relevant tool for the explainability of the model [34, 38]. Finally, it should be noted that random forests are a complex algorithm. This makes it powerful, but also prone to overfitting on the training sample [38]. If there are signs of the phenomenon, it can be managed through the hyperparameters or increasing the sample size.

Alternatively, the boosting algorithm is illustrated in Figure 3.9b [based on 34]. Readily it can be seen that the philosophy behind the architecture is entirely different from random forests. Trees are not grown in parallel, but sequentially [34]. From left to right, the first step is to create an initial prediction of the response. Usually this is just a constant, such as the average value of the sample [38]. Subsequently, the prediction errors — residuals — can be evaluated and here is the point from which the gradient boosting algorithm is rather unique in its further approach. Namely, it will fit a new regression tree onto the residuals. An updated prediction is then made based on the old one plus the estimated error, scaled by the learning rate. This is repeated until the sequence reaches a predefined length [34, 38]. The learning rate and the total number of trees are thus two important hyperparameters of the algorithm. So in essence, gradient boosting directly reduces the bias of its response. For that reason, it is one of the most accurate machine learning architectures that is currently available. They are often able to outperform random forests, although this comes at the disadvantage of being more complex to implement [38]. Tuning their hyperparameters need more attention to avoid overfitting, but one should also be careful for underfitting [34]. In addition, they entail longer training times. The reason is because the trees cannot be trained in parallel threads. Indeed, the sequential design forces the algorithm to train one tree after the other [34]. Finally, gradient boosting is of course also less interpretable than single regression trees. Although just as with random forests, they give insights into the relative importance of the input variables, which is considered a great advantage [38].

3.3.6. Neural Networks

The last type of models that is considered are feedforward neural networks (NN). Contrary to what one might think, NNs have been around for a while. However, they were only popularized with an increasing availability of computational power and the development of efficient training methods [34, 98]. Over the years, they have been commonly used as surrogates [78]. Their general principle is illustrated in Figure 3.10 [based on 98].



(a) Construction of a network.

(b) Mathematics of a neuron.

Figure 3.10: Architectural structure of artificial neural networks [adapted from 98].

Biological neurons have been the source of inspiration for the creation of NNs [34]. Essentially they are an interconnection of nodes, constructed according to Figure 3.10a. It starts from the input layer, which feeds data into the network. In general, each neuron corresponds to one dimension of the input vector, meaning that in practice there could be many more than three [34]. This information is then passed to the first hidden layer, which again consists of several neurons. Their input is processed according to Figure 3.10b. First, the inputs x_i are multiplied by a weight w_i, after which a bias term b is added. The outcome is then entered into a so-called activation function Φ. This triggers neurons and adds non-linearity to the network, allowing it to capture highly complex responses [34]. Popular functions for regression tasks are a ReLU (rectified linear unit), a hyperbolic tangent and a sigmoid [98]. Respectively, the former is simply equal to zero or its input if that is positive, the hyperbolic tangent scales its input between minus and positive one, and the latter scales its input between zero and one. The result of the activation function is then passed to the following layer, after which the process is repeated. The network in Figure 3.10a has two hidden layers, but in practice there can be as many as one wishes. Moreover, this also holds for the size of a layer. The example shows that both hidden layers each consist of three neurons; however, this is also determined per design. The architectural layout is thus highly flexible and must be carefully determined according to the data that is being modeled. The neural network ends in the output layer. As the name implies, this layer yields the response of the surrogate model, where each neuron typically corresponds to one output variable [34]. For regression tasks, there is generally no activation function in the neurons of the output layer, but that again is a design choice [34].

While the theory behind the architecture of NNs is actually rather straightforward, the main challenge lies in training them. The weights[3] should be adjusted so that the error between their output and the actual response is minimal [38]. This intensive process has been revolutionized with the introduction of backpropagation by Rumelhart et al. [85]. The essence of the training algorithm is as follows. It starts with randomly initializing the weights. A first batch from the sample data is then fed to the network, resulting in an output. This is called the forward pass [34]. Logically, the first results will be highly inaccurate, but that is normal. The next step is to calculate the prediction error, which is usually done through a loss function based on the squared errors [38]. So far this is actually quite similar to previous models. However, the main difference now is how NNs learn from their mistakes. They do so by attributing the error back to the neurons in proportion to their contribution. Simply put, the error is propagated back by applying the chain rule from the output layer, through the hidden layers until it arrives again at the input layer — the backward pass [34]. This information enables the training algorithm to update the weights of the neural network. It does so by following a learning rate weighted gradient descent on the associated error functions of the interconnections [34]. The forward and backward passes are continued multiple times with different batches until the NN converges to

[3]Bias terms are henceforth considered part of the weights; think of it as an extra neuron for each layer (except the output) which is always equal to one. The biases are then equivalent to the weights of their interconnections. The literature describes it as a bias neuron [34, 38].

a solution. Note that it is possible for batches to be used more than once. The terminology refers to a number of epochs for the number of times the whole training sample has been passed through the network [38].

Artificial neural networks join the group of tree ensembles as the most powerful algorithms available in machine learning. The deep structure with non-linearities allows them to capture complex relations, which may consist of numerous dimensions [34, 38, 98, 107]. Moreover, they naturally consider interactions between the input parameters [101]. NNs can be very accurate and are proven methods in their application as surrogate models [107]. The fact that they are extremely flexible — their construction and especially the number of hidden layers is completely customizable — is considered both an advantage and a disadvantage. Indeed, it gives the user a lot of room for optimization, but doing so is not an easy task and might require many manual iterations [38, 98]. On top of that, the training algorithm is computationally rather intensive and requires a lot of data. It may take several hours for the solution to converge [34]. In terms of interpretability, NNs are one of the worst possible alternatives. They are virtually complete black-boxes, making it difficult to understand how they get to their output [38, 101, 107]. Another challenge is that they are prone to overfitting [38, 107]. This must be carefully kept in mind when training, otherwise the model will make inferior predictions for unexplored regions in the domain. Finally, Hastie et al. [38] mention the risk of local minima in the loss function. A consequence may be the convergence of the training algorithm to a solution which is sub-optimal. Nonetheless, Géron [34] states that this phenomenon is actually seldom.

3.3.7. Comparison Between the Methods

Now that the six modeling categories have been discussed, along with the advantages and disadvantages of every candidate, it is time to compare them and select the best alternatives. However, before we can do that, at first it is important to identify the criteria which determine whether a method is considered suitable or not. With the goals of the research in mind, two drivers can be readily set: accuracy and interpretability. On the one hand, the responses of the surrogates need to be sufficiently close to those of AATOM. This is evident, but otherwise they would be rather ineffective for further usage and analysis. On the other hand, improving the system understanding is also an important driver for meta-modeling AATOM. The selected methods should therefore be sufficiently transparent so that their behavior can be thoroughly analyzed. Next, another criterion is meeting the requirements. Recall that in section 3.2 an adaptive sampling strategy was determined based on a surrogate model's variance; not all of them have the ability to provide insights into their uncertainty, so the models' capabilities must be taken into account. A final aspect to include in the trade-off is feasibility. This is broad, but mainly relates to the practical accessibility of a model. They will be evaluated based on the required programming effort, and time to train and optimize the hyperparameters.

Due to its natural integration with the selected sampling strategy, the first method that we choose is Gaussian process regression. In the end, the model is already trained because of the active learning procedure. However, that is not the only reason. Apart from perfectly meeting the requirements, GP regression is also praised for its high accuracy and is fairly feasible to use. Notwithstanding, a problem is the lack of transparency. They do indicate the uncertainty of their predictions, but other than that their behavior is difficult to interpret. This can be solved by using the collected sample to train more interpretable models, such as linear approaches. One of their strengths is the low complexity, while still being able to generalize acceptably well. They are not the most accurate alternatives, but that is not the purpose either. The popularly used second-order polynomial is selected because of its high interpretability and feasibility — they have no hyperparameters, so no optimization is required. Furthermore, we prefer them over linear regression as they still manage to capture relations that are non-linear and non-monotonic. The final category that is interesting to consider are the tree ensembles. Random forests and gradient boosting are among the most powerful machine learning algorithms available. Accuracy is thus the main driver for them to be selected. On top of that, they are much more feasible compared to neural networks. The latter is rather complex and requires a lot of effort to properly optimize. While gradient boosting tends to outperform random forests, we will use them both regardless for the following reason. In the proof of concept study by De Leeuw [20], random forests regression was selected as the main meta-model for AATOM. Hence, to compare results, it will be necessary to also use the method. The disadvantage is that this requires the optimization of both methods, although the selection of a clever tuning algorithm in subsequent section 3.4 is expected to mitigate the issue.

Altogether, we select four methods to meta-model AATOM. GP regression because it perfectly meets the requirements, a second-order polynomial because of its interpretability, and the highest accuracy is likely achieved by random forests and gradient boosting. Moreover, these four methods are all considered feasible. LR, MARS, RBFs, SVMs, single decision trees and NNs are not taken into account because they are either too inaccurate, too complicated or too difficult to interpret.

3.4. Hyperparameter Optimization

Selecting the model architectures is one thing, they also should be properly optimized to get the most out of them. The performance of machine learning algorithms can be severely affected by the choice of hyperparameters [42]. In concrete terms, this means that the parameters should be chosen as such to minimize the prediction error given a certain training sample. Finding the optimal combination can be a difficult and, above all, a computationally demanding task [42]. Hence, it will not be surprising that this topic has received some scholarly attention. The most popular approaches are now briefly described, their advantages and disadvantages are summarized in Table 3.3, and finally we select a suitable method for current research.

The most widely adopted approach is a grid search [42]. The principle behind this method is rather straightforward: the user defines a number of parameter combinations, after which the search algorithm tests them all exhaustively [106]. The combination corresponding to the best performance is then selected for the machine learning model. Instead of manually defining combinations, an alternative approach is to express the parameter spaces by means of statistical distributions [5]. This is the random search; the algorithm draws random samples from the distributions and tests their performance. This process is repeated until a computational budget is reached, or when the machine learning model is sufficiently accurate [109]. Thereafter, the best combination is chosen. The simplicity and generally good performance contributed to the popularity of both methods. But more than ever, scholars are emphasizing their naivety [5, 42]. They do not learn throughout the search process and keep drawing candidates from the entire parameter space. This makes them rather inefficient and has led to adoption of Bayesian optimization (BO). With less iterations, BO usually succeeds to find a combination that is close to the global optimum [106]. It works as follows. A surrogate model is fitted to the objective (e.g., a prediction error metric), then the parameter combination is selected that leads to the surrogate's optimum, after which this combination is evaluated on the actual machine learning model. Finally, the surrogate is updated with the new information and the process continues sequentially until a predefined number of iterations is attained [5, 106]. Note the analogy with the selected adaptive sampling approach from section 3.2, although the fundamental difference is that the goal now is to find a global optimal hyperparameter setting rather than to fit a model as closely to AATOM's response as possible. Besides BO, another commonly used method that offers the possibility to search for global optima are evolutionary metaheuristics. This is a category of algorithms based on the theory of evolution, guided by

Table 3.3: Advantages and disadvantages of the considered hyperparameter tuning methods.

Method	**Advantages**	**Disadvantages**	**Ref.**
Grid search	Simple, transparent, allows for multi-threading	Suffers from dimensionality, requires experience on promising candidates, only limited ranges are practical	[5, 109]
Random search	Larger and higher dimensional search spaces are possible, can be easily stopped and resumed, allows for multi-threading	Inefficient, remains computationally intensive, finding the optimal combination is not guaranteed	[5, 109]
Bayesian optimization	Efficient, high chances to find a combination close to the global optimum, no need for prior experience regarding promising candidates	Difficult for multi-threading, rather complex	[106, 109]
Evolutionary metaheuristics	Effective without being too complex, good chances to find a combination close to the global optimum	Multi-threading not always possible, experience on promising regions for initialization is desirable for a fast convergence, consists of complicated hyperparameters itself	[106, 109]

the principle of "survival of the fittest" [2]. The high-level idea is to start with a population of parameter combinations that is randomly initialized. The individuals are then evaluated by means of an objective, which essentially determines their "fitness" (e.g., a prediction error metric). The best ones have the highest chances to survive, after which crossover and mutation is performed on them to create a new generation. This process is again continued until a computational budget is reached, or when the machine learning algorithm is sufficiently accurate [2, 106, 109].

Now that the main working principle of promising hyperparameter tuning strategies are discussed, we will compare the advantages and disadvantages in Table 3.3 to select the most appropriate method. The benefits of a grid and a random search are indeed attractive. They are not too complex and have a good chance of finding a solid parameter setting. However, they remain rather inefficient and do not guarantee a global optimum. We therefore have the ambition to choose a more sophisticated method. In particular, our preference goes to Bayesian optimization over evolutionary metaheuristics because it directly searches for the best parameter combination, without requiring too much of optimization itself. It is an efficient algorithm with a high chance of finding a solution close to the global optimum. Moreover, no prior information is necessary about promising parameters — the algorithm takes already care of that. Furthermore, the complexity of BO is not deemed an issue because of its similarity with the adaptive sampling strategy from section 3.2. A lot of knowledge has already been gathered about the method. From the practical perspective, Bayesian optimization is really well integrated with the scikit-learn package in Python using scikit-optimize [39, 75]. This greatly reduces the programming effort required during the implementation. The only challenge could be the difficulty for multi-threading, although the computational demands are not expected to exceed the available budget. Hence, Bayesian optimization is a solid choice to tune the hyperparameters of the machine learning models for this research.

Finally, it has been mentioned that the tuning strategies evaluate the performance of machine learning models during the optimization. While this may sound trivial at first, an important consideration is which data set to use. On the one hand, validation on the training set produces spurious results because performance metrics have to be calculated on unseen data. On the other hand, the test set is required for the overall validation (see section 3.5) and is therefore not available for the hyperparameter optimization. The ideal solution would be to collect a separate data set specifically for this purpose [38]. However, recall that sampling from the AATOM simulator is computationally intensive. A more practical and very popular approach is to use cross-validation (CV). The main principle is illustrated in Figure 3.11 [75]. The training set is partitioned into K subsamples, which are often referred to as K-folds in the literature. Typically, the number of folds is set to 5 or even 10 [38]. The CV algorithm then trains the machine learning model on all folds except one, which is used for evaluating the performance. The illustration shows the training folds in green and the validation fold in blue. This is repeated K times, testing on a different fold each time. Finally, the overall cross-validation performance metric for a specific hyperparameter combination is obtained by averaging the individual metrics calculated on the underlying validation folds [38]. In summary, CV uses the data of the training sample more efficiently, although it has the disadvantage of being more computationally demanding. Per tested hyperparameter set, the machine learning model needs to be trained K times — once for every fold. Ultimately, the model is retrained on the whole training sample with the hyperparameters that resulted in the best cross-validation performance. This is then the trained surrogate model of AATOM, ready to be used for further analysis.

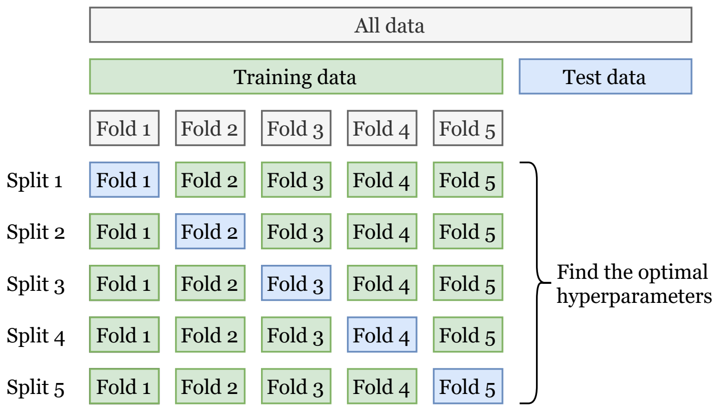


Figure 3.11: Hyperparameter tuning with a K-fold cross-validation approach [75].

3.5. Model Validation

The previous section often mentioned that the performance, prediction error, or validation metrics would be evaluated. This can be done in several ways, so current section will therefore present common indicators from the literature. Undoubtedly, the coefficient of determination R^2 is one of the most popular metrics [16]. It is calculated according to Equation 3.19 [from 11, 14]. Again, $f(\boldsymbol{x}_i)$ denotes the actual response of AATOM for a certain input $\boldsymbol{x}_i$, $\hat{f}(\boldsymbol{x}_i)$ the the surrogate model's predicted response and $\overline{f}$ the average of all true responses in the test set. The size of the sample is given by m.

$$R^2 = 1 - \frac{\sum_{i=1}^{m}\left(f(\boldsymbol{x}_i) - \hat{f}(\boldsymbol{x}_i)\right)^2}{\sum_{i=1}^{m}\left(f(\boldsymbol{x}_i) - \overline{f}\right)^2} \tag{3.19}$$

The performance indicator can be interpreted as follows. It gives the proportion of variation in the response of AATOM that is explained by the input parameters in the surrogate model $\hat{f}$ [16, 26]. Hence, the closer to one, the better the model. While the coefficient of determination is a very useful and informative metric, it does not directly measure the prediction error. An alternative would the root-mean-square error (RMSE), which is the square root of the mean squared error (MSE) as indicated in Equation 3.20 [from 3, 28].

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{\sum_{i=1}^{m}\left(f(\boldsymbol{x}_i) - \hat{f}(\boldsymbol{x}_i)\right)^2}{m}} \tag{3.20}$$

Unlike the coefficient of determination, the RMSE is expressed on the same scale of the response. The benefit is that it indicates the expected prediction error of the surrogate model. It is thus a good measure for its uncertainty. Consequently, the lower the RMSE, the more accurately a meta-model is able to predict the true response of AATOM. Instead of squaring and taking the root, another approach is to use the absolute value of the prediction error. This is the mean absolute error (MAE), as given in Equation 3.21 [from 11, 14].

$$\text{MAE} = \frac{\sum_{i=1}^{m}\left|f(\boldsymbol{x}_i) - \hat{f}(\boldsymbol{x}_i)\right|}{m} \tag{3.21}$$

Logically, the RMSE and the MAE are closely related. There is however an important distinction between the two. Since the RMSE involves squaring the discrepancy between the actual and predicted response, it is naturally more sensitive to outliers [16]. So the greater the difference between the two indicators, the more the surrogate model is affected by outliers. With this in mind, it is always a good idea to evaluate both indicators. Finally, like the RMSE, the MAE is also expressed on the same scale as the response. While this has both advantages and disadvantages, the performance of surrogate models for different responses cannot be compared when they are expressed on other scales. The mean absolute percentage error (MAPE) resolves the issue by calculating the prediction error relatively. It is formulated according to Equation 3.22 [from 16].

$$\text{MAPE} = \frac{\sum_{i=1}^{m}\left|\frac{f(\boldsymbol{x}_i) - \hat{f}(\boldsymbol{x}_i)}{f(\boldsymbol{x}_i)}\right|}{m} \tag{3.22}$$

The relative nature of the MAPE makes it possible to compare the performance for different responses. However, one should be careful when using it on low values. The closer the responses are to zero, the more the indicator tends to exaggerate the error [16]. That is because of the fraction in Equation 3.22, where the prediction error is in the numerator and the true response in the denominator. Logically, when the former is quite large and the latter close to zero, the MAPE explodes easily. This is not necessarily a problem, though the results should be interpreted accordingly.

During the discussion it became clear that the different key performance indicators (KPIs) all have their advantages and disadvantages. For that reason, we do not select just one metric for the overall validation of the surrogate models, but rather compute all four. This gives a better insight into the bigger picture and ensures a stronger interpretation of the results. Notwithstanding, the hyperparameter optimization from section 3.4 requires one indicator to quantify the model performance for the tested parameter combinations. The effect of different metrics on the optimization process is rather marginal, so we choose the most commonly used coefficient of determination for this. A final consideration is the test set. It was mentioned earlier with Figure 3.11 that a separate sample must be isolated for the validation of the models. Indeed, a surrogate model cannot be trained and tested on the same data. The usual procedure is to split the sample in a training and a test set. Typical proportions of data going to the test set are 20% to 25% [34, 38]. However, it is important to understand that this is not as straightforward when the sample has been collected adaptively. Doing so could jeopardize the benefits of active learning, which is of course undesired. We choose to solve the problem by creating an additional data set. More specifically, there will be randomly sampled from AATOM using the input parameter combinations that have not been selected during the adaptive sampling process. This continues until the size of the test set is about 20% of the whole sample. Once again, the sole purpose of the test set is to validate the surrogate models and therefore will strictly not be used for any other purpose.

3.6. Selected Methodology

In conclusion of this chapter, an overview of the selected methods is presented in Figure 3.12. It essentially shows the whole process how the surrogate models will be generated. The first step is the design of experiments. An initial sample is created using a Hammersley sequence with a size of 10 times the number of dimensions in the input vector. This is in line with the recommendations of Loeppky et al. [62], who examined the effect of initial sample sizes on Gaussian process surrogates [61]. After that, the adaptive sampling algorithm takes over. The Gaussian process model is trained on the available data such that the expected improvement for global fit acquisition function can indicate the most interesting region. The identified data point is then sampled from AATOM and the process is repeated. This continues until the stopping criterion is reached. Subsequently, the available training sample is used to train four machine learning models: a second-order polynomial, a Gaussian process regressor, a random forest regressor and a gradient boosting regressor. If the model consists of hyperparameters, they are tuned with a Bayesian optimization algorithm under a K-fold cross-validation strategy. Thereafter, the models are retrained on the whole training set with their optimal parameters. The last step is to validate them. To do so, an independent test set must be sampled first. This is done by randomly picking input parameter combinations that have not been selected by the active learning process until the proportion of the test set equals 20% of all collected data. Finally, the R^2, RMSE, MAE and MAPE are calculated when the test data is available, which then allows the interpretation of the results. After this, the surrogate models are ready for further analysis, as will be explained in chapter 4.

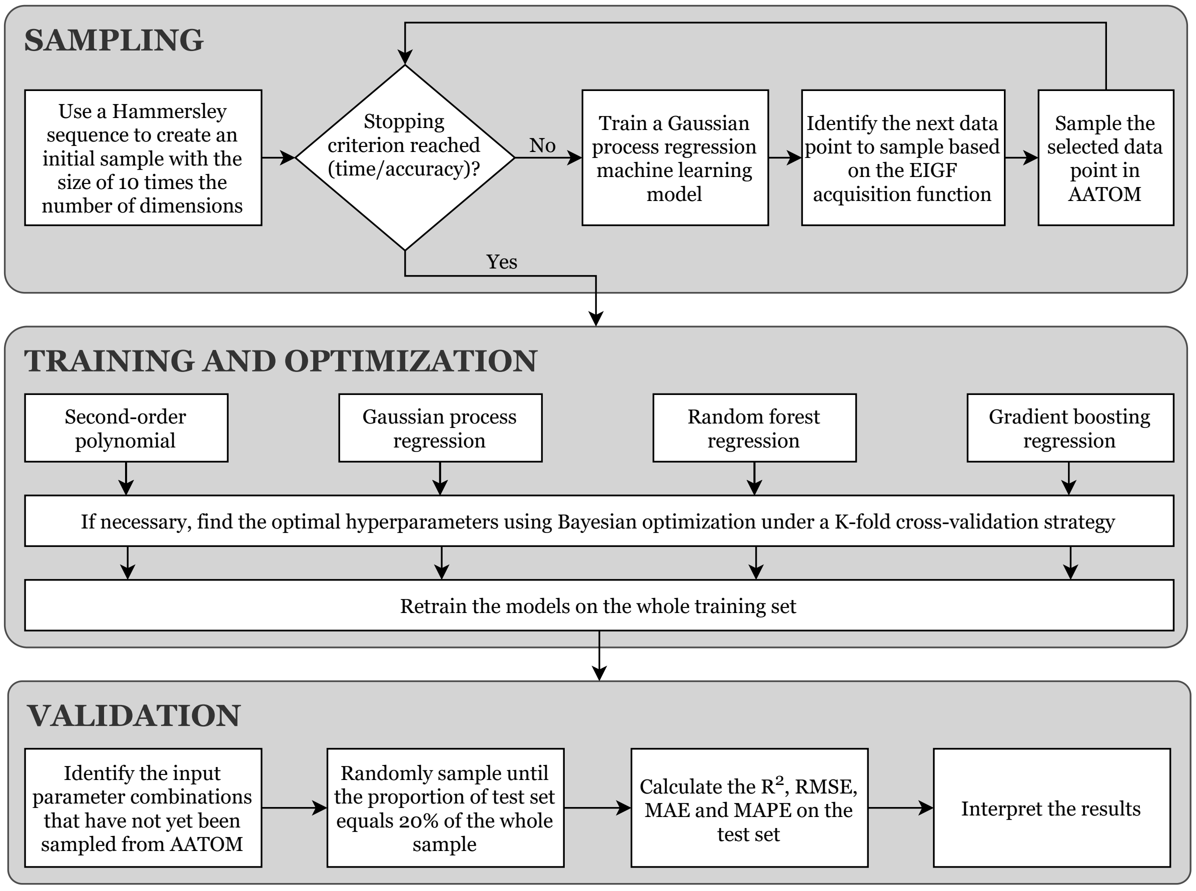

SAMPLING
Use a Hammersley sequence to create an initial sample with the size of 10 times the number of dimensions
Stopping criterion reached (time/accuracy)?
No
Yes
Train a Gaussian process regression machine learning model
Identify the next data point to sample based on the EIGF acquisition function
Sample the selected data point in AATOM
TRAINING AND OPTIMIZATION
Second-order polynomial
Gaussian process regression
Random forest regression
Gradient boosting regression
If necessary, find the optimal hyperparameters using Bayesian optimization under a K-fold cross-validation strategy
Retrain the models on the whole training set
VALIDATION
Identify the input parameter combinations that have not yet been sampled from AATOM
Randomly sample until the proportion of test set equals 20% of the whole sample
Calculate the R², RMSE, MAE and MAPE on the test set
Interpret the results

Figure 3.12: Overview of the selected methodology.

4 System Understanding

When the concept of surrogate modeling was introduced in section 3.1, we argued that it can play an important role in understanding the underlying system. Since surrogates are much faster than the original model, extensive analyses can be performed, but also their intrinsic properties can lead to interesting conclusions. The traditional approach to explain a model's behavior has been to conduct a sensitivity analysis (SA) [96]. It allows to investigate the effect of input parameters on the output, both on a local and a global level. The usefulness of a SA is undisputed, but further research in this area has actually been thriving lately. Fueled by the European Union's adaption of the General Data Protection Regulation, scholars have shown a profound interest in understanding and explaining the reasoning behind machine learning algorithms [58, 72]. Indeed, the black-box nature makes them difficult to interpret for the user. Seemingly simple questions, whether the outcome is trustworthy or not, or how they arrive at their solutions, are actually quite challenging to answer. The literature calls it explainable artificial intelligence (XAI), which even goes so far as questioning whether accuracy or explainability should be optimized — perhaps some of the accuracy can be abandoned to achieve greater transparency [81]. Therefore, due to their identical goals, we will further exploit the common ground of traditional approaches and XAI in an attempt to extract relevant information about the complex dynamics in airport terminals using the surrogate models. The purpose of this chapter is not to provide an exhaustive overview of all available methods for achieving this, but rather to touch upon the most important ones. Neither will there be a selection of the most promising, they are all useful in their own way and can be deployed during the analysis according to the needs. A common distinction in the literature is between methods that are independent of the model and methods that are specific to a particular architecture [71]. Respectively, section 4.1 elaborates on the model-agnostic approaches and section 4.2 on the model-specific ones.

4.1. Model-agnostic Approaches

Interpretation methods that are independent from the machine learning algorithm are referred to as model-agnostic [71]. They can be applied to essentially any model; only the effect of the input parameters on the output is analyzed. The internal working principle of an algorithm is thus disregarded [9, 25]. This makes model-agnostic approaches flexible and gives them a large user base. However, while they do provide insight into the behavior of responses, it remains unclear how exactly the model arrived at its outcome [25]. Nevertheless, only a handful of machine learning models are capable of producing model-specific information, so model-agnostics are often the last resort. Based on the studies by Belle and Papantonis [10], Elshawi et al. [25], Molnar [70], and Borgonovo and Plischke [13], we have divided the most common approaches for regression models into five categories as follows. Subsection 4.1.1 is first with the traditional sensitivity analysis. Secondly, a popular way to gain more knowledge about the nature of the relationships between inputs and outputs, along with their influence, is by creating dependency plots. Three related methods for doing so are described in subsection 4.1.2. Next, subsection 4.1.3 continues with methods to extract the relevancy of features. This includes the well-known importances, but their interactions are also considered. Fourth, subsection 4.1.4 discusses LIME, which stands for local interpretable model-agnostic explanations. It is an common approach for the local interpretation of a machine learning model's output. Lastly, SHAP is reviewed in subsection 4.1.5. This is the abbreviation for Shapley additive explanations, a method based on game theory.

4.1.1. Sensitivity Analysis

First and foremost, there is the sensitivity analysis (SA). The general purpose of a SA is to examine how the responses of a model behave with regard to changes in its parameters. Put differently, it involves assigning response variability to the causative inputs [13, 86]. As would be expected, several ways exist to realize this. We distinguish between local and global approaches, which will now be discussed in respective order.

Local methods analyze the sensitivity of the model around a particular point in the input space [13, 96]. More specifically, the effect on the response is measured per proportional change in one of the input parameters. One way to obtain this information is by evaluating the gradient. Namely, partial derivatives naturally describe the sensitivity along the individual dimensions [13]. While such an approach is common in the literature, the one-at-a-time (OAT) sensitivity analysis is actually much more popular. It produces similar results, and is one of the easiest and most straightforward methods to get insight into the local response behavior [13]. The formal expression is given in Equation 4.1 [from 13]. The main principle is that all input parameters $\boldsymbol{x}^0_{-i}$ are kept constant except one x_i. Then, this feature is changed in the positive direction. The extent Δx_i^+ is at the discretion of the user, one just needs to make sure that all parameters are changed in the same relative manner (e.g., each by 10%). Otherwise, their effects on response would be difficult to compare. The next step is to plug the changed parameter along with the unmodified ones into the machine learning model, resulting in a new output. The sensitivity $\Delta_i^+ \hat{f}$ to the parameter under investigation is ultimately represented by the difference it causes in the response, with respect to the baseline value $\hat{f}(\boldsymbol{x}^0)$. Vice versa, the same can be done for the negative direction.

$$\Delta_i^+ \hat{f} = \hat{f}(x_i + \Delta x_i^+, \boldsymbol{x}^0_{-i}) - \hat{f}(\boldsymbol{x}^0) \tag{4.1}$$

The advantage of the OAT sensitivity analysis is that it is a simple and evident approach that quickly gives a solid indication of the direction and magnitude of a particular parameter change on the model's output value. It does not involve too much complexity. Moreover, the results can be easily displayed in a tornado diagram, which shows the effects sorted by magnitude for both negative and positive changes for all parameters [13]. On the contrary, its greatest weakness is that variable interactions are not considered. For example, input parameters can influence one another, which might result in an entirely different model reaction when they are varied altogether [13]. This could lead to fallacious conclusions and therefore a more sophisticated approach would be appropriate in such non-linear cases.

In contrast to local methods, a global approach examines the model behavior over the entire input space [96]. In its simplest form, a good first step to visualize the global relationships between the different inputs and the output is to create scatter plots [13, 86]. It gives a first indication of the overall picture. However, to obtain more quantifiable insights, one must go beyond that. We consider the most common variance-based approach by Sobol [90], which attributes the variance in a model's response to the input variables in proportion to their contribution [96]. Under the assumption that the inputs are independent from each other, the output of the surrogate model $\hat{f}(\boldsymbol{x})$ may be decomposed according to Equation 4.2 [from 86, 96]. It shows that a response can be expressed by a sum of functions with an increasing number of dimensions, where the total dimensionality of the input vector is denoted by n.

$$\hat{f}(\boldsymbol{x}) = \hat{f}_0 + \sum_{i=1}^{n} \hat{f}_i(x_i) + \sum_{i=1}^{n}\sum_{j>i}^{n} \hat{f}_{ij}(x_i, x_j) + \sum_{i=1}^{n}\sum_{j>i}^{n}\sum_{k>j}^{n} \hat{f}_{ijk}(x_i, x_j, x_k) + \cdots + \hat{f}_{ijk\ldots n}(x_i, x_j, x_k, \ldots, x_n) \tag{4.2}$$

From here, Sobol [90] proved that the variance can also be broken down. This is the famous analysis of variance decomposition, as given in Equation 4.3 [from 86, 96]. Essentially, the expression demonstrates that the total variance of a machine learning model's output can be split up into separate parts, which are directly related to the corresponding input parameters and their interactions.

$$\sigma^2 = \sum_{i=1}^{n} \sigma_i^2 + \sum_{i=1}^{n}\sum_{j>i}^{n} \sigma_{ij}^2 + \sum_{i=1}^{n}\sum_{j>i}^{n}\sum_{k>j}^{n} \sigma_{ijk}^2 + \cdots + \sigma_{ijk\ldots n}^2 \tag{4.3}$$

If all terms are then divided by the total variance σ^2, we arrive at the standardized Equation 4.4 [from 86]. It is the sum of so-called Sobol sensitivity indices, which naturally adds up to one. The index of a particular term is a direct measure of its sensitivity. Hence, it shows how much the variance of the model's output is affected by the term [86]. This applies not only to the first-order indices S_i, one for each input dimension, but also to the higher-order interactions S_{ij}, S_{ijk}, etc.

$$\sum_{i=1}^{n} S_i + \sum_{i=1}^{n}\sum_{j>i}^{n} S_{ij} + \sum_{i=1}^{n}\sum_{j>i}^{n}\sum_{k>j}^{n} S_{ijk} + \cdots + S_{ijk\ldots n} = 1 \tag{4.4}$$

Analyzing the first and higher-order indices gives a good idea about the global sensitivity. Notwithstanding, the number of indices can easily explode with an increasing dimensionality. This is resolved by calculating the total index for a certain input parameter. A three-dimensional example is given by Equation 4.5 [from 86]. The total sensitivity index for the second dimension S_{T2} is the sum of all components from Equation 4.4 in which that dimension was involved. Thus, it measures the total contribution of a parameter with respect to the variation in the output of a machine learning model [86].

$$S_{T2} = S_2 + S_{12} + S_{23} + S_{123} \tag{4.5}$$

The global variance-based SA is widely adopted because it gives a full picture of the sensitivity. The variation in a model's output can be rather easily attributed to particular input parameters and interactions [86]. Moreover, the results are straightforward to interpret and non-linear models can be analyzed without any issues [90, 96]. Nonetheless, the method tends to be computationally expensive, as calculating the indices requires many model interrogations [86]. This clearly shows one of the biggest benefits of using a surrogate instead of AATOM itself — it would take a lot of time otherwise. Finally, one should be careful about taking the variance as a true proxy for the variability in the response [13, 96]. Technically, the two concepts are not entirely equivalent, although it does provide a solid and robust understanding of the sensitivity of the variables.

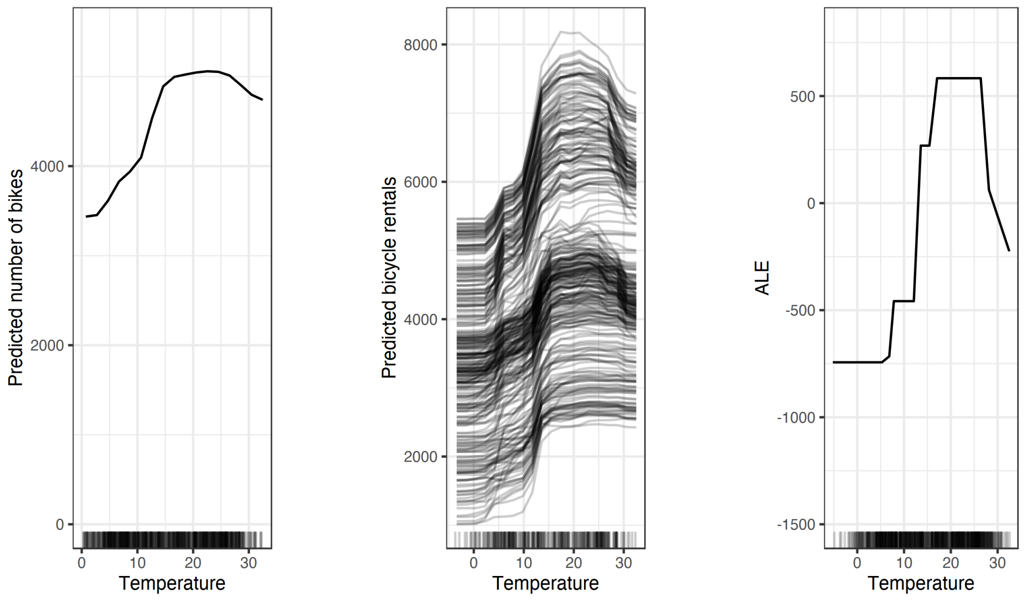
4.1.2. Dependency Plots

A sensitivity analysis shows to what extent the output of a machine learning model is affected by changes in the input, but says nothing about the actual form of the relationship. Such information can be obtained by analyzing the dependency of the outcome on the input parameters [10, 70]. The strength of this approach is that one can easily see how responses behave as a function of a feature. For example, it can detect whether the nature of the relation is (non-)monotonic, (non-)linear, and so on [70]. The literature on XAI describes three general trends to visualize the dependency, being the partial dependence plot, the individual conditional expectation and the accumulated local effects [25, 70, 71]. They are commonly abbreviated as PDP, ICE and ALE. The three methods are closely related, and will now be further discussed in respective order.

The concept of a PDP was introduced by Friedman [30]. It is a global method which essentially shows the marginalized effect of a particular input variable on the response of a machine learning model [70]. Formally, the partial dependence PD is described by Equation 4.6 [from 30, 70]. In practice, however, it is approximated by the summation on the right. The investigated features are denoted by $\boldsymbol{x}_s$; usually this is only one, although it is also possible to consider two at once. All other features are represented by $\boldsymbol{x}_c$, p is the probability density function, and m is again the size of the sample. Then, the outcome's partial dependence on $\boldsymbol{x}_s$ is calculated by taking the mean response value for all varying $\boldsymbol{x}_c^{(i)}$ in the sample [70]. This results in a function that shows how the model behavior is affected by solely $\boldsymbol{x}_s$.

$$\mathrm{PD}_{\boldsymbol{x}_s}(\boldsymbol{x}_s) = \mathbb{E}_{\boldsymbol{x}_c}\left[\hat{f}(\boldsymbol{x}_s, \boldsymbol{x}_c)\right] = \int \hat{f}(\boldsymbol{x}_s, \boldsymbol{x}_c)\, p(\boldsymbol{x}_c)\, d\boldsymbol{x}_c \approx \frac{1}{m}\sum_{i=1}^{m} \hat{f}\left(\boldsymbol{x}_s, \boldsymbol{x}_c^{(i)}\right) \tag{4.6}$$

To see what dependency plots look like in practice, an example of bicycle rental is shown in Figure 4.1. The results are retrieved from Molnar [70]. Essentially, the example aims to predict the daily number of bicycles that will be rented from a particular company, based primarily on weather-related information. A random forest machine learning algorithm is trained to make the predictions. One of the features is the temperature in degrees Celcius on the day in question, which we will now further examine. Figure 4.1a depicts the PDP on the left. The plot indicates that the relationship between temperature and the predicted number of rental bicycles is non-linear and non-monotonic. Initially, the number of bicycles increases with warmer temperatures, but it reaches a ceiling between 15 and 25 degrees. If the temperature rises further, then the number of rentals starts to decrease again. This seems logical: people like to cycle with a comfortable temperature, which is neither too cold nor too hot. The main benefit of visualizing the dependency between features and a machine learning model's response this way is its intuitiveness [10, 25, 70]. There is no ambiguity and a PDP can easily be understood by the user. In addition, they are not technically challenging to generate in practice [70]. An important drawback, however, is that they assume strict independence between the input variables [25, 70, 71]. That could be a problem if some features are correlated — averaging the model's outcome over $\boldsymbol{x}_c^{(i)}$ in Equation 4.6 can lead to parameter combinations that are unlikely in reality. Another disadvantage of PDPs is that it is practically impossible to visualize the dependency for more than two input variables [70]. The plot would become too complex to interpret otherwise. Also, the distribution of the sampled data points should ideally be considered at the same time, as can be seen just above the horizontal axis in Figure 4.1a. The risk



(a) Partial dependence plot. (b) Individual conditional expectation. (c) Accumulated local effects.

Figure 4.1: The three approaches to plotting dependencies, applied on an example of bicycle rental [70].

of not doing so is that too much attention is paid to parts in the domain about which truthful information is actually rather scarce [70]. A final weakness of PDPs is that heterogeneity cannot be detected [10, 70]. Indeed, only the average relationship is considered instead of individual instances. This carries the risk that important information can be hidden in the average, potentially leading to erroneous conclusions.

One way to solve the average-related issues of PDPs is to simply not take the average, and so the individual conditional expectation (ICE) was introduced. Instead of averaging the model's response over $\boldsymbol{x}_c^{(i)}$, one can also plot all individual cases [70]. Hence, for all instances i in the sample with size m, the ICE draws every $\hat{f}(\boldsymbol{x}_s, \boldsymbol{x}_c^{(i)})$ in one plot over the domain of $\boldsymbol{x}_s$. Note that this is the same function that was originally averaged with the approximation term on the right in Equation 4.6. Consequently, the curve of a PDP is in fact the mean of all the individual curves in an ICE plot. Applying it to the same bicycle rental example as before, the ICE plot is shown in Figure 4.1b. The graph does not demonstrate any major differences among the instances. A conclusion is therefore that the relationship between the temperature and the number of rental bicycles is fairly homogeneous [70]. Thus, for this case, the PDP is a valid summary of the nature of the interconnection. The advantages and disadvantages of PDPs and ICE plots are similar because of their kinship. ICE plots remain intuitive and can still be easily understood by the user [25, 70]. However, an area where ICE plots are superior to PDPs is their ability to reveal heterogeneity in the dependency of a model's outcome on a feature [10, 70]. For that reason alone, it is always a good idea to interpret both plots side by side. Conversely, ICE plots no longer allow to analyze the dependency of the response on two features simultaneously, and they tend to get hectic due to all the individual curves [25, 70]. Although, like PDPs, the main drawback remains their assumption of strict independence between the input variables [25, 70]. To illustrate the issue, if the solar irradiance was one of the other features of the bicycle rental example, a positive correlation with the temperature would be expected. One may then wonder how to interpret the ICE curves in the upper region of the temperature domain for instances that actually have a very low irradiance. Indeed, some of the parameters combinations could be nonexistent in reality, which can lead to erroneous visualizations.

While PDPs and ICE plots are popular methods in the field of explainable artificial intelligence, the fact that they assume the absence of any correlation between the input variables has sparked the interest of scholars to come up with an alternative approach. Apley and Zhu [7] therefore introduced the accumulated local effects (ALE) plot. The mathematics behind ALE plots are rather complicated, so for the sake of the literature survey we will explain them intuitively. First and foremost, the essence remains the same as for PDPs and ICE plots: they intend to visualize the dependency of a machine learning model's response to a particular feature [70]. However, ALE plots distinguish themselves by the way they do this. Recall again the bicycle rental example. To obtain the expected number of bicycles, the PDP uses the mean response for a particular temperature, say 15 degrees Celcius. To mitigate the assumption of feature independence, an ALE plot uses the conditional distribution of an input parameter instead of the marginal [70]. Concretely, the domain of the investigated feature is split into a number of intervals. The local effect is then obtained by subtracting the response value for the feature assuming the interval's lower bound from the response value for the feature assuming the upper bound. This is repeated for all data points that are located in the interval, after which the average difference is calculated. Applied to the rental example; if the bin width of the interval were two degrees, the local effect at 15 degrees Celcius would be the mean difference in predicted bicycles between 16 degrees and 14 degrees. The strength of using differences rather than average response values is that it isolates the effect of the investigated feature from any other influences [70]. While the benefit of this approach is evident, interpreting the differences as they are would be challenging. Therefore, ALE plots accumulate the results across the intervals and center the ultimate curve around zero [70]. To illustrate this with an example, the plot for the effect of temperate on the predicted bicycle rentals is shown in Figure 4.1c. The interpretation is as follows. If the temperature equals 20 degrees Celcius, then its primary effect on the number of rental bicycles is about 575 bicycles greater than average [70]. So an ALE plot must be seen as the reflection of a feature's principle effect against the mean outcome of a machine learning model, at a specific location in the domain. Hence, a wrongful interpretation is to consider the curve as the expected behavior of the model when varying the respective input parameter, as argued by the creators Apley and Zhu [7]. Logically, the main advantage of ALE plots is that they are more reliable than PDPs or ICE plots, since feature independence is not assumed [7, 70]. This would be especially true in the case of correlation among the input variables. Moreover, they are again fairly straightforward to understand and are also much more efficient from the computational perspective [7, 70]. They can be considerably faster than PDPs. Notwithstanding, ALE plots do entail some drawbacks. They are definitely more complex than PDPs and ICE plots, and require the determination of the ideal bin width for the intervals [70]. On top of that, detecting heterogeneity is fairly difficult, if not impossible. A final notion is that it remains a burden to separately analyze input parameters that are actually correlated, even though ALE plots do not require them to be independent. In such cases, a better alternative may be to jointly consider their effect on the outcome of the model [70].

4.1.3. Feature Relevancy

Since the outcome of a machine learning model is expressed as a function of the input variables, one may wonder which ones are more determinative than others. In this sense, examining the feature relevancy is a useful model-agnostic approach to understanding them. We will discuss feature importances first and then feature interactions. Both are global methods, with the former measuring how strongly the model depends on the individual parameters and the latter to what extent parameters influence one another [25, 71].

The principle behind calculating feature importances is actually rather evident. The idea was concretized by Fisher et al. [27] in the field of XAI [70]. In essence, the method relies on perturbations. Once a machine learning model is fitted, the first step is to select an error metric. Most likely, this is one of the performance indicators from section 3.5. Thereafter, the respective feature is randomly permuted over its domain and the associated model error is constantly reassessed. This process is performed on all input variables, after which the information can be plotted in one graph. A box plot is drawn of the increases in model error due to the permutations for each feature. They are listed in decreasing order to enhance the interpretability of the graph. A feature is considered more important as the discrepancy with the original error increased during the permutation process [70]. Put differently, the more a model is dependent on a particular parameter, the more the prediction error enlarges when it has no longer access to the respective information. Calculating feature importances is popular because it provides one comprehensive overview of the bigger picture [27, 70]. Not only are they very understandable, they also take interactions into account. Namely, the permutation process eradicates any influence of the examined feature, so the result represents its total importance. Lastly, the method needs to interrogate the machine learning model a few times for the calculations, but it is not computationally demanding as retraining the algorithm is not necessary [70]. On the other hand, a drawback of permutation-based feature importances is that there is no consensus yet on whether to calculate the results on the training sample or on the test set. There are valid arguments for both options [70]. In addition, the randomness caused by the perturbations can lead to instability in the results. This can be solved by increasing the number of permutations per feature, although that naturally requires more calculations [70]. One has to find an optimal balance. A much bigger issue is that the feature importances are also prone to the same assumption of PDPs and ICE plots. Indeed, they deem the input variables to be completely independent. According to Molnar [70], this entails two threats. On the one hand, the random permutations could result in infeasible parameter combinations that may be used during the calculations. On the other hand, if two or more features are correlated, their true importance may be divided amongst them. Such cases are rather difficult to interpret. Finally, an important note is to not confuse the current approach with the popularly used feature importances of regression tree (ensemble) models, which use the node impurity [27]. The latter is model-specific and calculates how much a feature can diminish the variance on average [70].

Since importances cannot distinguish between main and interaction effects, they often go hand in hand with the calculation of feature interactions. This is the phenomenon where input variables exert influence on one another so that the response of a machine learning model is no longer a linear addition of the main effects [71]. A post hoc approach to investigate the strength of interactions is to isolate their impact on the variability in the outcome of the model. If these results are then divided by the total variability, proportional influences are obtained [25, 70]. This way of reasoning is the principle idea behind the seminal work of Friedman and Popescu [31], who proposed the H-statistic [70]. The theory is based on partial dependencies, as discussed with Equation 4.6 in the previous subsection 4.1.2. The expression for the H-statistic of a particular feature j with any of the other parameters is given in Equation 4.7 [from 31, 70].

$$H_j^2 = \frac{\sum_{i=1}^{m}\left[\hat{f}\left(\boldsymbol{x}^{(i)}\right) - \mathrm{PD}_j\left(x_j^{(i)}\right) - \mathrm{PD}_{-j}\left(\boldsymbol{x}_{-j}^{(i)}\right)\right]^2}{\sum_{i=1}^{m}\hat{f}\left(\boldsymbol{x}^{(i)}\right)^2} \tag{4.7}$$

The sample size is again denoted by m. In essence, the numerator offsets the model's response against the partial dependence on feature j and the partial dependence on all the others except j. What remains of the total variance can therefore be associated to solely the interaction effects of x_j. This is then normalized by the denominator to arrive at the proportion of the model's variability that is described by the feature's interactions [31, 70]. Hence, the H-statistic is always between zero and one, with a value of zero corresponding to an input parameter that does not affect any other parameters. Subsequently, Friedman and Popescu [31] recommend to further examine those features with the greatest values for the evaluated statistic. In particular, they refer to the interaction effects between two variables j and k, which can be obtained according to Equation 4.8 [from 31, 70]. This is the expression for the H-statistic between two specific features.

$$H_{jk}^2 = \frac{\sum_{i=1}^{m}\left[\mathrm{PD}_{jk}\left(x_j^{(i)}, x_k^{(i)}\right) - \mathrm{PD}_j\left(x_j^{(i)}\right) - \mathrm{PD}_k\left(x_k^{(i)}\right)\right]^2}{\sum_{i=1}^{m}\mathrm{PD}_{jk}\left(x_j^{(i)}, x_k^{(i)}\right)^2} \tag{4.8}$$

Similar to the earlier Equation 4.7, current H-statistic isolates the pure variance of the partial dependence from x_j and x_k that is caused by the influence the two parameters exert on each other [31]. The value is also normalized, so a statistic of zero means that there is no interaction between x_j and x_k. Vice versa, a statistic of one means that the model is not subject to any of their main effects [70]. In that case, the response is thus only affected by the interactions among parameters j and k. To emphasize once again the difference between the two H-statistics H_j^2 and H_{jk}^2; the first represents the total interaction strength of x_j on any of the other features, while the second represents the specific two-way interaction strength between x_j and x_k on the machine learning model [70]. The fact that H-statistics are dimensionless and unambiguously clear is one their main advantages. There is no doubt because the statistic directly measures the proportion of model variability due to the examined feature(s) [70]. Moreover, they do not assume any functional form between the parameters — all types of interactions are considered [25, 70]. A last benefit is that they allow to calculate interactions of even higher orders than presented, though this also increases the complexity [31]. Conversely, one of the biggest drawbacks is its computational intensity, as the calculations require many model interrogations [25, 70]. A possible approach to alleviate the burden is by not using all the m data points, but that would also make the statistic less stable. One must therefore carefully weigh both concerns. Another issue according to Molnar [70] is the absence of a proper test statistic for hypothesis testing. While the H-statistic measures the strength of interactions, it is rather hard to objectively assess from which threshold the effect should be considered significant. Finally, the method again suffers from the assumption of feature independence [70]. This makes sense, because it is directly based on partial dependence functions, which are subject to the presumption. Nevertheless, H-statistics remain a useful tool to understand the behavior of machine learning models, especially in extension to analyzing two-way PDPs [70].

4.1.4. Local Interpretable Model-agnostic Explanations

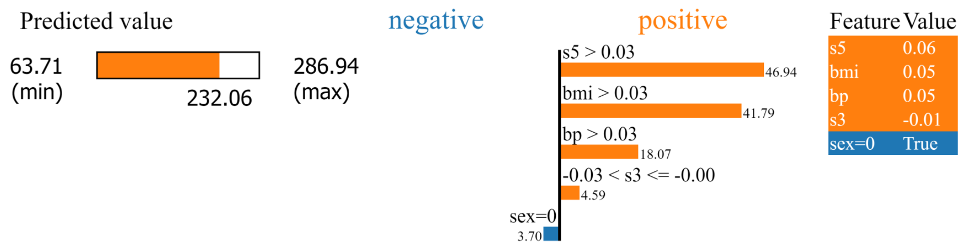
The next considered method for interpreting model behavior is LIME, which is short for local interpretable model-agnostic explanations. It originated from the research by Ribeiro et al. [80], and is currently one of the most popular approaches in the field of XAI [10]. From a high-level perspective, the idea of LIME is to explain the reasoning behind black-box algorithms by means of training a local model that is transparent, and therefore easy to interpret [10]. This requires some further explanation. First of all, it is a local technique. So the purpose is not to explain the machine learning model's behavior across the entire domain, but rather for smaller demarcated regions of interest. Furthermore, LIME uses transparent surrogates to facilitate the local explanations. The choice of the meta-model is not etched in stone, although it is common to use linear regression or decision trees, preferably with regularization [70, 80]. Altogether, the creators emphasize that LIME should be locally faithful [80]. This means that the explanations of the surrogate at a particular point in the domain must be in accordance with the local behavior of the black-box algorithm. An implication is that the usage of local characteristics is enabled. Hence, features that are of great importance on a global level may not be relevant at all in a local region, and vice versa [80]. More formally, the principle behind LIME can be written according to Equation 4.9 [from 70, 80].

$$\xi(x) = \underset{g\in G}{\arg\min}\left[\mathscr{L}\left(\hat{f}, g, \pi_x\right) + \Omega(g)\right] \tag{4.9}$$

In the expression, $\xi(x)$ is the local explanation of LIME at the point x. The purpose is then to find a surrogate g from the set G with transparent and interpretable models, which is as faithful as possible to the black-box's local response $\hat{f}$ without being too complex [70]. The fidelity is represented by a loss function $\mathscr{L}$, which in practice would be one of the error metrics from section 3.5. At the same time, one should refrain from making the surrogate too opaque, as indicated by $\Omega(g)$. This term can be regarded as the regularization of the meta-model. Examples are to limit the decision trees' depth or to increase the number of regression weights that are equal to zero [80]. Lastly, the extent to which LIME considers a region around x to be local is described by π_x. In other words, this parameter thus defines the part of the domain to be used for generating explanations in the local behavior of the model [70].

Now that the philosophy is clear, we take a closer look at how exactly LIME obtains its insights. The explanation is based on the step-by-step descriptions by Molnar [70] and Ribeiro et al. [80]. Recall that it is a local method, so one has to choose a particular location of interest x first. Subsequently, data is sampled by perturbations in the feature space and the corresponding responses are retrieved from the black-box algorithm. The closer a data point is to x, the more influence it is assigned through π_x. In practice, this is accomplished by a kernel function measuring the proximity, with the points closest to x given the highest weights. Next, the weighted sample is fed to a transparent surrogate. The model is then fitted to the data, after which its internal characteristics can be interpreted to explain the local behavior and phenomena of the original black-box algorithm. An example of how this looks in practice can be found in Figure 4.2 [based on 24, 80]. A random forest algorithm is used to predict people's development of diabetes a year after performing various tests. Its target value is expressed on a quantitative but unitless scale. The results of LIME are generated based on an L2-regularized regression meta-algorithm. In the visualization, the behavior of the model is explained at a particular point x in the domain for which the disease was estimated to become 232.06. This is depicted in the lower left, along with the range of the black-box function. The five most important (standardized) features are tabulated at the right under "Feature Value". In the middle, the individual contribution of each variable is visualized. The stronger their impact, the higher they are listed. For example, if parameter $s5$ is greater than 0.03, it results in an increase of 46.94 to the intercept of the fitted surrogate. Vice versa, the prediction is decreased by 3.70 because the feature value of sex equals zero. The intercept and the prediction of the local

```
Intercept 125.1496032579106
Prediction_local [232.83322279]
Right: 232.06
```



Feature	Value
s5	0.06
bmi	0.05
bp	0.05
s3	-0.01
sex=0	True

Figure 4.2: Example of local insights by LIME, applied to data from a study on diabetes. The data comes from Efron et al. [24] and the results were obtained using a tutorial on the code of Ribeiro et al. [80] that was attached to their research.

meta-model are shown at the top left. To connect the dots, LIME arrives at an outcome of 232.83 by adding the individual contributions of all important parameters to the intercept[1]. One can see that this result is very close to the original outcome from the random forest model of 232.06. In summary, the essence of LIME is to interpret the local behavior of black-boxes by means of fitting a transparent surrogate on perturbed data, where nearby data points have a greater influence on the explanations [70].

One of LIME's greatest strengths is its overall flexibility [70, 79]. Essentially, the user can build and apply the method as needed; it is fully customizable. This results in many possibilities for a wide variety of applications, but an inherent disadvantage is that it also leads to more degrees of freedom during the implementation. For example, one might be wondering which type of surrogate is best to approximate the black-box algorithm, how to define the kernel function, what the best approach is to select the most important features, or how to perturb the data when collecting a sample [70, 79]. Answering these kinds of questions is not always trivial and may take a number of iterations under a trial and error strategy. On top of that, the complexity of the meta-model must be carefully weighted against its explanatory power. These two concepts are oftentimes conflicting. Notwithstanding, once the architecture for a particular case has been determined, Ribeiro et al. [79] praise LIME's low switching costs to different black-boxes. This makes it easy to compare the results for different models. Furthermore, despite its flexibility, the method remains rather straightforward to understand and utilize [70]. This also applies to their explanations, which are not too different to interpret. Moreover, they include metrics for the reliability, helping critical users to put their results into perspective [70]. Finally, a last concern by Molnar [70] is that the insights produced by LIME can be quite unstable. For example, the conclusions from nearby data points may be completely different. One should therefore always keep this in mind and preferably investigate the robustness of the explanations in more detail.

4.1.5. Shapley Additive Explanations

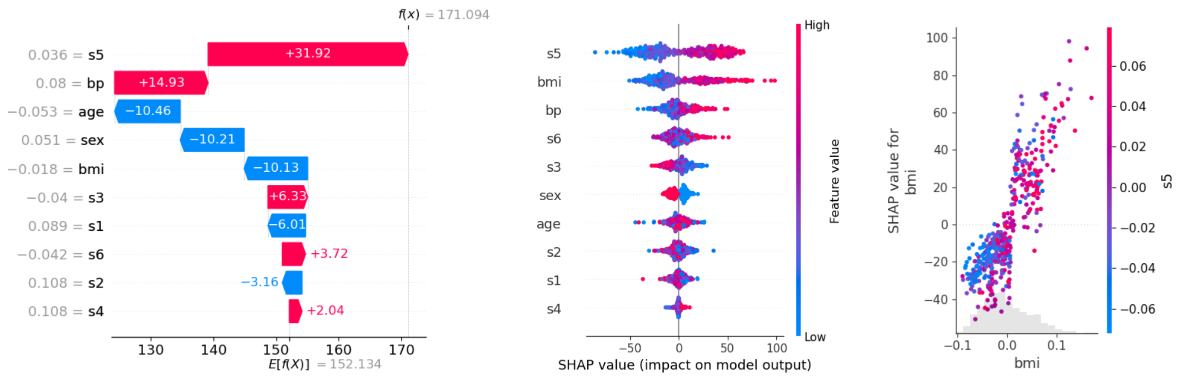
Finally, we discuss SHAP, which next to LIME is also one of the more recent additions to model-agnostic approaches in XAI research. It stands for Shapley additive explanations and is considered an eminent interpretation technique [10]. This is mainly due to its solidified theoretical foundation, which the alternatives often lack [10, 70]. To give an example, Molnar [70] mentioned that LIME explains the behavior of a model by fitting a local surrogate around the point of interest, but there is no theoretical basis to prove the validity of such an approach. This is in stark contrast to SHAP, which is based on cooperative game theory [10]. Lundberg and Lee [63] applied these long-standing fundamentals to interpret the local responses of machine learning models. At the heart of the method are Shapley values, which is a way of estimating the individual contributions of the input parameters to the model's outcome. The high-level principle is as follows, with features seen as game players and the local response as payoff. First, various coalitions are constructed from all possible subsets of the players, both inclusive and exclusive a particular feature in question. The machine learning model is then retrained on all coalitions, each eliciting different reactions from which marginal contributions can be obtained. These can be weighted over all possibilities, so that their sum results in a certain value for the examined feature in accordance with its contribution to the payoff [10, 63, 70]. This is the Shapley value, which essentially divides the prediction (payoff) in proportion to the impacts of the respective input variables (players) [25]. In other words, it gives a fair view on how a machine learning model's output is affected by which features. The mathematical expression is shown in Equation 4.10 [from 63, 70].

$$\phi_{x_i} = \sum_{S \subseteq \{x_1, x_2, \ldots, x_n\} \setminus x_i} \frac{|S|!\,(n - |S| - 1)!}{n!} \left[\hat{f}_{S \cup \{x_i\}} \left(x_{S \cup \{x_i\}} \right) - \hat{f}_S \left(x_S \right) \right] \tag{4.10}$$

The Shapley value of a particular feature i at a certain location x in the domain is denoted by ϕ_{x_i} and the subset S represents the possible combinations of the n features except x_i. For all S, the machine learning model is first trained and evaluated on the coalition including x_i, and then on the same, but this time excluding x_i. Thereafter, their difference is taken as can be seen by the expression's subtraction on the right. The discrepancy is the marginal contribution of x_i to the prediction for that particular coalition S. Subsequently, this result is multiplied by the fraction on the left, which yields a weight as a function of the subset size $|S|$ and the total number of features n. The magnitude of the coefficients is U-shaped in relation to the number of players in the subset, so both smaller and larger coalitions are given more weight. Ultimately, by adding the weighted marginal contributions of x_i for every coalition together, one obtains the Shapley value of a particular feature i at location x [63, 70].

While at first, the theoretical background of SHAP may seem rather abstract, the method becomes more accessible when applied in practice. For that, we revisit the diabetes example from LIME in subsection 4.1.4.

[1] That is, $232.83 \approx 125.15 + 46.94 + 41.79 + 18.07 + 4.59 - 3.70$. Small deviations are due to rounding.



(a) Waterfall plot. (b) Beeswarm summary plot. (c) Dependence scatter plot.

Figure 4.3: Example of the possibilities with SHAP, applied to data from a study on diabetes. The data comes from Efron et al. [24] and the results were obtained using a tutorial on the code of Lundberg and Lee [63] that was attached to their research.

Three common visualizations are presented in Figure 4.3 [based on 24, 63]. On the left in Figure 4.3a, there is the waterfall plot which shows the Shapley values of all features at a particular data point $\boldsymbol{x}$. The chart should be read from bottom to top. Namely, it starts from the mean response: 152.13 in this case. Then, the contribution of each feature is listed in ascending order, until it reaches the model's response 171.09 for the instance $\boldsymbol{x}$. With that in mind, an important element for proper interpretation is to view Shapley values as the contribution of their respective input variables to the discrepancy between the actual and the average response at a given point $\boldsymbol{x}$ in the domain [70]. In the example, $s5$ affected the outcome by 31.92 in the positive direction from its expected value. Note that Shapley values can be either negative or positive and they are expressed in the same unit as the target value. Waterfall plots are an excellent tool for local explainability, however, it would be useful if feature contributions can be visualized for multiple data points in the feature space. Hence, Lundberg and Lee [63] introduced beeswarm summary plots, as depicted in Figure 4.3b. This shows at one glance how the machine learning model is being influenced. For all $\boldsymbol{x}$ in the sample, the Shapley values are calculated for each parameter and plotted along the horizontal axis. The features are sorted vertically according to a descending mean absolute SHAP value. Furthermore, one dot represents one instance, while the dots are colored accordant with the value of the respective feature. To give a specific example, the higher the value of $s3$, the more negative its Shapley value, meaning that the response is affected more in the negative direction and vice versa. Lastly, the vertical dispersion indicates the density of the data points. The third graph in Figure 4.3c is useful to visualize the relationship between a feature and its Shapley values. In this case, it shows that a higher bmi affects the target value positively compared to the mean. But there is more, because the plot also allows to show interactions with other parameters using colors. For example, the machine learning model predicts that patients with a (standardized) bmi of about -0.025 have a lower value for the development of diabetes within a year if their (standardized) test results of $s5$ are high, and vice versa. Finally, the distribution of the data points over their input space is displayed in gray at the bottom. Notice that this differs from the vertical dispersion in the beeswarm plots, as these show the distribution relative to the domain of the Shapley value. In short, SHAP is a versatile and powerful approach, which makes it possible to clearly visualize and interpret the contribution of features to the prediction of a machine learning algorithm at both a local and global level [70].

Looking at SHAP's strengths, it makes sense that it has become one of the more prominent methods for interpreting the behavior of machine learning models. All types of influences and interactions are taken into account in a fair way, creating a complete picture of how responses are influenced [70, 71]. Moreover, this is achieved on the basis of valid theoretical arguments [10, 25, 70]. Another advantage is that SHAP has the ability to fully dissect the change in model output with respect to an unambiguous datum — the mean prediction [70]. The different influences can then be linked back to their origin. Furthermore, since global explanations are made up of local ones, they are consistent with each other [70]. This is quite a unique benefit over the alternatives. Notwithstanding, SHAP is also exposed to some disadvantages. Scholars are especially vocal about its computational requirements [10, 25, 70, 71]. In many cases it is infeasible to obtain the exact Shapley values. To remedy that, the creators Lundberg and Lee [63] suggested to approximate them, though one has to accept the ensuing consequences, such as additional assumptions [10]. Also, SHAP leads to definite values as is, which must be interpreted accordingly. This is in contrast to LIME, for example, where the result is a transparent model. These models allow for further analysis, like examining the effect of small perturbations [70]. It makes sense that this is more difficult with Shapley values. Lastly, as with several of the other interpretation methods, the features are again presumed to be independent and therefore any possible correlation between them is ignored [70].

4.2. Model-specific Approaches

Unlike model-agnostics, model-specific approaches explain the behavior of machine learning algorithms from within [25, 71]. They use internal properties to interpret how responses come about. Consequently, the application of these methods is severely restricted to algorithms that are of the same nature, as they depend on intrinsic information. This is an evident drawback, but it should be borne in mind that model-agnostic approaches do not explicitly reveal the internal working principles of a model [25]. They do explain how the outcome is affected, albeit on the basis of an intermediate step where proxies are the rule rather than the exception. For direct interpretations, one should resort to model-specific approaches. Logically, the more transparent a machine learning model, the easier it becomes to understand its internal reasoning [70]. That was the main reason in subsection 3.3.7 why we specifically selected a second-order polynomial as part of the alternatives. Namely, it allows a thorough analysis of the regression coefficients, which must be one of the most elementary approaches of model-specific analysis [70]. The method is further discussed in subsection 4.2.1. The second considered architecture was Gaussian process regression. While highly praised as powerful surrogate models, their internal reasoning remains tremendously difficult to understand. Surprisingly, the academic efforts on making them interpretable are actually rather scarce. An exception is the very recent study by Yoshikawa and Iwata [108], who presented a promising method for discovering local feature contributions. Notwithstanding, due to its recency, practical implementations are not yet available to the best of our knowledge. We therefore restrict ourselves to solely model-agnostic approaches for the Gaussian process regression model. This is different for the last two considered surrogates: random forests and gradient boosting regression. Recall from subsection 3.3.5 that both models are made up of individual regression trees. Hence, in principle they are not complicated, but their ensemble structure introduces a lot of complexity. That is why they are extremely powerful, but the downside is that their internal reasoning becomes difficult to interpret. To overcome this and take advantage of both worlds, Friedman and Popescu [31] proposed a method to extract the key decision rules [70]. It is known in the literature as RuleFit, which is further elaborated upon in subsection 4.2.2.

4.2.1. Analysis of Regression Weights

Linear regression is one of the most established procedures for modeling statistical relations. Since the details behind the architecture were already explained in subsection 3.3.1, they are not discussed further here. The model is popular because it is not only very transparent, it also comes with various techniques to assess its validity [9]. This is important, because the approach is not sophisticated. As a result, capturing more complex relationships can be challenging, which could jeopardize the overall accuracy [70]. Hence, before pursuing any further analysis, one should first make sure that the model is sufficiently trustworthy. It would be pointless to infer about system dynamics if the surrogate model cannot mimic the responses of AATOM after all. Logically, this includes the common validation metrics from section 3.5, but specific to linear regression are also the statistical significance tests [9]. For example, the F-statistic can be used to test whether the dependent variable is influenced by the independent variables. In this case, the null hypothesis is that the features do not affect the output, as opposed to the alternative that the model fits better by including them. Similarly, this can also be done for the effects of individual input parameters, where the statistical significance can be determined with t-statistics [9, 38]. These are well-known and widely accepted tests to evaluate whether and which parameters truly influence a response. Furthermore, they also enable calculating the confidence intervals of the regression coefficients, which helps to extract the certainty of the features' true effects [38].

If the linear model proves to be valid, one can use it for further analysis. In particular, the regression weights form the basis for drawing conclusions about the dynamics of the underlying system. For example, should the OLS method define the coefficient of some continuous feature to be 100, then it can be interpreted as the effect on the response of one unit change [70]. In other words, if the value of the feature increases by one, ceteris paribus, the value of the response increases by 100 and vice versa. Regression weights are in that sense a global measure of sensitivity. However, it is crucial to understand that these coefficients have dimensions, and therefore an order of magnitude. For example, if the model expresses the outcome in degrees Celcius and one of its input variables is expressed in kilograms, then the regression weight will have the unit degrees Celcius per kilograms. As a result, different coefficients are likely to be differently scaled. This makes them rather challenging to compare and does not explain which features have the largest impact on the dependent variable. Nevertheless, Borgonovo and Plischke [13] mentioned that analyzing regression weights is a successful method for global sensitivity analysis, but they refer to standardized coefficients which omit the scaling issues. If the input parameters were not standardized beforehand, it can be done according to Equation 4.11 [from 13]. Here, β_i is the standardized coefficient of feature i and w_i is the original weight. The former is obtained by scaling the latter with the ratio between the standard deviation of the respective feature and the standard deviation of the model's outcome [13].

$$\beta_i = w_i \frac{\sqrt{\sigma_i^2}}{\sqrt{\hat{\sigma}^2}} \tag{4.11}$$

The fact that the coefficients of the linear model are now all expressed on the same unitless scale makes them comparable. Moreover, they actually become natural measures for the model's sensitivity to the corresponding features [13]. That is, the higher the absolute value of a standardized regression weight, the more the feature affects the response. Lastly, the sign of β_i indicates the direction of the relationship. If it is negative, increasing the value of the parameter results in a decreasing outcome and vice versa.

Analyzing standardized regression coefficients has the benefit of being well accepted by the research community. That is mainly because its theoretical background is sound, allowing for extensive statistical analysis. Thanks to this, the model-specific approach is mature, has a proven track record and comes with a lot of prior experience [13, 70]. Moreover, the sensitivity measures can be readily obtained, without requiring many further calculations [13, 70]. A last advantage is the method's transparency and accessibility. One can easily understand how its results are attained and the linear architecture enables decomposition, which enhances the overall interpretability [9, 70]. However, a challenge remains the general mediocre performance of linear models. Their simplicity often prevents them from capturing complex relations, or requires the user to define non-linearities explicitly, as for example with the selected second-order polynomial [70]. The response of the model is by all means bound to its format, which in many cases is the prime reason for its poor predictive power. It makes sense that the surrogate must be sufficiently accurate, otherwise the trustworthiness of the model-specific explanations could be questioned [13]. Finally, one should be careful with multicollinearity. If there would any presence of correlation among features, then the standardized regression weights might not reflect their true sensitivity [70]. This could lead to fallacious conclusions, which is of course undesirable.

4.2.2. RuleFit

Tree-based machine learning algorithms are among the most popular alternatives, mainly because of their overall impressive performance [34]. However, this comes at the expense of interpretability. They are usually so complex that users can no longer understand their reasoning. Therefore, Friedman and Popescu [31] suggested to extract the rules from the ensembles and then to determine which are the most dominant. While the details behind regression trees and their ensembles were expounded in subsection 3.3.5, the general principle of how decision rules can be abstracted is depicted in Figure 4.4 [70]. The tree in the example consists of four rules, as indicated by the red arrows. Note that they do not necessarily have to end up in the terminal leaves. Earlier decisions are also considered, as is the case with r_1. This can be done for all trees in an ensemble, leading to a myriad of rules. To determine which ones are the most important and what their impact is, RuleFit uses regularized linear regression with both the decision rules and the direct effects of the input parameters [70]. The parameter value of a decision rule equals one if the condition is met, and zero other-

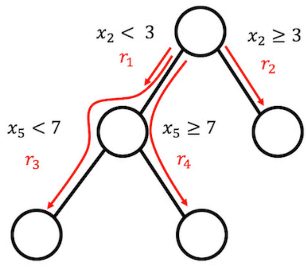


Figure 4.4: Abstracting decision rules from regression trees [70].

wise. The further interpretation of the regression is comparable to ordinary linear models, since it concerns the analysis of the coefficients. Notwithstanding, it resolves one of the most impeding limitations of linear regression. By including decision rules, the algorithm can naturally capture interactions between the parameters, which would otherwise have been much more difficult [70]. Going into further detail, the approach requires two stages. First the rules must be generated and abstracted, to then determine which ones are the most important. We now elaborate on the two steps in respective order.

Since RuleFit remains model-specific, decision rules are directly retrieved from a fitted surrogate. The two selected tree ensembles — random forests and gradient boosting — qualify, so one can adopt this approach to interpret them both. Essentially, the first stage omits the outcome of the machine learning model and aggregates solely its rules. These can be formally expressed according to Equation 4.12 [from 31, 70].

$$r_i(\boldsymbol{x}) = \prod_{j \in S_i} I\left(x_j \in s_{ji}\right) \tag{4.12}$$

Here, one of the possible rules r of a tree i is as follows. For each input parameter j that is part of the considered set of parameters S in the tree, the indicator function I is equal to one if the value of parameter j is in the subset s as defined by the corresponding split of the tree [70]. If all the indicator functions' conditions hold for a particular data point $\boldsymbol{x}$, then the rule yields one, which is the mathematical equivalent of a true statement. This seems abstract, but an example will show that it is actually rather straightforward in practice. Revisiting Figure 4.4, the formal expression of the fourth rule would be $I(x_2 < 3) \cdot I(x_5 \geq 7)$. The rule is true if both conditions are. From the moment either one is not, its indicator function becomes zero, which immediately affects the entire rule. Conditions are thus connected by means of logical conjunction. Finally, an important consideration is the depth of the individual trees, as it directly determines the maximum number of conditions in a rule [31, 70]. Recall from subsection 3.3.5 that this was one of the hyperparameters. Too great a depth impairs the interpretability, so one should keep the trees shallow. Molnar [70] recommends a number no greater than three.

The first stage results in a large number of decision rules, where one may or may not be more important than the other [70]. Together with the input parameters, they form the input for a regularized linear regression model, which can identify the most dominant factors. Subsequently, a regression model is created to express the features and the decision rules by means of a weighted sum. This is in a similar fashion as Equation 3.8 from subsection 3.3.1. However, for the present purpose it would be problematic to calculate the coefficients using OLS. The reason is that there are simply too many input dimensions to the linear model, which does not promote the interpretability. Hence, some degree of sparsity would be highly desirable, making LASSO[2] a better alternative [70]. Applied to RuleFit, the regression coefficients for the features $\boldsymbol{w}_f$ and for the decision rules $\boldsymbol{w}_r$ are determined according to Equation 4.13 [from 31, 70].

$$\left\{\boldsymbol{w}_f, \boldsymbol{w}_r\right\} = \underset{\boldsymbol{w}_f, \boldsymbol{w}_r}{\arg\min} \left[\sum_{i=1}^{m} \mathscr{L}\left(f\left(\boldsymbol{x}_i\right), \hat{f}\left(\boldsymbol{x}_i\right)\right) + \lambda \cdot \left(\sum_{i=1}^{n} \left|w_{f_i}\right| + \sum_{i=1}^{k} \left|w_{r_i}\right|\right)\right] \tag{4.13}$$

The cost function consists of two parts. The left term contains a loss function $\mathscr{L}$ which measures the discrepancy between the outcome of the regression model $\hat{f}$ and the actual outcome of AATOM f for all data points $\boldsymbol{x}_i$ in the sample with size m [70]. Logically, one wants to reduce this model error metric as much as possible. But not at all costs, because the aim is also to achieve sufficiently sparse coefficients. This interest is defended by the second term on the right. It is what mathematicians refer to as the L1-norm, which is essentially the weights' sum of the absolute values [38]. In the expression, the number of input parameters is again denoted by n and k refers to the number of decisions rules obtained from the first stage. Since the coefficients themselves are included in the cost function, many will be forced to be zero and that is exactly the purpose [70]. The extent to which this happens can be controlled by the regularization hyperparameter λ. The higher it is, the more the weights are penalized. Ultimately, the cost function leads to sparse coefficients that determine the linear regression model. These can be further analyzed in a next step to draw conclusions about the relevancy of the features and the decision rules [70].

The analysis of regression coefficients is actually quite similar to what we explained in subsection 4.2.1; one should just keep the two different kinds of parameters in mind. The weights $\boldsymbol{w}_f$ of the features can be interpreted as before. That is, one unit increase in its value, ceteris paribus, adds the respective weight to the response and vice versa [70]. This does not apply to the weights $\boldsymbol{w}_r$ of the decision rules because the multiplication of the indicator functions leads to a binary result. A rule is either true or false, and its value

[2]LASSO is the abbreviation for least absolute shrinkage and selection operator.

can therefore not shift by one unit. Consequently, these coefficients must be interpreted as the effect on the prediction when a rule is true [70]. However, there is more than only the coefficients. The fact that decision rules are part of the algorithm makes it possible to calculate their support. Essentially, that is a metric which indicates the relative number of data points abiding the rule [31, 70]. For example, if the support of the first rule r_1 in Figure 4.4 would be 80%, it means that 80% of the instances in the sample have an x_2 lower than three. The final by-product of RuleFit is that features importances can also be obtained. Nonetheless, the calculation is somewhat more complicated than for earlier interpretation methods. The reason is the combination between direct influences and decision rules, where some features can even be present in multiple rules [70]. The absolute value of the regression coefficients form the basis, but they must first be scaled by their standard deviation to make them comparable. In essence, the importances where a particular feature is involved are added to one another, although the contribution of the decision rules part is averaged, because otherwise the direct influence would be eroded [31, 70]. That said, the interpretation remains similar to before. The higher the importance, the more the model relies on the feature in making its predictions [70].

RuleFit is promising because it combines the strengths of linear models and tree ensembles all in one [31, 70]. On the one hand, the inclusion of decisions rules allows the model to capture complex relationships, and to naturally consider interaction phenomena between the input variables. On the other hand, the end product remains a transparent linear regression model. This makes it a powerful approach without compromising interpretability. Moreover, it entails some flexibility, since one can for example change the depth of the regression trees and tune the regularization hyperparameter [70]. The user can thus optimize the model according to the necessities. Lastly, RuleFit yields an actual model rather than just some explanations. This provides the opportunity to perform additional analyses, which could enhance the interpretability even further [70]. Notwithstanding, there are also some drawbacks and weaknesses. One of the main issues is that some of the rules might be overlapping, which deteriorates the explanatory power. For example, the model can consider $x_2 < 3$ and $x_2 < 7$ as two separate decision rules. This is rather difficult to interpret, because the second rule will always be true if the first is. Hence, one cannot always read regression weights as the isolated effect of a rule on the response, as others may be interfering [70]. Furthermore, RuleFit remains a linear model. Despite the consideration of decision rules, there is no guarantee of comparable accuracy to the tree ensembles. In the end, the architecture is no more than a weighted sum [70]. Another issue is that a large number of rules may be necessary, making the architectural reasoning more difficult to explain [70]. The regularization hyperparameter must therefore balance the prediction error against the complexity of the model. Finally, while RuleFit is model-specific because it directly analyzes the decision rules from a tree ensemble, an intermediate step is required for the actual interpretation. This is an inherent weakness of the approach, yet of key importance to provide any insight into the behavior of model.

5 Research Proposal

As a natural consequence of the literature survey on the state-of-the-art, knowledge gaps have arisen that we aim to fill. Therefore, this chapter presents the proposal for the intended research. First, section 5.1 lists the objective and research questions, along with their motivation and novelty. Subsequently, the relevance of the study is argued in section 5.2. This mainly concerns which stakeholders benefit from which specific contributions to the academic knowledge. Lastly, the intentions are translated into practice by section 5.3, which divides the work into concrete packages and covers the further planning of the project.

5.1. Objective and Research Questions

Reviewing existing literature on airport terminal modeling, surrogate modeling and interpretable machine learning has uncovered a number of important scientific lacunae. In particular, there is a need for highly accurate but comprehensible models that are capable of explaining the complex dynamics in sociotechnical environments, such as airport terminals. This is not a straightforward exercise, as these systems are characterized by emergent properties resulting from natural human behavior. Nonetheless, agent-based simulation models can approximate the real-life situations with high fidelity, although they suffer from substantial computational requirements. Hence the interest in surrogates, which are deemed promising alternatives at much lower intensity. Altogether, this leads to the objective of our research, which can be described as:

> *The research aims to accurately abstract and explain the dynamics of complex airport terminal operations by means of creating high-performing and interpretable surrogate models based on a detailed and validated agent-based terminal simulation model.*

Achieving the objective requires a multidisciplinary approach. More specifically, the boundaries of current academic knowledge are being pushed along three dimensions. Namely, the ambition to sample adaptively from AATOM, the accuracy improvement of the state-of-the-art surrogate models, and the further analysis of the meta-models for understanding the system. Sampling is the first step, where the strategy is determined by the design of experiments. It makes sense that choosing the right parameters and then the right data points in the associated parameter space is essential to arrive at a sufficiently informative sample — one cannot expect excellent performance from a surrogate model if its training data is of poor quality. When the sample is considered adequate, the second step is to get the most out of it. This includes the architectural choices and optimization of the meta-models, as well as how to measure their performance in a fair way. Finally, the surrogates also form the basis for obtaining a detailed understanding of airport terminal operations. Important considerations are their transparency and interpretability, although opaque models with an inherent back box character can be analyzed as well using model-agnostic techniques. These three dimensions can be summarized in the research question, which aims to fill the identified gaps in an effort to accomplish the objective. It forms the main backbone of our intended research and reads as follows:

> *How can machine learning algorithms simultaneously achieve high accuracy and high explainability in the meta-modeling process of a computationally expensive agent-based simulation model for airport terminal operations?*

In order to steer the thesis in the right direction and to remove ambiguities, the research question has been divided into clear sub-questions according to the three dimensions:

1. How and to what extent is the selected adaptive sampling strategy capable of extracting high-quality information from AATOM?
   (a) Which input and output variables are most appropriate to use in the surrogate modeling process?
   (b) What is the required computational budget to consider the sample as sufficiently informative, i.e. when and how is the stopping criterion reached?
   (c) How does the sample look like? Are the collected data points well distributed across the entire parameter space or do they tend to cluster around certain locations?
   (d) What is the optimal balance between exploration and exploitation of the domain, and does this change during the sampling process?
2. What is the overall performance of the surrogate models in terms of accuracy compared to the true output of AATOM?
   (a) What are the most appropriate key performance indicators to measure the accuracy and are there considerable differences between them? If so, how can the differences be explained?
   (b) How does the performance of the selected surrogate models compare? Which are the most and least accurate?
   (c) How large are the differences in achieved meta-model accuracy for different responses of AATOM?
   (d) To what extent is the accuracy of the surrogate models affected by the optimization of their hyper-parameters?
   (e) To what extent are the emergent properties of the airport terminal system preserved in the surrogate models?
3. How can the dynamics behind airport terminal operations be explained based on the surrogate models?
   (a) Which of the model-agnostic approaches are most successful in extracting and visualizing the relevant rules, variables and characteristics from the surrogate models?
   (b) Which of the model-specific approaches are most successful in extracting and visualizing the relevant rules, variables and characteristics from the surrogate models?
   (c) What are the main differences between the insights from model-agnostic and model-specific approaches and how can they be explained?
   (d) Which rules, variables and characteristics of airport terminal operations are most relevant to understand the system dynamics and make it explainable?
   (e) What are the main strengths and weaknesses of currently available methods and what would be promising directions for future research in the field of interpretable machine learning for meta-modeling purposes in particular?

To conclude this section, we briefly emphasize the novelty of the intended research as compared to the existing academic knowledge. First and foremost, active learning and meta-modeling are definitely not new concepts in engineering. However, their application to airport terminal operations remains rather unexplored, especially when combined with AATOM. It is therefore important to continue the work of De Leeuw [20], as there are still many unknowns in what could revolutionize managerial and engineering approaches in the domain. Secondly, this also holds for the application of interpretation methods to the surrogates, despite the fact that the associated academic knowledge is still in its infancy. Research on explainable artificial intelligence is flourishing and promising directions are being identified. We contribute by juxtaposing the state-of-the-art approaches on actual case studies, and strive for synergies with surrogate modeling in particular. So in that regard, the novelty lies not only in their application to AATOM's meta-models, but also in their mutual comparison when being applied in practice. This makes it possible to indicate promising directions for further research into the usage of meta-modeling to explain the complex dynamics of sociotechnical systems. A final note is that highly detailed agent-based models remain essential to develop decent emulators. Therefore, the aim of our work is by no means to replace one with the other, but rather to exploit the common ground to achieve high accuracy at low computational costs for modeling airport terminal operations.

5.2. Relevance of the Project

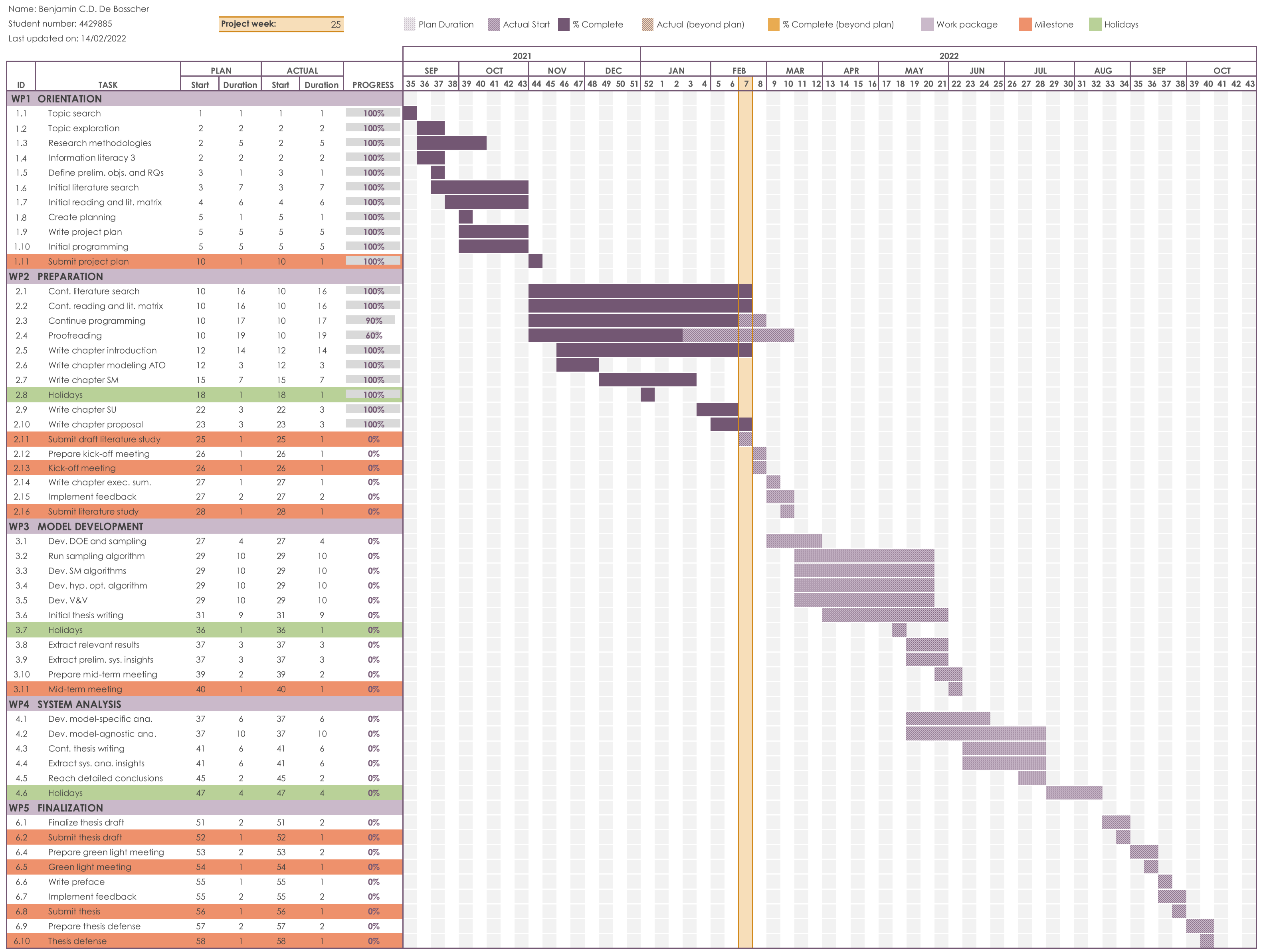
At first, one may wonder for whom it would be useful to create meta-models of another model that are known in advance to be less accurate. In line with the research areas as mentioned in section 3.1, there are two important reasons: to enhance the understanding of airport terminal processes and to support process optimization and design tasks. The main purpose of the former is to thoroughly understand the system and its emergent properties. So it is about the dynamics, key drivers, rules and variable relationships in regard to the operational settings of the processes in passenger terminals. Such information is particularly relevant to airport and airline managers, as they have to make key decisions about the terminal operations based on a particular flight schedule. Here is also the common ground with the latter reason: if the surrogate models prove to be sufficiently accurate, they can even be used directly in the decision-making. That is only possible because they are much faster and less susceptible to response stochasticity in contrast to AATOM itself. This could improve the way managers make their day-to-day decisions — the meta-models can ensure more agile and lean terminal processes by aligning the operational settings exactly with the expectations from the flight schedule. In other words, they would enable managers to better meet user demands in the future. Doing as such with the agent-based simulation model would require a lot of computational efforts, which may not be feasible from the practical perspective. However, apart from the operational side, the usefulness of surrogates can also be extended to the engineering field. For example, if a new terminal is being built, the layout is designed based on prospected traffic patterns and passenger flows. It makes sense that fast models of the terminal are greatly welcome to find the most optimal design in an acceptable time frame, especially since these tasks are often characterized by large parameter spaces. After all, more efficient calculation methods are never a superfluous luxury.

Synthesizing the two mentioned reasons, our research leads to the following practical benefits. The first and most obvious is that surrogate models are much more efficient and faster than AATOM. Even though they are less accurate than the agent-based simulation model, their savings in computational expenses far outweigh the drawbacks. Secondly, a direct consequence is that faster models allow for a more extensive analysis of underlying systems and the usage of the models in optimization and design tasks. Current thesis focuses in particular on the former, where these insights are expected to reveal promising extensions to the research on the common ground of surrogate modeling and interpretable machine learning. This is especially relevant at the moment since the academic knowledge of explainable artificial intelligence is advancing rapidly. Finally, it results in powerful tools for rapid but robust decision-making. With such tools, airports can ultimately operate more efficiently by better deploying resources such as personnel and services in relation to the flight schedule.

5.3. Planning of the Project

Now that the research opportunities have been defined in detail, the ambitions are translated into practice by drawing up an extensive planning. First, the work is split into different packages, each consisting of several specific tasks. Then it is estimated how long it takes for them to be completed, after which they are added to a Gantt chart. This is done so that they are time-bound, making dependencies and parallel tasks clearly visible. Lastly, important milestones and holidays are also included to get a full picture of the project's timeline.

The Gantt chart is depicted in Figure 5.1. Overall, the project is divided in five work packages, henceforth abbreviated as WP. The first two are already completed by the time of writing. Respectively, WP1 took about two months and was mainly concerned with the orientation phase of the thesis. It started with the search for a topic and research methodologies, while it ended with the submission of the project plan before the deadline of November 5, 2021. After that, the preparatory work was done in WP2, which took about 4 months in total. The key deliverable was the current literature survey, immediately followed by the kick-off meeting on February 25, 2022. Collecting, reading and processing literature were therefore the core tasks of the package, although the foundation for the actual research has also been laid, such as preparing the selected version of AATOM for the sampling process. Next, the third WP contains the tasks for the surrogate model development. This includes sampling, programming the algorithms, hyperparameter tuning, verification and validation, extraction of results, preliminary model interpretation and initial writing of the thesis paper. WP3 is estimated to last approximately 3 months and ends with the mid-term meeting in early June 2022, where the first results will be presented. After developing the models, they are subject to further analysis to explain and interpret the reasoning behind their behavior. This is carried out in WP4, which will take about 2 months. The main tasks are to design and perform model-specific and model-agnostic analyses. Finally, WP6 is concerned with finalizing the thesis. The planning reserved another 2 months for this last package.



Name: Benjamin C.D. De Bosscher
Student number: 4429885
Last updated on: 14/02/2022

Project week: 25

ID	TASK	PLAN Start	PLAN Duration	ACTUAL Start	ACTUAL Duration	PROGRESS
WP1	**ORIENTATION**					
1.1	Topic search	1	1	1	1	100%
1.2	Topic exploration	2	2	2	2	100%
1.3	Research methodologies	2	5	2	5	100%
1.4	Information literacy 3	2	2	2	2	100%
1.5	Define prelim. objs. and RQs	3	1	3	1	100%
1.6	Initial literature search	3	7	3	7	100%
1.7	Initial reading and lit. matrix	4	6	4	6	100%
1.8	Create planning	5	1	5	1	100%
1.9	Write project plan	5	5	5	5	100%
1.10	Initial programming	5	5	5	5	100%
1.11	Submit project plan	10	1	10	1	100%
WP2	**PREPARATION**					
2.1	Cont. literature search	10	16	10	16	100%
2.2	Cont. reading and lit. matrix	10	16	10	16	100%
2.3	Continue programming	10	17	10	17	90%
2.4	Proofreading	10	19	10	19	60%
2.5	Write chapter introduction	12	14	12	14	100%
2.6	Write chapter modeling ATO	12	3	12	3	100%
2.7	Write chapter SM	15	7	15	7	100%
2.8	Holidays	18	1	18	1	100%
2.9	Write chapter SU	22	3	22	3	100%
2.10	Write chapter proposal	23	3	23	3	100%
2.11	Submit draft literature study	25	1	25	1	0%
2.12	Prepare kick-off meeting	26	1	26	1	0%
2.13	Kick-off meeting	26	1	26	1	0%
2.14	Write chapter exec. sum.	27	1	27	1	0%
2.15	Implement feedback	27	2	27	2	0%
2.16	Submit literature study	28	1	28	1	0%
WP3	**MODEL DEVELOPMENT**					
3.1	Dev. DOE and sampling	27	4	27	4	0%
3.2	Run sampling algorithm	29	10	29	10	0%
3.3	Dev. SM algorithms	29	10	29	10	0%
3.4	Dev. hyp. opt. algorithm	29	10	29	10	0%
3.5	Dev. V&V	29	10	29	10	0%
3.6	Initial thesis writing	31	9	31	9	0%
3.7	Holidays	36	1	36	1	0%
3.8	Extract relevant results	37	3	37	3	0%
3.9	Extract prelim. sys. insights	37	3	37	3	0%
3.10	Prepare mid-term meeting	39	2	39	2	0%
3.11	Mid-term meeting	40	1	40	1	0%
WP4	**SYSTEM ANALYSIS**					
4.1	Dev. model-specific ana.	37	6	37	6	0%
4.2	Dev. model-agnostic ana.	37	10	37	10	0%
4.3	Cont. thesis writing	41	6	41	6	0%
4.4	Extract sys. ana. insights	41	6	41	6	0%
4.5	Reach detailed conclusions	45	2	45	2	0%
4.6	Holidays	47	4	47	4	0%
WP5	**FINALIZATION**					
6.1	Finalize thesis draft	51	2	51	2	0%
6.2	Submit thesis draft	52	1	52	1	0%
6.4	Prepare green light meeting	53	2	53	2	0%
6.5	Green light meeting	54	1	54	1	0%
6.6	Write preface	55	1	55	1	0%
6.7	Implement feedback	55	2	55	2	0%
6.8	Submit thesis	56	1	56	1	0%
6.9	Prepare thesis defense	57	2	57	2	0%
6.10	Thesis defense	58	1	58	1	0%

Figure 5.1: Planning of the entire thesis project, extracted in week 7 of 2022.

This is likely an overestimation, but it was done on purpose to incorporate for uncertainties that may arise throughout the project. WP6 consists of several milestones and important deadlines. Chronologically, that is a review of the draft paper, the green light meeting, the submission of the thesis and finally the defense in October 2022. As always, actual progress deviates from the planning, so the Gantt chart will be revised once again at the remaining milestone meetings, being the mid-term and green light meeting.

In summary, the project was scoped in WP1, after which the existing knowledge around the topic has been synthesized in WP2. Subsequently, realizing WP3 and WP4 provides the answers to the research question, in order to achieve the objective. Ultimately, WP5 completes the project and finalizes the last milestones and deliverables. There are two closing remarks. First, the time for the literature study and thesis itself — WP2 to WP5 — adds up to about 11 months in total. While this may seem against the standard of 9 months, holidays are also included along with a buffer against unforeseen circumstances. Delays are of course undesirable and are therefore anticipated early. Second, the work packages are generally planned in a sequential order. That makes sense because they usually depend on the previous one. Notwithstanding, there are some preparatory tasks which can be completed earlier. The Gantt chart shows them at the time they should be carried out. Task 1.10 is an example; it covers the initial programming steps in WP1 to set up the interface between AATOM and Python. In principle, the task is thus not related to writing the project plan, despite it appearing so.

III

Supporting Work

A
Preparing AATOM for Surrogate Modeling

The starting point of our research was the presumption that AATOM has been verified, validated, and calibrated. This is confirmed by the creators in Janssen et al. [51] for the original version, although the question arises whether this also holds for the later developments of the model. Despite the validation and calibration efforts mentioned by Mekic et al. [68], they remained rather unclear about the extent to which extensions have been verified. Since surrogates can only be as accurate as the training data they are fitted upon, meta-modeling requires access to models that are virtually error-free and produce data of high quality. The current appendix investigates whether that is indeed the case for the latest version of AATOM. More specifically, section A.1 describes the test procedure along with an overview of the identified issues, after which section A.2 continues with the implemented solutions.

A.1. Testing the Model

Testing the source code of software for programming bugs is usually done in multiple steps. Traditionally, it starts with unit testing individual components and ends with integration testing entire systems, and their interconnections [35]. While such an approach leads to rigorous verification of AATOM, it is also time consuming and rather complex to execute properly afterwards, independent of the code development. For that reason, a more practical approach is preferred, for example on an explanatory basis. An advantage of AATOM is that it comes with a graphical user interface (GUI) which shows exactly what is happening inside the airport terminal, along with a live tracking of the output parameters. Not only is this useful for getting acquainted with the simulation model in general, it also proves to be an excellent tool for debugging and verification purposes. Therefore, to test the model's robustness and see whether it behaves as expected, several simulations are performed with different input settings while manually observing the GUI. This includes examining both different traffic scenarios as well as different airport strategies, like the check-in counter staffing. The downside of such an approach is that it is rather infeasible to automate the identification of software issues. Instead, one has to constantly monitor the user interface, which naturally takes a lot of time. Moreover, another risk is that there might be bugs that are visually difficult to detect. Notwithstanding, despite these drawbacks, the explanatory approach is deemed more feasible and effective compared to the traditional one in preparing AATOM for being meta-modeled.

The test process started with examining the layout of RTHA that was already available in AATOM, as depicted in Figure A.1[1]. Two elements readily attract attention. On the one hand, the security checkpoint queue is largely exaggerated compared to reality, and on the other hand, there is a fictitious wall just to the left of the right-hand entrance. Both may seem strange at first, but the explanations are actually rather straightforward. That is, the former has been extended in order to accommodate all passengers in line. If this were not the case, and more agents want to enter the queue than its capacity allows, it would eventually lead to a total obstruction of the security checkpoint. Agents will block the queue entry for the remainder of the simulation due to undesired emergent behavior that occurs as a result of the interaction between the social forces of Helbing et al. [40] and the way the queuing activity is defined in the model. Consequently, extending the queue is a straightforward modification to avoid the erroneous phenomenon without endangering the overall validity. Ideally, one should improve the source code, but to respect the scope of the current research,

[1]The various elements on the map have been previously described by Figure 2.6 of the Literature Study in Part II.

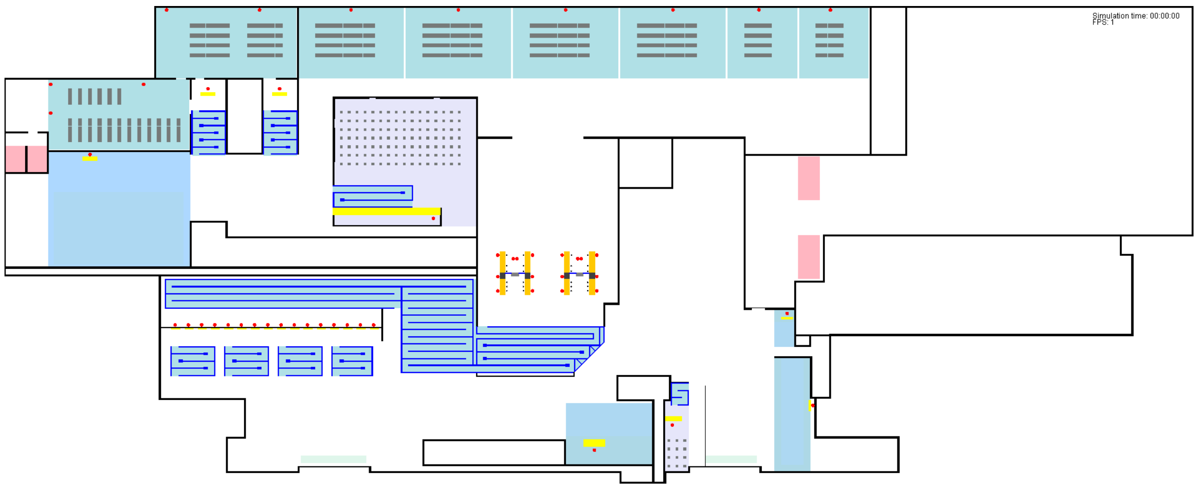
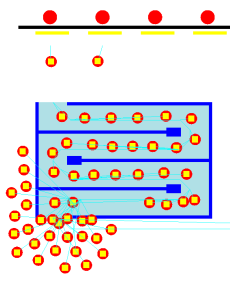
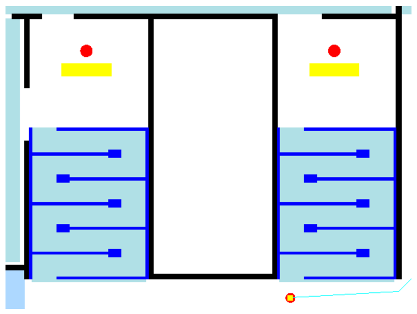


Figure A.1: Modeled layout of RTHA that was originally available in the latest AATOM version.

we held on to the rather pragmatic but effective solution. Secondly, the fictitious wall at the right-hand entrance is to adjust the agents' navigational behavior. As AATOM uses the pathfinding algorithms by Cui and Shi [18] and Harabor and Grastien [36], passengers are inclined to take the shortest walking route towards their destination [48]. The consequence of such behavior is that they can pass just next to the entrance of another queue, which is somewhere on their way. While this is not necessarily an issue, it sometimes leads to them being pushed inside if there happens to be some other passenger about to enter that particular queue. This is also an undesired effect of the social forces between the agents. The resulting problem is that most of the time, the two passengers cannot resolve the conflict on their own, with one wanting to leave and the other wanting to stay in line. This again leads to the total obstruction of the queue for the remainder of the simulation, which would falsify the output of AATOM because these conflicting situations do not occur in reality. In principle, the ideal solution would be to use an updated or even a different pathfinding algorithm, though a pragmatic solution is also possible by defining fictitious walls. These are walls that do not exist in practice, but they trick the navigation module in finding a path that avoids the entrance of a queue on the way to an agent's destination. The main advantage of this approach is that it easily resolves the issue. Going back to Figure A.1, agents entering the terminal are forced to remain right of the fictitious wall, and thus will not interfere with the queue for the establishment left of the right-hand entrance. However, the solution is sub-optimal because it does not solve the root cause of the problem, and on top of that, it influences the navigational behavior of the agents in an unnatural way. Even though passengers do not take the shortest path in reality either, forcing them to go around certain areas may affect the emergent properties during a simulation. Notwithstanding, since the new walking routes are very similar to the previous ones, the effect on the ultimate outcome is rather marginal. For that reason, we again adopt the pragmatic solution.

Now that it is clear why the layout of RTHA in AATOM differs slightly from reality, the actual testing procedure can be initiated. As mentioned, the model was consecutively given different input parameters while monitoring the GUI to identify possible errors. At the same time, the random seed was also varied to control the stochasticity, although logs were made so that particular scenarios in the airport terminal could be reproduced in case issues were found. Two kinds of problems soon came to light, yet surprisingly they had been discovered before. Namely, queues become obstructed if their capacity is exceeded and passengers are sometimes pushed into a queue where they do not want to be — these are the exact same issues for which Mekic et al. [68] already implemented pragmatic solutions in Figure A.1. It appears that other areas in the terminal are also affected by the unrealistic emergent behavior of agents. Examples from the GUI are shown in Figure A.2. On the left, Figure A.2a illustrates what happens when more passengers try to enter a check-in queue than there is available space. The agents typically block the entrance and are unable to resolve the situation themselves. One can see that they stopped entering the queue, despite a gap forming as the previous passengers still carry on. Needless to say, these congestions have a major impact on the outcome of simulations, so they must be eliminated before using AATOM for surrogate modeling. Interestingly, queues at the check-in counters are the most sensitive to the phenomenon: border control, and food and beverage outlets are generally unaffected. Moving on to the second problem, Figure A.2b and A.2c show a particular passenger that passes very close to a border control and a check-in queue, respectively. Their walking route is


(a) A congested check-in queue.


(b) An agent passing just next to a border control queue.

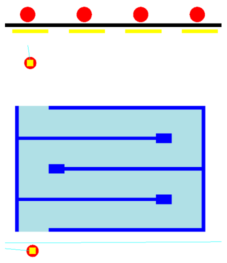

(c) An agent passing just next to a check-in queue.

Figure A.2: Frequently occurring issues in AATOM.

indicated with a light blue line, so it is clear they do not intend to go into any of the queues. Notwithstanding, if another passenger wants to enter that specific queue at the exact same time, the agent is pushed inside and a similar congestion arises as in Figure A.2a. Again, this is of course highly undesirable, as such situations are unlikely in practice and if they did occur, passengers would be able to resolve it themselves. Altogether, it seems that Mekic et al. [68] have only partially eliminated the queue blocking issues for the tested scenarios, so the following section A.2 will further explore the options to prevent this from happening.

In addition to the rather frequent queue obstructions, the explanatory tests revealed one other problem that requires a little more attention. Namely, it turned out that passengers suddenly stopped walking through one of the metal detectors at the security checkpoint. The occurrence must be very rare as it was only discovered once, although the consequences were serious for obvious reasons. It is depicted in Figure A.3 — an agent is clearly stuck at the right-hand metal detector, blocking the two respective lanes. As a result, only the left two lanes remain available, one of which was closed at the time due to airport strategy. This quickly led to a huge queue with increasing waiting times, causing many passengers to miss their flight. Despite being rare, several questions arose as to why this happened and how it could be resolved. The main concern was that it is caused by a bug in the source code of AATOM. And if this indeed turns out to be the case, does it affect other operations that may be less detectable in the GUI? While finding such a bug is most likely not straightforward, it is clear that no pragmatic solutions to the problem exist either. It does not involve the obstruction of queues, so elongations or fictitious walls will not help. Hence, a closer analysis of the issue along with an examination of the source code is necessary. It started by reviewing the general principles of the security checkpoint and the roles of its operators. The first hypothesis was that the bug is related to the walk-through metal detector. However, further analysis revealed that the detector had already made an observation, and that the operators were advised to conduct an additional test for detecting explosives traces. So, as a matter of fact, the passenger has actually passed the point where it appears to be stuck. This implies that the bug must be further along in the process. Furthermore, it was also found that it happened shortly after a particular flight had departed, of which several passengers were still at the security checkpoint. This causes AATOM to register them as having missed their flight. A direct consequence is that they want to destroy themselves: they want to be taken out of the simulation because their ultimate goal can no longer be reached. Knowing this, the real reason for the impediment turned out to be as follows. When an agent is destroyed while it is getting tested for explosive traces, the concerned operator is not informed. Therefore, the agent does not proceed with the next passenger that requires such a test, who then keeps waiting at the metal detector as it is neither allowed to collect its luggage. Ultimately, this leads to the total obstruction of the two particular lanes. In summary, closer analysis revealed that the issue from Figure A.3 only occurs under very specific circumstances. So it is

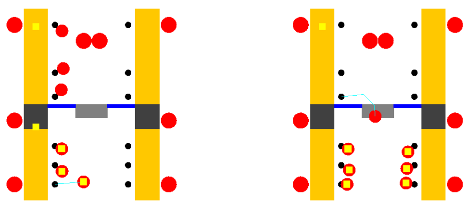


Figure A.3: An agent being stuck at the security checkpoint's walk-through metal detector.

indeed a rare software bug, although one should not forget the enormous impact on the operations when it does happen. For that reason, the following section will also consider a solution to this problem. Finally, a follow-up question arose as to whether the bug was also present at other places in the airport terminal. Since further testing and investigation of the source code did not uncover other cases, it seems to be a standalone issue in the explosive trace detection process.

In conclusion, the testing procedure identified three remaining problems in the latest version of AATOM: queues with insufficient capacity, agents being accidentally pushed into areas they do not want to be, and operators not being notified of removed passengers. Hence, finding suitable solutions to solve these three problems will be the main concern of section A.2. However, there is one final note before proceeding. Despite the explanatory tests have challenged the robustness of the model, it is unlikely that all of the bugs have been discovered. The most apparent ones have, but that is no guarantee for the less visible. Moreover, it is very difficult to eliminate all issues when the source code is as complex as AATOM's, especially with our limited research scope. Therefore, to build an additional layer of reliability around the model, outliers will be detected and removed from the sampled data points before being used further. In this way, unaccounted for irregularities are caught without the risk of impairing the data set. The removal of outliers is discussed in more detail in section B.2 of the Appendix B. Of course, the random seed is still being logged, so should a pattern arise caused by a specific bug during the sampling process, action can be taken accordingly.

A.2. Bug Fixes and Implemented Changes

The previous section identified three urging issues that must be resolved before AATOM can be used for surrogate modeling. We now take a closer look at which bug fixes are the most appropriate. However, for two of them this is actually rather evident. Recall that Mekic et al. [68] used pragmatic tricks to cleverly avoid the obstruction of queues. Regarding that, section A.1 has previously argued that adopting these solutions was the best option due to their simplicity and marginal effect on the outcome of the simulations. Moreover, it respects the scope of the current research, as revising the social forces model and the pathfinding algorithm would be a study in itself. Hence, the most suitable way to proceed is to slightly modify the terminal of RTHA in the model, which led to the new layout in Figure A.4. The largest change is at the check-in counters, where queues were expanded from 4 to 11 rows. To realize this, the outer walls of the terminal building in the lower left corner had to be altered first, although the effect on the simulation is negligible. While the extensions solved the capacity problems, agents were still being pushed inside the queues. That is why five fictitious walls were added in the corner: one on the right side of each queue and one just above the left entrance of the building. The latter is likely to have the biggest impact on the simulations, as it forces arriving agents to go left or right. Notwithstanding, it solves many issues because chances are minimized of them walking past queue entrances on the way to their goal. Especially the passengers who have already checked in beforehand are now more inclined to go right, directly to security. All in all, the effect of these seemingly large changes on the ultimate simulation responses is actually rather marginal. It is probably most noticeable by a slight increase in the total walking distance and therefore also the time it takes to reach the gate, although the latter will be

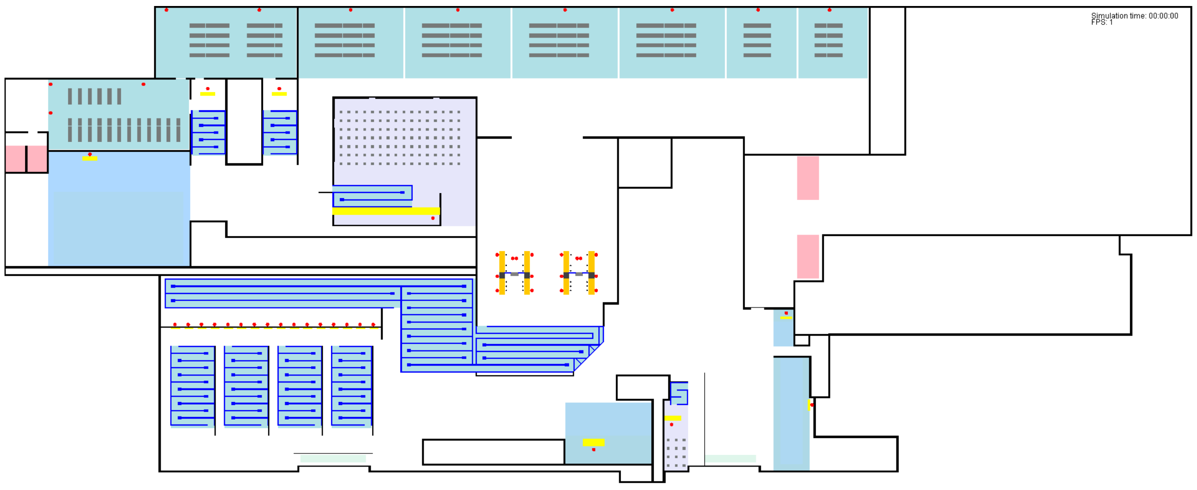


Figure A.4: Modified layout of RTHA that avoids the obstruction of queues.

much more affected by the waiting times in the queues. Furthermore, there were also some subtle changes. First of all, the queues at border control were sufficiently large, but the walls around have been extended into the open space below. Passenger leaving the store on the left to catch a flight at one of the Schengen gates (i.e., gates 1 to 6) tended to walk just past the queue entrances. This is prevented by the simple modification. Then, there is the main restaurant in the middle of the restricted area, where two fictitious walls were added. One at the entrance of the queue, and a more subtle elongation to the right of the operator agent serving customers. The former was necessary to give the arriving passengers a little more space from those who are leaving the restaurant from below to a particular Schengen gate. Otherwise, they may be pushed back into the restaurant. The latter, however, was more of a precaution to separate the incoming and outgoing flow. No issues were observed on the GUI, but the checkout area became rather busy as it was often along the shortest route for the exiting customers. Finally, the fictitious wall that was earlier discussed in section A.1 at the right-hand entrance of the terminal building is slightly extended compared to Figure A.1. The reason is that obstructions were still discovered, albeit rarely. That corner also turned out to be rather busy, with passengers entering the terminal building on the one hand, and others wanting to visit the coffee corner. The elongation created some additional space between the different flows of agents, decreasing the social pressure at the queue entrance. This prevents the entering passengers from being accidentally pushed inside. To conclude, fictitious walls and queue elongations solved many of the issues in the terminal, while their effects on the simulation responses are almost negligible. Hence, we therefore decided to adopt the modified layout in Figure A.4 for the current research.

The third issue — related to the explosive trace detection check — was less straightforward. No pragmatic solutions were available, so it was necessary to dive into the source code of AATOM. The bug was resolved by adding the following if-statement to the activity of the concerned operator agent.

```java
// This statement fixes a rare but impactful bug that when a passenger gets destroyed during
// the ETD check (e.g., because of missing the flight), it keeps blocking the security
// checkpoint because the operator is not informed.
if (passengerToCheck.isDestroyed()) {
   passengerToCheck = nextPassenger;
   nextPassenger = null;
   waited = false;
   goTo = false;
   if (passengerToCheck == null) {
      endActivity();
   }
   return;
}
```

In essence, it makes the operator slightly more intelligent. When it detects that the passenger to check is destroyed from the simulation, it will now continue with the next one. For that reason, the code inside the if-statement is very similar to how the operator normally completes its explosive trace detection check. The only difference is thus the condition on the first line. Adding the above snippet has no further deteriorating effects on the simulation. Therefore, it is the best possible solution to the identified software bug.

To conclude the current section, and actually the whole chapter, Table A.1 summarizes the encountered issues during robustness testing the latest AATOM version by Mekic et al. [68]. The most appropriate solutions are also included, along with their unwanted side effects. After making these changes to the RTHA terminal layout and fixing the bug in the source code of the software, one arrives at the ultimate version that is used in our research.

Table A.1: Summary of the identified issues in AATOM and their corresponding solutions.

Issue	**Solution**	**Side effects**
The queues at the check-in counters do not have sufficient capacity to accommodate all waiting passengers, leading to unrealistic obstructions.	Extending the queues naturally increases their capacity, so that passengers no longer have to wait outside.	The walking distance increases slightly and with it also the average queue time, although the effects are marginal compared to other factors, such as the personnel strategy or the number of passengers on flights.
Passengers can be accidentally pushed into arbitrary queues where they do not have to be. This can lead to unrealistic obstructions.	The addition of fictitious walls tricks the pathfinding algorithm to avoid the entrances of other queues en route to an agent's destination.	The pathfinding algorithm searches for the new shortest journey around the fictitious walls, slightly increasing the total walking distance and the average time it takes to reach the gate. However, the effects are marginal compared to the other factors, like the security checkpoint staffing strategy.
When a passenger's flight departs while it is being tested for explosive traces, the operator agent is not informed that the passenger is actually removed from the simulation, resulting in a total blockage of the respective lanes at the security checkpoint.	By improving the intelligence of the operator responsible for the explosive trace detection, it can observe whether the passenger to check has been destroyed from the simulation.	None, because the solution resolves the issue directly in the source code of AATOM.

B
Obtaining Stable Simulation Responses

Sociotechnical systems are characterized by a high degree of stochasticity, and so are their agent-based simulation models. This is a natural consequence when modeling human behavior, where cognitive, social, technical, and organizational factors play a major role. While the background of agent-based modeling is discussed in subsection 2.3.1 of the Literature Study in Part II, the current appendix examines to what extent AATOM's results are influenced by random events. This is commonly measured by the coefficient of variation, as elaborated upon in section B.1. However, aside from the natural variability, the responses can also be affected by particular flaws in AATOM. A lot of effort was put into Appendix A to solve the issues, although one can never be sure when using the model on a larger scale. It is therefore wise to add an extra layer of robustness by removing the outliers, which are usually a sign of faulty simulation runs. Nonetheless, this should be done with caution, as eliminating some of the correct outcomes can lead to a bias in the responses. The procedure is discussed in detail by section B.2.

B.1. Calculating the Coefficient of Variation

It is very likely that consecutive simulations in AATOM with the exact same input parameters will always lead to slightly different results. Namely, the stochasticity in the system makes every run unique, which is why the operations in the passenger terminal are almost never identical. This makes sense, because it is also the case in reality. Imagine the following scenario: by coincidence, two arbitrary days have the same flights, the same number of bookings, and the same airport strategies. The "input parameters" are thus identical. Nevertheless, it turns out that the average waiting time in the queue for the security check differs in the end. This is one example, but it also holds for the other responses. While there could be a myriad of possible causes, if just one passenger decides to arrive 10 minutes later, the emergent properties of the system are already changing. However, an airport terminal involves not one but hundreds of autonomous agents, all of whom make their own decisions. The situation can therefore be considerably different. This is clearly visible in the output data, as depicted by Figure B.1. The histograms show the distributions of three arbitrarily chosen responses for a random point in the feature space. They are all bell-shaped and mostly symmetrical, but there is indeed some variability. For example, the total expenditure in the left graph varies between €1,100 and

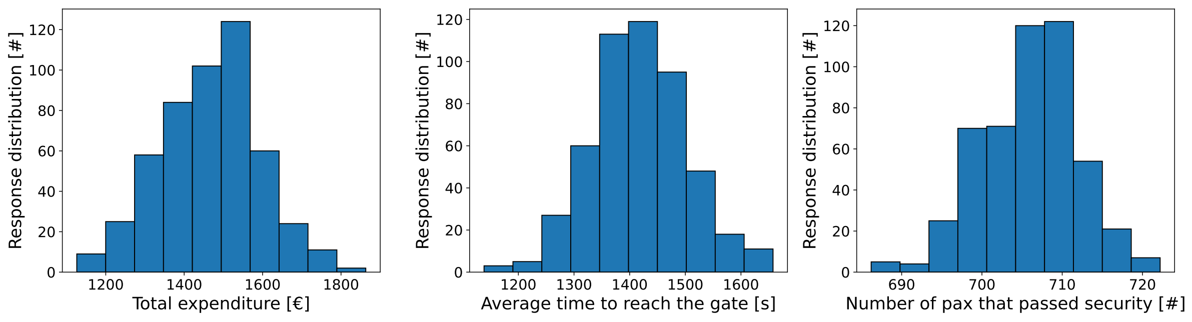


Figure B.1: The distributions of three responses when running simulations with the same input parameters.

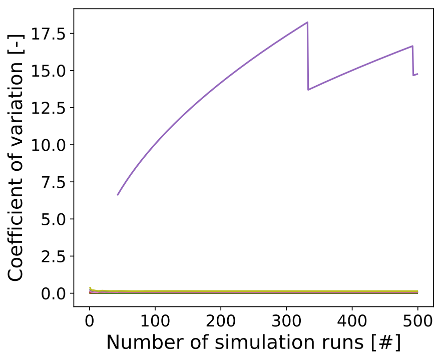
€1,900, which is quite a lot compared to the average of about €1,500. Consequently, the important question arises which of the simulation runs should be used in the ultimate data sample for surrogate modeling. Or is it better to perform multiple simulations with the same settings and take the average?

Unlike agent-based models, surrogates are deterministic in nature. They associate one input vector with one specific output vector, which means that it is difficult for them to consider stochasticity. As a result, there is no variation in the response of a black-box function if the same parameters are entered over and over again. This complicates the overall meta-modeling process, although it can be overcome rather easily by the law of large numbers [22]. In essence, the law states that if the simulation is run several times with the same settings in AATOM, the average response eventually approaches the value one would expect. Going back to the expenditure in the left graph of Figure B.1, the average of €1,500 is indeed a reasonable representation of the response from a deterministic point of view. So knowing that the effects of the stochasticity can be eliminated, an important follow-up question is how many simulations are required to consider the average result as stable. For this, one can resort to the coefficient of variation c_v — a common approach to describe the statistical dispersion of a particular outcome. It is calculated according to Equation B.1 [from 26]. The response's standard deviation and mean of consecutive simulation runs are indicated by s and $\bar{f}$, respectively.

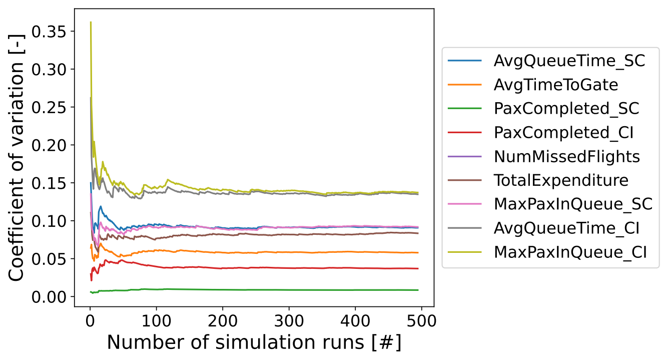
$$c_v = \frac{s}{\bar{f}} \tag{B.1}$$

The core idea is relatively simple: the lower the coefficient, the less a response is affected by sample instability. Note that one can even compare different output parameters, as the scale is omitted by the division [26].

The coefficient of variation becomes particularly interesting when plotted against an increasing sample size. Namely, it shows when the response stabilizes, allowing to determine the fewest number of simulation runs required to consider the average as the deterministic result. An example is shown in Figure B.2, which considers a data point with a moderate number of passengers in the terminal and a mediocre airport strategy[1]. The first attempt over 500 simulation runs resulted in Figure B.2a on the left. While the coefficient of variation is seemingly small for all responses, the curve for the number of missed flights in purple is rather problematic. Moreover, it is actually quite surprising that flights were missed at all — the input parameters called for a modest number of passengers, well below the breakdown limits of the airport terminal. This required further investigation, so the simulations of interest were rerun with their corresponding random seeds while examining the GUI in AATOM. It soon became apparent that not all problems had been resolved in Appendix A, as the bizarre results were caused by flaws in the simulation model. The occurrence of these phenomena is rare, although it has a considerable effect on the responses. Hence, it is crucial to somehow remove the erroneous simulation runs. A separate section B.2 is dedicated to this, because the issue was actually anticipated. After removing three faulty outliers, one arrives at the graph in Figure B.2b, which makes much more sense. Note that there is no longer a curve for the number of missed flights. The response's value was always zero, meaning the coefficient of variation is mathematically undefined. However, the curves for the other responses are precisely as expected. There is quite a bit of instability in the beginning, but it quickly diminishes over an increasing number of simulations. Overall, the check-in's average waiting time and maxi-



(a) Before removing the outliers.



(b) After removing the outliers.

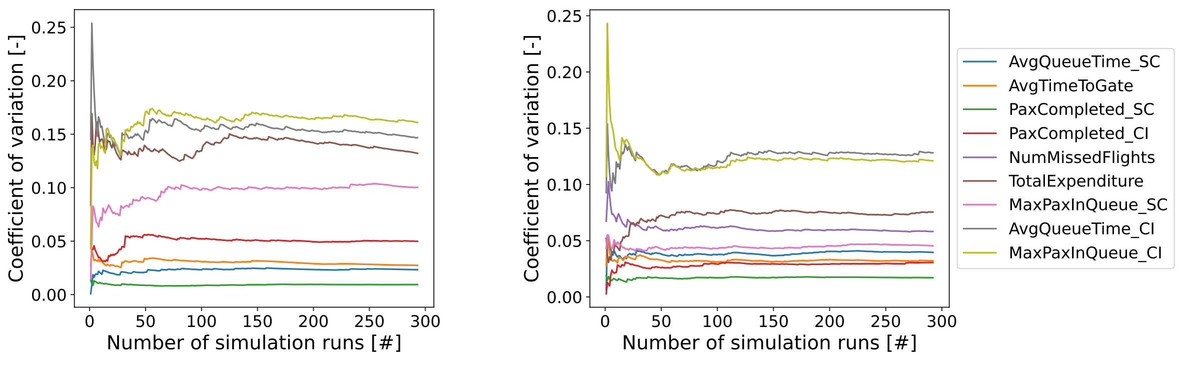
Figure B.2: The coefficient of variation against the number of simulation runs.

[1]The data point corresponds to a scenario where the airport opted for check-in strategy 5, security check strategy 2, a call-to-gate 30 minutes prior departure, and a load factor of 75% on all flights. See Appendix E for more information.

mum queue size are the most subject to variability, while throughput at security is the least. Notwithstanding, with the standard deviation of the former two stabilizing at about 15% of their mean, these results are definitely acceptable. Furthermore, the coefficients of variation appear to become stable from a sample size of approximately 175 runs for all responses. This suggests that there is no need to perform more simulations at a single data point, as the mean will not change much anymore.

While 175 repetitions may be sufficient for the previously chosen data point with a moderate number of passengers, it does not guarantee the same for other scenarios. That is, fewer or more passengers with different airport strategies could cause less or more instability. Therefore, two additional scenarios are also examined to see the effect of the crowd on the system's output parameters. The graphs are depicted in Figure B.3 for 300 simulation runs each. On the one hand, Figure B.3a, shows the results for a low-traffic scenario with excellent airport strategies, leading to very few passengers in the terminal[2]. This is in stark contrast to Figure B.3b, which considers a data point associated with extreme crowds. Such a situation can be elicited by setting the flights to fully booked in combination with poor airport strategies[3]. It is quite interesting to see that there are indeed some differences between the two scenarios. First of all, a higher number of passengers in the system seems to yield less variability in general. This may seem strange, although it actually makes sense as passengers are more restricted in their freedom. For example, if the waiting time at the security checkpoint equals one hour instead of 15 minutes, there is less time for non-aeronautical activities like shopping or dining. On the contrary, passengers have much more opportunities to enjoy themselves when their time spent in queues is almost negligible. This is just one example of how crowd size can influence the variability of the system's responses. Furthermore, note that there is a curve for the number of missed flights in the right graph, as opposed to the one on the left — the response even stabilizes rather quickly. Since no flights are missed with only a few passengers in the terminal, it again makes sense that the coefficient of variation for that case is undefined. Lastly, the effects of the crowd are also visible on the number of simulations required to obtain stable results. The low-traffic scenario appears to be consistent with the previous finding from Figure B.2b, while the the high-traffic scenario's output parameters are already stabilizing at about 125 runs. However, this is not so surprising, given the above explanation about the affected freedom of the agents. In conclusion, it can be said that the choice of input parameters does indeed influence the number of simulations required for stable results, albeit not excessively. The average response can be considered deterministic from at least 125 and 175 runs, respectively for a very quiet and a very busy airport terminal.

It is interesting to see that different scenarios in the airport terminal influence the variability of AATOM's output parameters, although in practice it is rather challenging to use this to our advantage. One cannot predict in advance whether a particular input setting will lead to many or few passengers in the system, except for some obvious cases as discussed in Figure B.3. A solution could be a more active approach by examining the coefficient of variation as data is being sampled. This seems promising in theory, but creates a lot of uncertainty because it is not known how many simulations will ultimately be needed. In fact, chances are high that the ideal number of runs will even be different per data point. Furthermore, hyperparameters must be



(a) Few passengers in the airport terminal.

(b) Many passengers in the airport terminal.

Figure B.3: The effect of the number of passengers in the system on the coefficient of variation.

[2]In this scenario, the airport opted for check-in strategy 2, security check strategy 4, a call-to-gate 60 minutes prior departure, and a load factor of 50% on all flights. See Appendix E for more information.

[3]In this scenario, the airport opted for check-in strategy 5, security check strategy 5, a call-to-gate 15 minutes prior departure, and a load factor of 100% on all flights. See Appendix E for more information.

chosen to determine when a response is considered stable, which is not so straightforward. For those reasons, along with some practical considerations, it is better to use a method that entails less uncertainty. Namely, the sampling process will be carried out on the online server of the Air Transport and Operations research group at the Delft University of Technology. This is necessary due to the computational intensity of AATOM; gathering sufficient data would otherwise be rather impossible. Looking at the implementation side of it, two interesting facts are that the supercomputer consists of 256 central processing units and sampling data points from the feature space is a sequential process. The latter means that one data point has to be sampled after the other: it is very difficult to consider multiple data points at the same time because of the active learning strategy. Therefore, to avoid wasting any of the available computational resources, we decided to run 256 simulations for each data point of the domain being sampled. The benefits of multiprocessing would otherwise be compromised if only a part of the available cores were used. This number is well above 175, which was previously set as the minimum for scenarios with the greatest variability. Nonetheless, additional runs never hurt, as it can only improve the stability even further — especially if the outliers are still to be removed. Hence, 256 simulations per data point is a solid choice, taking into account both technical requirements and practical possibilities.

B.2. Removing Erroneous Simulation Runs

In the previous section, Figure B.2a made it readily clear that Appendix A failed to resolve all issues in AATOM. Despite the efforts, however, this was expected in advance. The model is simply too complex to be flawless, especially when it is used for simulating airport terminals in their entirety. Removing the erroneous results is therefore crucial to ensure the overall integrity of the data. Namely, it increases the robustness against contamination, as was clearly the case in the first attempt to plot the coefficient of variation. The number of missed flights was most affected, so the response is further examined in Table B.1 by calculating the frequencies of the simulations' outcomes. Almost all runs end up being zero, which means that the irregularities in Figure B.2a are caused by solely three instances. It is remarkable that 0.6% of the 500 iterations has such an influence on the variability. However, they are also rather easy to detect, so it should not be too difficult to omit them from the ultimate data set. While the outliers were removed manually in the example, it must be automated for the overall sampling process. This is not only to create some consistency, but also because of the total amount of data that is expected to be gathered — it would be too much to do by hand.

Yet, detecting outliers is an area of research in its own right. It has already sparked the interest of many scholars, leading to countless possibilities. Among them are Hodge and Austin [41] and Rousseeuw and Hubert [84], who provide a comprehensive overview of the state-of-the-art. As the former concludes, it is important to understand that there is no one-size-fits-all. Choosing the right methodology strongly depends on the data and the ultimate goal. We have therefore not selected any of the options based on certain trade-off criteria, but considered the problem from a higher-level perspective. For example, instead of bluntly removing the odd-looking numbers, perhaps there may be some indicators that clearly show the presence of errors in a simulations run. With that in mind, we created a customized two-phased strategy as presented in Algorithm 1. The procedure combines multiple outlier detection methods and is designed to ensure maximum robustness: erroneous results should be removed, albeit without touching the correct ones. Otherwise, one risks to bias the data. The algorithm is now further explained in chronological order: first there are the logical indicators, and then the more generic detection methods.

Of course, the procedure can only be initiated when AATOM has finished sampling a particular data point. According to the previous section B.1, such a raw sample consists of 256 simulation runs, each yielding an output vector with the responses. In the first phase, the algorithm goes through all the runs while monitor-

Table B.1: Response distribution of the number of missed flights for moderate crowds in the terminal.

Number of missed flights [#]	**Absolute frequency** [#]	**Relative frequency** [%]
0	497	99.4
5	1	0.2
10	1	0.2
21	1	0.2
Total	500	100.0

Algorithm 1 Procedure for removing the outliers from a data point's sample.

```
\\Phase 1 – Logical indicators
for every simulation run in the data point's sample do
    if the maximum number of pax in one of the check-in queues $\geq 55$ then
        Remove the simulation run from the sample
    else if the maximum number of pax in the security check queue $\geq 345$ then
        Remove the simulation run from the sample
    else
        Keep the simulation run
    end if
end for
\\Phase 2 – Generic detection methods
for every simulation run in the data point's remaining sample do
    for all individual responses $f$ from the simulation run do
        $\mathrm{IQR} \leftarrow \mathrm{Q}_3 - \mathrm{Q}_1$                ▷ Calculation of the interquartile range
        if $\mathrm{IQR} = 0$ then
            zscore $\leftarrow \frac{f_i - \bar{f}}{s}$                ▷ Calculation of the standard score
            if $|$ zscore $| > 3$ then
                Remove the simulation run from the sample and continue with the next one
            else
                Keep the simulation run
            end if
        else
            if $f \in \left[\mathrm{Q}_1 - 2.5 \times \mathrm{IQR},\ \mathrm{Q}_3 + 2.5 \times \mathrm{IQR}\right]$ then
                Keep the simulation run
            else
                Remove the simulation run from the sample and continue with the next one
            end if
        end if
    end for
end for
```

ing five specific results. That is, it checks the maximum number of passengers who have been waiting at the check-in counters (four queues) and at the security checkpoint (one queue). These are the so-called logical indicators, as they reveal results with a high probability of error. Appendix A showed that the vast majority of AATOM's issues originate when passengers are queuing. Hence, it makes sense to examine the statistics of the queues most prone to unrealistic obstructions, namely those at check-in and security. When passengers can no longer be accommodated, either due to capacity constraints or an impediment, it often reflects in an unusually high number of agents inside. This even goes so far that there are more passengers than theoretically possible. Therefore, we know that a simulation is very likely to be faulty if one of the maxima is exceeded, which is 55 per check-in queue and 345 at security. The first phase of the algorithm takes advantage of this insight by proactively eliminating the concerning simulation runs, as can be seen up to line 10 in the pseudocode's if-statement. Note, however, that many of the impediments have been addressed by the implemented changes in Appendix A. Consequently, the limits are only occasionally exceeded, but since it still happens, the phase remains useful. Keeping track of the maximum queue sizes is thus a rather straightforward approach to detect outliers at an early stage.

While the first phase detects the evident cases, it fails to remove outliers that are harder to observe in the data. Therefore, we introduce a second phase with more generic detection methods. This is to ensure as much robustness as possible. Hodge and Austin [41] and Rousseeuw and Hubert [84] presented several options, so an important question arises as to which method is most suitable. The authors agree on the popularity of two methods: the usage of z-scores on the one hand, and box plots on the other. The advantage of both methods is that they are relatively straightforward and fairly robust, though they are limited to univariate instances [41, 84]. Notwithstanding, this should not be a problem as the responses are considered on an individual basis for simplicity. If one of them turns out to be erroneous, the entire simulation is removed from the sample. The fundamental difference between the two methods is that the standard score is based on the mean while box plots use the median. This makes the latter considerably less attracted to the outliers themselves, as is often the case for the former. Rousseeuw and Hubert [84] explain this phenomenon with a numerical example on page 74, which proves that z-scores can be affected in such a way that they become ineffective. For that reason, box plots are preferred over z-scores. Since the main principle behind box plots is rather trivial, it will not be discussed here, but we refer to section 16.4 of Dekking et al. [22, p. 236]. More interesting, however, is how they can be utilized to detect outliers. For that, one calculates the interquartile range (IQR), which is the difference between the upper (Q_3) and the lower quartile (Q1) [84]. Response values that do not fall within the interval $\left[Q_1 - 1.5 \times \text{IQR}, Q_3 + 1.5 \times \text{IQR}\right]$ are then typically referred to as outliers. Nevertheless, the factor 1.5 is a general rule of thumb and should preferably be adapted to the data set — more on that later [41]. Using box plots is a robust approach to detect and eliminate the erroneous simulation runs, although there is one problem. During the implementation, sometimes many instances were removed by the algorithm. This seemed rather strange at first, but closer analysis revealed that it is caused by one specific response. Namely, there are certain scenarios in the airport terminal with long queues, causing a handful of passengers occasionally to miss their flight. Nevertheless, in more than 50% of the simulation runs, everyone manages to catch their flight. In such cases, the IQR is zero, resulting in the removal of all instances where passengers did miss flights. This is of course highly undesirable, inasmuch as it distorts the data. A rather evident solution is to check the IQR first. Box plots can then only be used if it is not equal to zero, which is usually the case in practice. However if it is, another detection method should be chosen. While box plots were previously preferred over standard scores, this is actually the other way around when the IQR equals zero. Z-scores are not based on quartiles, but rather on a response's mean and standard deviation [84]. Therefore, they are not susceptible to the same problem as with box plots. The scores z_i are obtained according to Equation B.2, where $\bar{f}$ is the mean response, s the standard deviation, and f_i the response's value for a particular simulation run i [from 84].

$$z_i = \frac{f_i - \bar{f}}{s} \tag{B.2}$$

The standard scores are then calculated for all instances, and those whose absolute value exceeds a certain threshold are considered outliers. Rousseeuw and Hubert [84] mention 2.5 as a suitable candidate, although one has to tailor it to the requirements and the data. Moreover, if done right, it can even solve the aforementioned issue that z-scores may be affected by the outliers themselves — we will come back to this. Altogether, Algorithm 1 shows the second phase with the generic detection methods from line 11 up to the end. It goes through all the individual responses of the data point's remaining simulation runs and calculates the IQR first. In the rare cases it actually equals zero, the standard score is used to determine whether or not the instance is an outlier. Otherwise, the decision is based on the interval of the box plot. Note that if one of the responses is marked as an outlier, the entire simulation run will be discarded and not just the response in question. The algorithm thus operates univariately for the sake of simplicity, though the consequences are borne at a multivariate level. In the end, if one output parameter turns out to be an outlier, others most likely will be too, meaning the whole simulation is actually erroneous. Hence, this is not considered a problem. It actually makes the algorithm even more robust because responses are looked at on an individual basis, just as they will be meta-modeled. After eliminating all outliers in the second phase, the output parameters of the remaining simulation runs are each averaged, which then yields the ultimate deterministic response vector for the given data point.

Finally, there is one more question. It was mentioned earlier, but the algorithm consists of two hyperparameters: the factor by which the IQR is subtracted from and added to the lower and upper quartiles, respectively, and the absolute z-score from which response instances are considered outliers. Since they are quite difficult to optimize automatically, it was done on a trial-and-error basis by adjusting the parameters while examining the plots of the coefficient of variation, as in Figure B.2. Furthermore, the number of discarded simulations was also monitored throughout the process. The hyperparameters must be determined before starting the active learning process, although the inital set can already be sampled — the Hammersley sequence does not require any meta-model to be trained. Therefore, we use the initial sample for this, because it is known as a very uniform data set that takes the entire feature space into account. Ultimately, this led to the choices on lines 17 and 23 in Algorithm 1, with a z-score threshold of 3 and an IQR factor of 2.5. For the initial sample, the average removal is 1.95 simulation runs per data point and its standard deviation is 2.72. No outliers were found in 39 of the 100 data points, while there was a maximum of 12 outliers in one case. These results are certainly acceptable, especially considering that a total of 256 simulations were performed for every data point. Hence, we conclude that the outlier removal algorithm is effective and performs as desired.

C
Insights into the Sampled Data Points

The greatest strength of adaptive sampling is that it actively searches for the most interesting data points. At the same time, however, this also makes the strategy quite precarious, as it entails a lot of uncertainty. One never knows in advance what part of the feature space will be explored, and how many data points are exactly necessary to end up with a sample that is sufficiently informative. Therefore, this appendix aims to gain more insight by analyzing the collected data, being the initial, adaptive, validation, and test sets. It starts with section C.1, which discusses the distribution of the data points in the feature space. For example, there will be verified whether the initial and adaptive sampling strategies behave as expected. Next, section C.2 continues by presenting the most relevant summary statistics of the responses. This gives a first impression into the output of AATOM. Section C.3 comes last and elaborates on the stopping criterion of the active learning algorithm. More specifically, it examines what exactly triggered the criterion and how many data points were ultimately needed.

C.1. Distribution of the Data Points in the Feature Space

Analyzing distributions of the data points is useful because it immediately shows which parts of the domain have been explored, and to what extent. In order to obtain the broadest possible view, the sample is examined from two perspectives. Namely, subsection C.1.1 and subsection C.1.2 look at the features from their own dimension and from a two-dimensional perspective, respectively.

C.1.1. One-dimensional Perspective

As discussed by the methodology in section 3 of the Scientific Paper in Part I, the total training sample consists of an initial and an adaptive set. The size of the former was recommended to be 10 times the number of input parameters, which corresponds to 100 data points. For the latter, however, this cannot be determined in advance because it depends on the stopping criterion. More on that in section C.3, although we use 200 data points as an initial guess to have enough data to evaluate the criterion upon. This means that the training set provisionally contains a total of 300 data points. With that knowledge, the sample can be visualized by plotting the distributions of its input parameters. The result of that is depicted in Figure C.1, where bar charts were used for categorical features and histograms for the numerical ones. Note that only 4 out of 10 are shown, as the passenger distributions on the other flights were similar to those on flight 7 at the bottom left. The graphs cover the entire training sample, though they are broken down into chunks to see differences between the subsets. First of all, there is the initial data set in blue that was sampled using the Hammersley sequence. According to Table 3.1 of the Literature Study in Part II, the method is praised for its low discrepancy, uniformity, and space-filling behavior — properties that are clearly observable in the one-dimensional distributions. On the other hand, one of the main weaknesses was a poorer coverage at the boundaries of the domain. This is also visible, especially with the two categorical features at the top. Altogether, it is reassuring to see that the characteristics of the initial set match the expectations from the Literature Review. Secondly, the adaptive data set is displayed in the other colors, which is thus again broken down into smaller parts. The reason is to examine whether there are substantial differences between the exploration and the exploitation phases of the active learning process. Namely, it starts with the first 50 data points in the orange adaptive set 1 and ends with the last 50 in the purple set 4. The other two are in between with the remaining 100 data points.

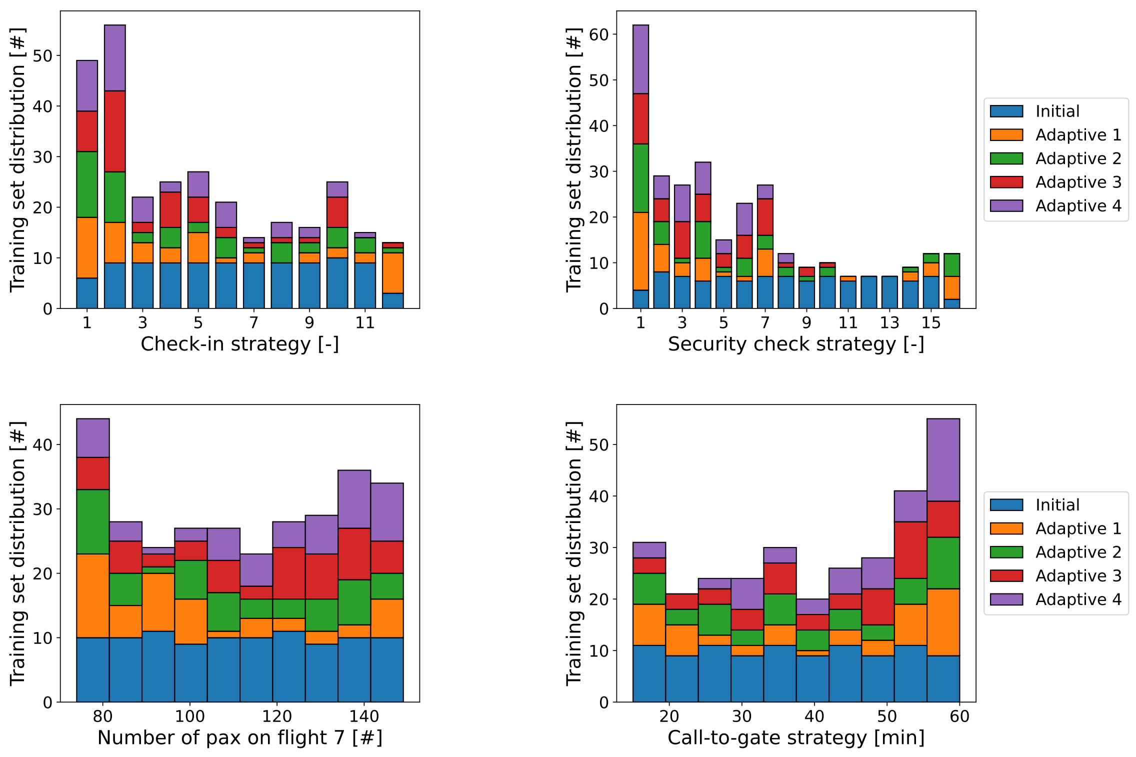


Figure C.1: One-dimensional training sample distributions of four input parameters.

Since the EIGF acquisition function uses a decaying strategy, the first set focuses mainly on exploration while the last one focuses on exploitation. Again, this can be observed in the distributions. The density of the former in orange is greatest at the edges of the feature spaces, which makes sense as the Hammersley sequence was known to underperform there. Hence, the active learning algorithm immediately seems to compensate for this deficiency. The latter in purple is different, as the largest densities are not necessarily located at the boundaries. Instead, it heavily depends on the feature in question, suggesting that the algorithm is indeed searching for the most difficult areas to predict. For the check-in, these are the two extremes: either a very bad or a very good strategy[1]. In terms of security checkpoint occupancy, surrogate models appear to struggle the most with the worst possible strategies, while the best ones are seemingly no problem at all[2]. For the number of passengers on the flights, it is to a greater extent spread out over the domain, although the edges receive slightly more attention. Lastly, simulations with high call-to-gate strategies are apparently harder to predict than lower ones, as can be seen in the lower right histogram. Altogether, these findings are also in line with the expectations. The adaptive sample is clearly no longer uniform, starts with a preference for exploration, and ends with predominantly exploitation. Note that as a result, some areas not revisited after the initial sampling procedure, which is the case for security checkpoint strategies 12 and 13. Yet, we do not consider this a problem, because if it were, the active learning algorithm would have sampled there.

In addition to the training sample, the current research also requires a validation set to optimize the hyperparameters and a test set to evaluate the out-of-sample performance of the surrogate models. According to the methodology in section 3 of the Scientific Paper in Part I, the two sets should be collected using random sampling so that the proportion of each equals 20% of the entire data set. Knowing that the training set has 300 data points for now, both the validation and test set must contain 100 data points to meet this requirement[3]. The one-dimensional distributions of two arbitrary input parameters for each sample are shown in Figure C.2 and Figure C.3, respectively. The graphs for the other features have been omitted for the sake of the report, although they are very comparable to the ones we discuss. They are all fairly well-distributed across the domain, but not quite as uniform as the Hammersley sequence. Nevertheless, this cannot be expected

[1]See Table E.2 in Appendix E.
[2]See Table E.3 in Appendix E.
[3]I.e., $20\% \times (300+2m) = m$, where m denotes the sample size of either the validation or the test set.

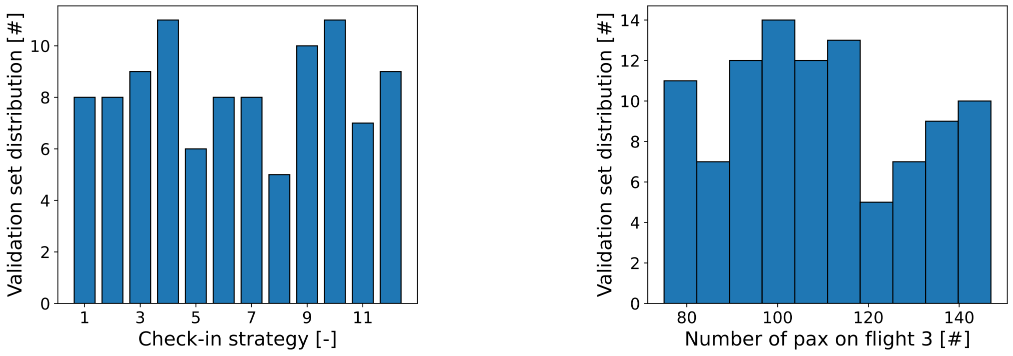


Figure C.2: One-dimensional validation sample distributions of two input parameters.

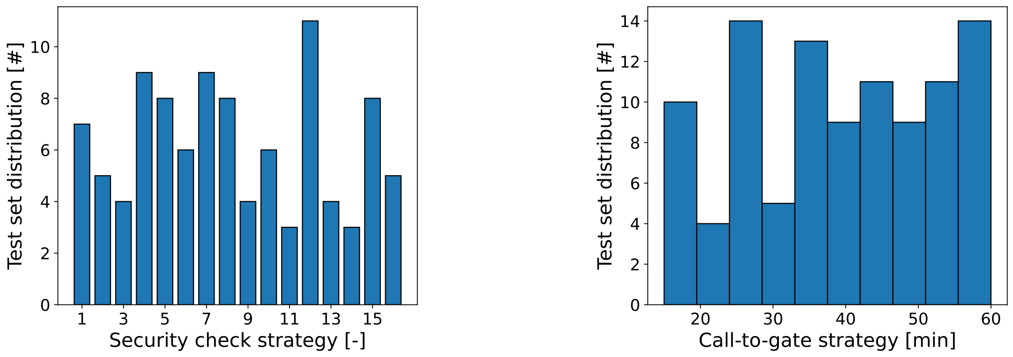


Figure C.3: One-dimensional test sample distributions of two input parameters.

from random sampling. The important finding is that there are no visible patterns or signs of severe clustering around certain areas, despite some settings have indeed been selected more often than others. This suggests that the two are presumably very decent validation and test sets, as a degree of randomness is desirable in order to be as independent of the training sample as possible. In conclusion, the distributions of the data sets reveal no surprises, which verifies the expectations from the Literature Survey.

C.1.2. Two-dimensional Perspective

While the one-dimensional distributions of the features may be in line with expectations, there is no guarantee that the same is true for other dimensions. Indeed, different perspectives can yield different insights, despite the fact that increasing dimensionality makes the analysis more challenging. The two-dimensional perspective is still feasible, but higher than that becomes rather impractical for human interpretation. Therefore, in addition to the previous subsection, we limit ourselves to two dimensions as can be seen in Figure C.4. The plots show the distributions of the initial sample between two particular features. For example, the first histogram on the left visualizes where the Hammersley sequence chose its 100 data points in the domains

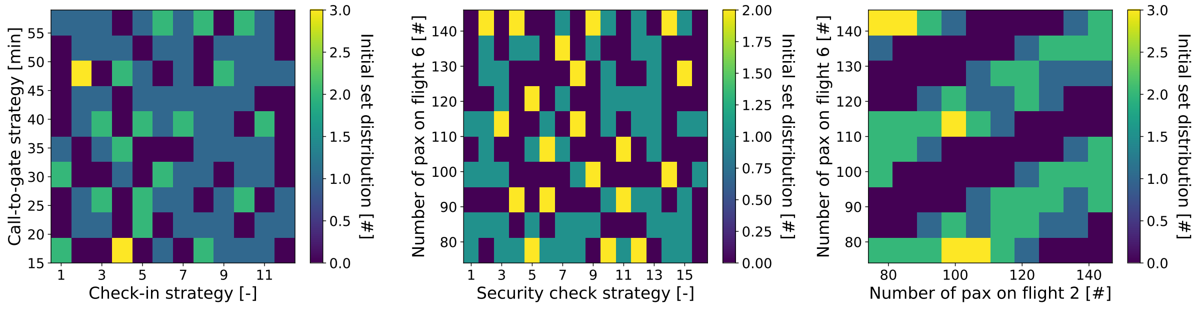


Figure C.4: Two-dimensional distributions of the initial sample.

of the call-to-gate and check-in strategy. In general, almost all graphs are very similar to the two on the left — most of them have been omitted for the sake of the report. It is clear that the initial sampling method maintains its uniformity, low discrepancy, and space-filling properties in 2D. Moreover, the poorer coverage at the boundaries remains, albeit less apparent than in 1D. Hence, the advantages and disadvantages of the Hammersley sequence from Table 3.1 of the Literature Study in Part II seem to withstand a higher dimensional perspective. The histogram on the right, however, is a different story. It is quite surprising to see such a pattern in the distribution of the number of passengers on flight 2 and 6. Despite being rare, these phenomena are of course undesirable because they do not occur in practice. Namely, it is an artifact of the sampling procedure. Recall that the Hammersley set is a mathematical quasirandom sequence, along which the sampler selects data points. This is beneficial because it is accompanied by some favorable properties, such as uniformity and a low discrepancy in high dimensional feature spaces. Yet, there is no guarantee that this also applies to subspaces of the hypercube, as is clearly the case here. Some degree of patterning was expected, but not to the extent discovered in the right graph. It proves the relevance of considering multiple perspectives, as such phenomena are nearly impossible to detect in one-dimensional visualizations. The example will be analyzed in more detail, but first we discuss histograms of the adaptive sample.

These are depicted in Figure C.5. Again, only a small selection is shown, as it would be rather abundant to present them all. The adaptive distributions are very different from the ones of the initial set, although this was to be expected from the one-dimensional analysis by Figure C.1 in the previous subsection C.1.1. In fact, they are very consistent with the earlier findings and reveal no surprising elements. The distributions are certainly not uniform, they are no longer space-filling, and therefore leave gaps in the domain. This makes sense, since the purpose of active learning was solely to explore and exploit the feature space, using prior knowledge from the initial sample. More interestingly, however, is that the two-dimensional histograms disclose the most difficult areas to predict in relation to feature interdependencies. For example, in the left graph, the acquisition function favored check-in strategy 1 and 2 with either low or high passenger numbers on flight 6. The figure in the middle is quite extreme, with many simulations being selected that have check-in and security check strategy 1. None of the other feature combinations have such a skewed distribution, although it does make sense for these two as they are both extremely bad strategies[4]. They result in large queues, for which the system responses are seemingly harder to predict. Lastly, the selected data points for the number of passengers on flight 1 and 7 are more spread out across their domain, as can be seen in the right graph. Nonetheless, the regions where both flights are either quite empty or fully booked are sampled more often. In summary, the two-dimensional distributions of the adaptive sample are in line with expectations. Even though there are no surprises, the visualizations remain useful as they reveal the feature settings that surrogate models struggle with the most.

Now that the adaptive sample has been discussed, we revisit the instance of the initial sample with the observed pattern. The two-dimensional histogram on the right in Figure C.4 shows the overall distribution, although it remains somewhat unclear exactly which data points were selected by the Hammersley sequence. A scatter plot would allow a closer examination, as can be seen in Figure C.6. The focal case between the number of passengers on flight 2 and 6 is depicted in Figure C.6a on the left. While the scatter plot looks as expected from the histogram, there are two elements that stand out. On the one hand, it appears that successive data points are sampled along oblique lines across the two-dimensional feature space. On the other hand, these lines seem to huddle together, resulting in three main clusters. This is rather inconsistent with the

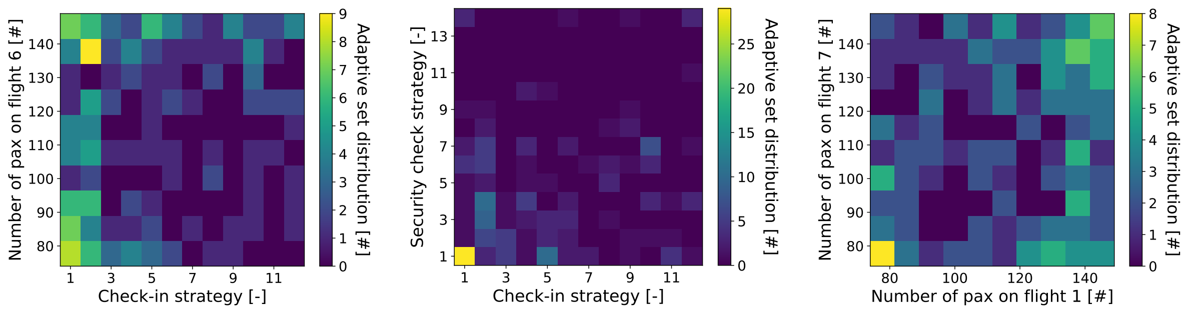
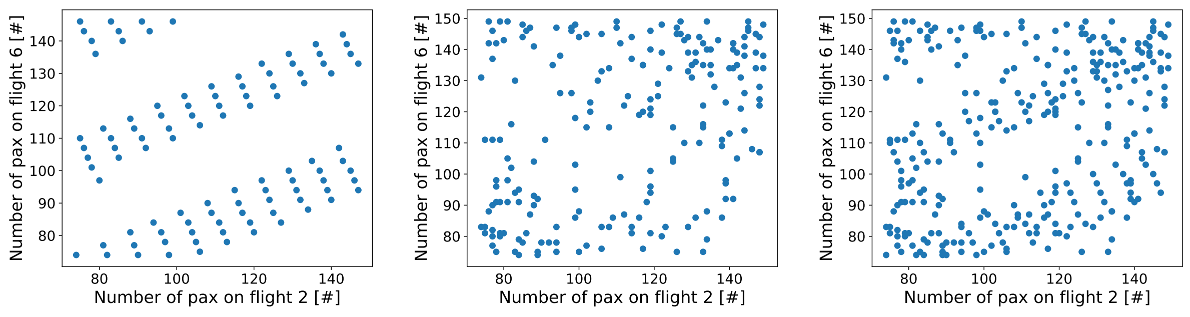


Figure C.5: Two-dimensional distributions of the adaptive sample.

[4]See Table E.2 and Table E.3 in Appendix E.



(a) Initial sample. (b) Adaptive sample. (c) Total training sample.

Figure C.6: Closer examination of the sample patterns between the number of passengers on flight 2 and 6.

properties of the Hammersley sequence, as previously discussed by the Literature Study in Table 3.1. Clearly, the two-dimensional space does show discrepancies, is no longer completely space-filling, and loses its uniformity. Therefore, a deeper analysis is necessary to find out the reason behind these observations. We start by going back into the literature, because it is unlikely that scholars have not encountered the phenomenon before. Kocis and Whiten [56] investigated low-discrepancy sequences and more specifically the effects of dimensionality and sample size. Interestingly, they confirm the two findings from Figure C.6a in their own implementation and devote them to the following. The gaps are caused because the sequence selects data points with a certain periodicity in the domain of the two input parameters, as part of the multi-dimensional feature space. The higher the dimensionality, the larger the discrepancies, which explains the three clusters. Furthermore, if the number of dimensions continues to expand, the data points tend to align. This is clearly visible in Figure C.6a as well. Notwithstanding, the authors mention that the Hammersley sequence is sufficiently uniform for 10 dimensions or less, which was indeed one of the reasons that it was never considered a potential problem during the selection of the initial sampling method. Their arguments are discordant with our findings, but there is a natural explanation for this. Namely, Kocis and Whiten [56] mainly examine samples with a size between a thousand and a million data points, which is substantially higher than our choice of a hundred. As discussed in section 3.6 of the Literature Study in Part II, Loeppky et al. [62] recommended that the initial sample size be 10 times the dimensionality when continuing the sampling procedure with Gaussian process regression. While their argument does not take into account the limitations of the initial sampling method, keep in mind that there is indeed an adaptive part that follows. This was of course not the case in the experiments of Kocis and Whiten [56]. Consequently, one can be lenient towards the initial data set, despite it not being ideal in some of the features' two-dimensional combinations. As a confirmation, we added the scatter plot of the adaptive sample in Figure C.6b, which altogether results in Figure C.6c. The exploration and exploitation efforts of the active learning algorithm compensate the deficiencies of the Hammersley sequence to a large extent. There are still some gaps, but the ultimate training sample in the right scatter plot is certainly acceptable. In addition, remember that the occurrence is rare and that the two-dimensional perspective is only a subspace of the hypercube in which the surrogate models actually operate. We therefore accept the imperfections of the initial sampling method, but include them as limitation in the Scientific Article in Part I.

Finally, the distributions of the validation and test sets are examined from the two-dimensional perspective. A selection of the histograms is depicted in Figure C.7 and Figure C.8, respectively. They are actually all quite similar and exactly as expected. As with the one-dimensional perspective, they fill the domain rather decently, even though there are some discrepancies. However, this is typical for random sampling and it should therefore not be compared to space-filling methods, such as the Hammersley sequence. One can see the randomness — there are no visible patterns or severe clustering of data points, nor are the graphs subject to some statistical probability distribution. The fact that there are again no surprises reinforces the expectations from the Literature Review in Part II, leading us to believe that the validation and test samples are indeed more than fit for purpose. To conclude the two-dimensional perspective, the distributions are generally in line with those in 1D and the notions from the literature. Only the artifacts of the Hammersley sequence were not anticipated to be so apparent in the initial sample, despite the active learning algorithm mitigates most of the patterns. Nonetheless, we closely monitor the ultimate out-of-sample performance of the surrogate models to see if there would be any deteriorating effects.

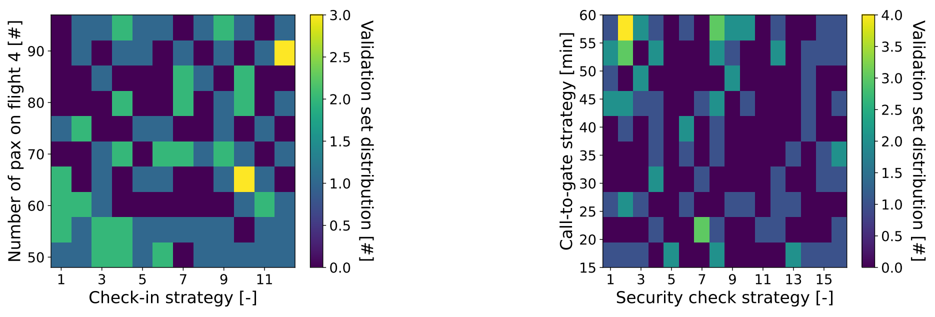


Figure C.7: Two-dimensional distributions of the validation sample.

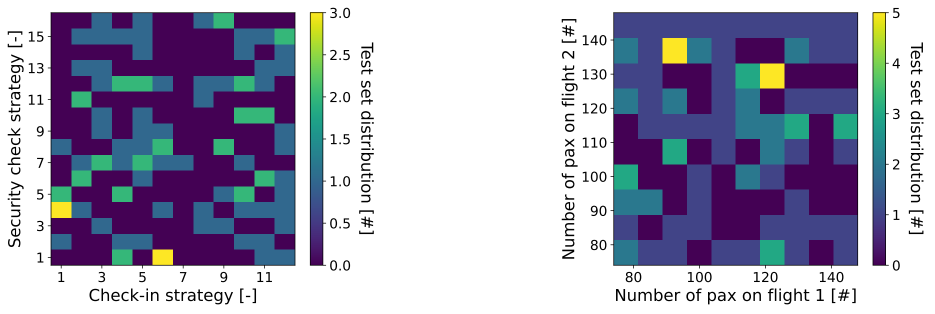


Figure C.8: Two-dimensional distributions of the test sample.

C.2. Summary Statistics of the Responses

Besides visualizing how the input parameters were chosen over their domain, it is also interesting to take a closer look at the corresponding responses. Doing so provides a first insight into the output of AATOM, which will be meta-modeled in the current research. The most relevant summary statistics of the training sample are tabulated in Table C.1. From left to right, there is the mean response, its 95% confidence interval (abbreviated as CI), the standard deviation (abbreviated as SD), and the minimum, median and maximum value of the response. The averages are in line with the expectations. It is interesting to see their interconnections: the average time it takes to reach the gate from entering the terminal, for example, includes the waiting time at security. For those who have not checked in beforehand, the waiting time at check-in is also included. These results show that, on average, the majority of time in the terminal is spent at security. Furthermore, note that the throughput at security is higher than the one at the check-in counters. However, this is very logical. The latter only counts passengers who do so at the airport itself, while everyone has to go through the security checkpoint — there is no way around that. Subsequently, the numbers of passengers who completed check-in and security have the smallest confidence interval and the number of missed flights the largest, relative

Table C.1: Summary statistics of the responses calculated from the training sample.

Response	**Unit**	**Mean**	**95% CI**	**SD**	**Minimum**	**Median**	**Maximum**
AvgQueueTime_SC	[s]	808.61	[766.92, 850.29]	366.89	0.00	718.34	1915.81
AvgTimeToGate	[s]	1379.96	[1342.30, 1417.63]	331.50	0.00	1317.14	2216.35
PaxCompleted_SC	[#]	682.54	[672.06, 693.01]	92.22	0.00	681.90	825.65
PaxCompleted_CI	[#]	356.72	[350.89, 362.56]	51.36	0.00	364.62	447.20
NumMissedFlights	[#]	17.65	[13.07, 22.23]	40.29	0.00	0.00	187.47
TotalExpenditure	[€]	1211.85	[1177.81, 1245.88]	299.55	0.00	1191.48	1841.82
MaxPaxInQueue_SC	[#]	125.41	[119.60, 131.22]	51.16	0.00	118.44	254.96
AvgQueueTime_CI	[s]	275.53	[265.42, 285.63]	88.94	0.00	254.55	681.44
MaxPaxInQueue_CI	[#]	11.94	[11.64, 12.25]	2.69	0.00	11.55	21.81

to the mean. The same can be said of the standard deviation, although the result for the latter response is rather extreme. Namely, the standard deviation of the number of missed flights is more than twice as high as its mean. This shows a high degree of dispersion in the response, which must be right-tailed since it cannot be negative for obvious reasons. Furthermore, the minimum value is zero for all output parameters. While this may seem strange at first, it actually makes sense because flights are canceled if their occupancy rate is less than 50%. Hence, there is a scenario where this applies to all flights, resulting in zero agents in the airport terminal. Such a situation is of course not very plausible in practice, although it is technically possible. In the end, airlines still have the flexibility — albeit limited — not to use their allocated time slots, which are fixed. Flight cancellations therefore remain an option to ensure the versatility of the surrogate models. Next, the median is another interesting statistic, and especially its discrepancy with the mean. It shows to what extent responses are influenced by extrema. We only consider those whose median is outside the 95% confidence interval of the mean, which is true for all except throughput at security and total expenditure. Apart from the throughput at check-in, the remaining responses seem to have a lower median than the average. The number of missed flights is skewed rather extreme, as its median value equals the minimum. This means that in at least 50% of the simulated scenarios, everyone catches their flight. Finally, there are the maxima in the last column, which show no peculiarities. The only response with an exceptionally high maximum is again the number of missed flights. That nearly 190 passengers may not be at their gate on time during the simulated schedule is a lot. However, the reason behind it is evident as it must be caused by a very busy routine with extremely poor airport terminal strategies.

Even though the summary statistics in Table C.1 show the most important properties of the responses at a glance, they are all subject to different scales. It is therefore rather difficult to compare them with one another. This can be resolved by standardization and plotting the results in box plots, as can be seen in Figure C.9. The insights from before are now also directly visible in these graphs. For example, the minimum and median of the number of missed flights do indeed coincide, resulting in a strongly right-tailed distribution. Furthermore, apart from the latter, the interquartile ranges of the remaining responses are more or less comparable. They are not extremely dispersed and while there may be degrees of skewness, the median is never really far from the average. However, it is rather interesting that most of the responses' skew is caused by the values outside the interquartile range. This is especially visible in the throughput at the check-in counters and at security, which show heavy left-tails. Other than that, the waiting time at security and the associated maximum number of passengers in the queue generally appear to be the least variable. All in all, there is certainly some heterogeneity between the responses, albeit not extreme. The distribution for the number of missed flights is skewed the most, so it will be interesting to see how this affects the meta-model performance.

Finally, in addition to statistics and standardized visualizations, it is also relevant to check whether the responses are correlated. This shows how they are influenced altogether and proves the presence of significant relationships, if any. A correlation matrix is presented in Table C.2, along with their p-values. First and foremost, the vast majority of correlation coefficients are rather high, indicating strong positive relationships. These results are not surprising in principle, though some are very high. For example, the correlation be-

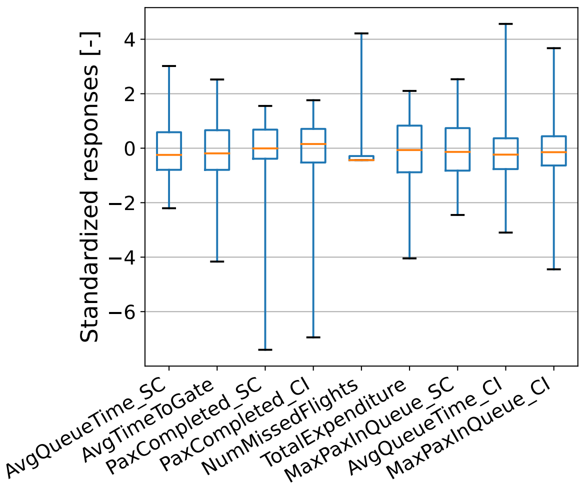


Figure C.9: Standardized box plots of the responses obtained from the training sample.

tween the average time to reach the gate and the average queue time at security is almost perfect. The latter is a direct part of the former, so it does make sense, but also implies that waiting time at security is the main determinant of the total time spent in the airport terminal. Notwithstanding, one must beware of the fallacy of questionable causes [26]. The fact that there is correlation does not necessarily mean that one is caused by the other. An example is the relation between the throughput at security and the maximum number of passengers in the check-in queues. One can see that they are moderately correlated with a coefficient of 0.56. Practically speaking, however, the two responses do not have much in common. Instead, they are both simultaneously affected by the number of passengers in the terminal. The more agents in the system, the higher the two responses are likely to be, which explains the correlation. Hence, the coefficients ought to be interpreted as such: it shows how the input parameters influence AATOM's responses in relation to one another. Furthermore, note that none of the output parameters are negatively correlated with a significance level of at least 5%, and that the total expenditure has the least overall correlation. Other than very weak positive relationships with the throughput at check-in and security, the response has no significant coefficients, making it rather standalone. Lastly, a remarkable finding is the result between the number of missed flights and the security checkpoint's throughput. They are almost unrelated, while there is a weak correlation with the throughput at check-in. This may seem strange, but it actually makes a lot of sense. Think of it as follows. The busier it gets, the more agents that have to pass through check-in and security. The same goes for those who end up missing their flight, but one should realize that this is most likely caused by long waiting times at the checkpoint — note the fairly strong correlation between these two. Indeed, the main bottleneck of the system is at security, which becomes saturated as the crowd grows. At that point, people will start to miss their flight while the throughput remains constant. The checkpoint cannot handle more passengers than its maximum capacity allows. With this knowledge, in combination with the fact that the check-in counters can achieve a much higher throughput in the current terminal layout, one can explain the difference in correlation coefficients with the number of missed flights. These were the most relevant insights from the matrix. Although the results were not surprising per se, they do provide a thorough understanding into how the responses behave in relation to one another.

Table C.2: Correlation matrix of the responses obtained from the training sample.

Response	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1. AvgQueueTime_SC								
2. AvgTimeToGate	0.98**							
3. PaxCompleted_SC	0.47**	0.53**						
4. PaxCompleted_CI	0.75**	0.78**	0.92**					
5. NumMissedFlights	0.80**	0.74**	0.01	0.41**				
6. TotalExpenditure	-0.11	0.02	0.18**	0.13*	-0.10			
7. MaxPaxInQueue_SC	0.92**	0.91**	0.70**	0.86**	0.57**	-0.07		
8. AvgQueueTime_CI	0.36**	0.45**	0.17**	0.31**	0.41**	-0.08	0.31**	
9. MaxPaxInQueue_CI	0.62**	0.70**	0.56**	0.70**	0.47**	-0.04	0.66**	0.86**

Note: * $p < 0.05$, ** $p < 0.01$

C.3. Reaching the Stopping Criterion

A crucial element of the adaptive sampling process is the stopping criterion. This is a predefined condition that determines when the active learning algorithm should stop, since it has no natural ending. The theoretical background is discussed in the Literature Study's subsection 3.2.2 in Part II, though we opted for a hybrid strategy which combines time constraints with out-of-sample meta-model accuracy. In the end, however, it turned out that the availability of time slots was less of an issue than initially thought. This allowed more attention to be paid to surrogate model performance, the result of which is shown in Figure C.10. The graph shows the responses' coefficient of determination over an increasing adaptive sample size. It starts from the initial and ends with the entire training set, which explains the domain of the horizontal axis. Recall that the 200 data points of the active learning process are still a result of the initial guess as stated by section C.1. The plot is created after gathering the entire training sample, and after the surrogates were tuned according to Appendix D. This allows to evaluate the stopping criterion on those models that are ultimately used for

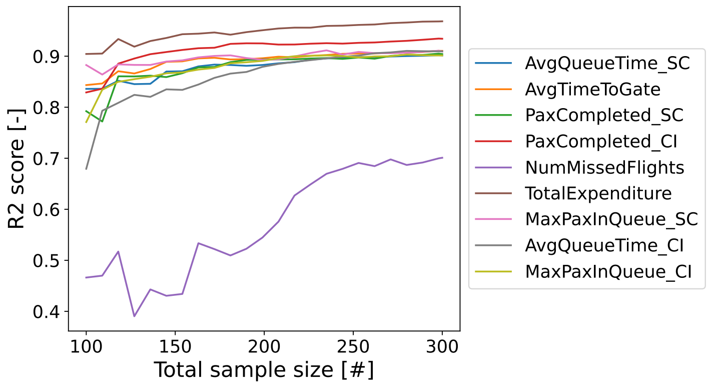
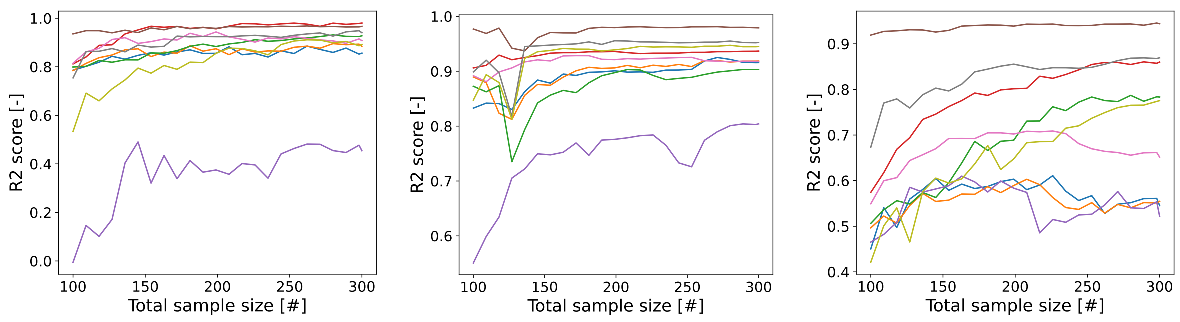


Figure C.10: Out-of-sample performance of Gaussian process regression over an increasing sample size.

the system analysis in a later phase of the research. If the current adaptive sample size proves to be insufficient, additional data points are collected and the whole process is repeated. To determine that, the curves in Figure C.10 must be analyzed, leading to the following conclusions. First and foremost, the out-of-sample performance of the Gaussian process regression models does indeed seem to improve with a larger adaptive sample size. This applies to all nine responses of AATOM. Nonetheless, different responses yield different levels of accuracy and they do not all increase in the same way. Some rise more in the beginning, some more towards the end, and others are in between. There is thus a degree of heterogeneity, although this was to be expected knowing that the input parameters affect them differently. While these insights are certainly interesting, it is more important to analyze the shape of the curves. Namely, they indicate to what extent meta-model accuracy improves when the adaptive sample is expanded. From the perspective of Figure C.10, it appears that the coefficient of determination for all responses has practically flattened as the total training set approaches a size of 300 data points. This suggests that most of the performance improvement potential has been captured and there is no need to continue the active learning process — the initial guess of 200 adaptive data points was thus actually rather good. Notwithstanding, it is a good idea to also consider more than one perspective, as other surrogate model architectures or validation metrics can lead to different conclusions. The acquisition function may solely use Gaussian process regression results to determine the next input parameter combination to sample, it does not mean that the stopping criterion cannot be evaluated on the other architectures.

So similar graphs are visualized in Figure C.11, but now obtained from gradient boosting, polynomial regression, and random forests, respectively. Recall that these are the remaining machine learning architectures that were selected by subsection 3.3.7 of the Literature Survey in Part II. The legend has been omitted for the sake of the report, although the same color code is used as in Figure C.10. Clearly, there is quite a bit of heterogeneity; not only among the responses, but also between the meta-models themselves. Each curve is essentially different. Nevertheless, it seems that the vast majority of output parameters benefit from an increasing sample size. There are some exceptions, especially with random forests, but this architecture has greater problems than responses with diminishing accuracy. We do not elaborately compare the per-



(a) Gradient boosting. (b) Polynomial regression. (c) Random forests.

Figure C.11: Out-of-sample performance of different meta-models over an increasing sample size.

formance of surrogate models here, as this is covered by section 5 of the Scientific Paper in Part I, albeit readily clear that Gaussian process regression, gradient boosting, and polynomial regression generally outperform random forests. The curves in Figure C.11c are indeed rather disappointing. Nonetheless, also from the perspective of different surrogate models, the out-of-sample coefficient of determination appears to be flattening for almost all responses as the adaptive sample size approaches 200 data points. Next to a comparison between machine learning architectures, the same can be done for various validation metrics. This is shown in Figure C.12 for the average time passengers need to reach their gate after entering the airport terminal. As the overall patterns are similar, we only discuss one response to keep the report within reasonable bounds. The graphs show the coefficient of determination, the root-mean-square error, the mean absolute percentage error, and the mean absolute error from top left to bottom right. Moreover, they contain the results for all considered surrogate architectures. One can see that the curves for each type of meta-model are generally comparable across the distinct performance indicators, despite there are of course some subtle differences. More importantly, however, they again seem to flatten near 200 adaptive data points. This confirms our previous finding and suggests that the initial guess is indeed sufficient, especially considering that multiple perspectives have led to the same conclusion. The potential increase in the accuracy of surrogate models by additional sampling is most likely marginal, which does not outweigh the associated computational requirements. Therefore, we conclude that the active learning algorithm reaches its stopping criterion at 200 data points. Consequently, the total training sample consists of 100 data points from the initial sample and 200 data points from the adaptive sample, making a total of 300.

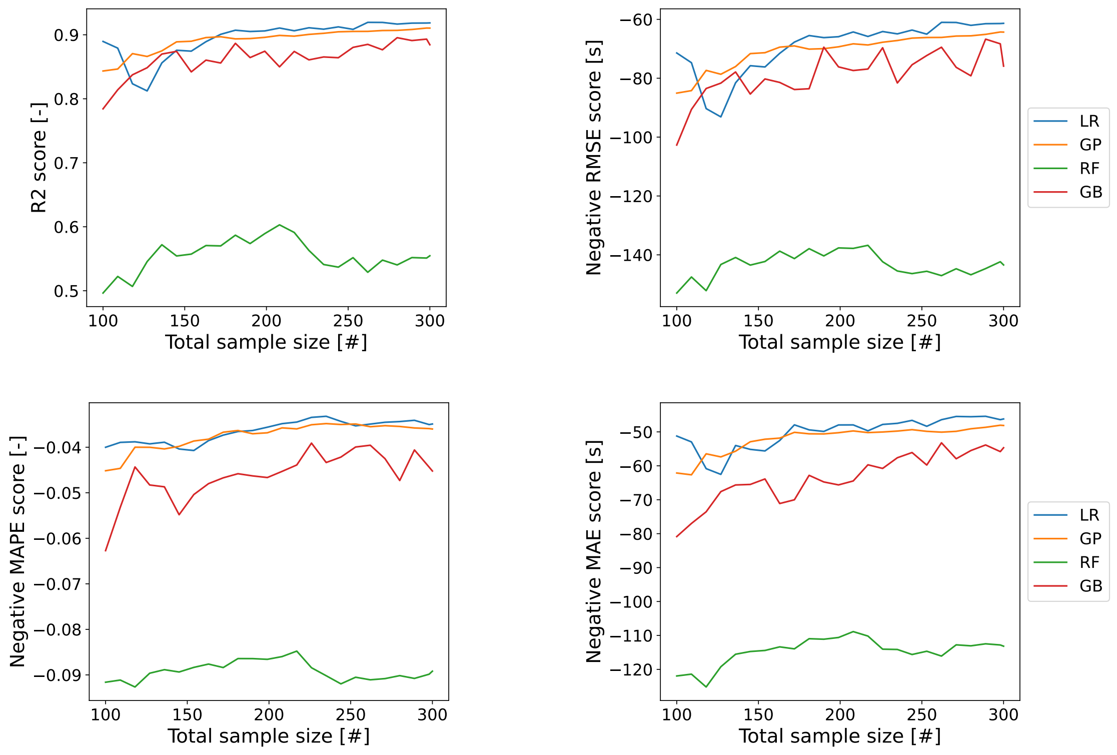


Figure C.12: Comparison between different validation metrics for the average time to reach the gate.

D
Tuning the Meta-models' Hyperparameters

Hand in hand with machine learning comes the optimization of hyperparameters. These are predefined settings that determine how an algorithm is constructed and functions. Consequently, their effect on meta-model performance should not be underestimated, which is why they must be tuned. In section 3.4 of the Literature Study in Part II, we selected Bayesian optimization (BO) as the most suitable method from the alternatives for this task. The outcome of that is presented in the current appendix, starting with an overview of the considered model parameters and their corresponding search space in section D.1. It also contains arguments as to why exactly these parameters are optimized while some of the others are not. Next, there is section D.2, which elaborates on the optimization and its results. This includes the settings of the Bayesian optimizer, the convergence of solutions towards global optima, and an overview of the ultimately selected hyperparameter values.

D.1. Considered Model Parameters

Before thinking about tuning hyperparameters, it is first necessary to determine which settings should be taken into account. Some are evident, while others are more dependent on the problem the model is faced with. Also, it could be that there are specific challenges, such as under- or overfitting, which may require additional attention. Nevertheless, one should refrain from considering too many parameters, as otherwise the dimensionality would complicate the overall optimization process. Several iterations under a trial-and-error strategy have led to the final overview in Table D.1. While the models were originally selected in section 3.3 of the Literature Study in Part II, the selection was slightly modified later in the research process[1]. The table is consistent with the latest arguments in section 3 of the Scientific Paper in Part I. Furthermore, the parameters are tabulated per considered machine learning architecture, along with a short description and their search space. The descriptions are mostly based on the official documentation of the scikit-learn package [75].

First of all, there is the linear regression model (LR), which is effectively implemented as a regularized higher-order polynomial. This naturally leads to an architecture with two hyperparameters: the degree of the polynomial and the regularization strength. The former essentially determines the format of the analytical function between the features and response. The higher, the more complex the model, allowing it to capture higher-order relationships. Higher degrees may seem beneficial at first, though they also entail some specific challenges. Namely, it substantially increases the computational intensity and makes the model more susceptible to overfitting. We have therefore limited the search space to a maximum of four, as this choice still turned out to be feasible. Nevertheless, overfitting remained a rather persistent issue. Despite being undesirable, it was not surprising to encounter the phenomenon. Recall that one of the arguments for surrogate modeling was to lighten the computational burden of using AATOM. Notwithstanding, data is still needed to create the surrogates. The training sample is therefore kept to a minimum, although this in turn increases the possibility of overfitting. So in a sense, it actually becomes an inherent problem with meta-modeling. The traditional remedy for an unsatisfactory generalization is to regularize the machine learning model. In short, this is a mechanism that prevents an algorithm from becoming too complex [34]. While it can be im-

[1] The main difference concerns the linear regression model. At first, the order of the polynomial was fixed at two, as it appeared to be the most common choice in literature. Yet, this is rather subjective and not based on the underlying data. We therefore thought of it as a hyperparameter so that it can be tuned by the Bayesian optimization algorithm.

Table D.1: Overview of the surrogate model hyperparameters [75].

Model	**Parameter**	**Description**	**Search space**[‡]
LR	Degree	Degree of the polynomial function	$\{1,2,3,4\}$
	α	Strength of the L1 regularization penalty	$\{h \in \mathbb{R} \mid 0 \leq h \leq 10\}$
GP	α	Level of noise, which is implemented by means of an L2 regularization penalty	$\{h \in \mathbb{R} \mid 0 < h \leq 1\}$
	l	Length scale of the Matern kernel function	$\{h \in \mathbb{R} \mid 1e-5 \leq h \leq 1e5\}$
	ν	Smoothness of the Matern kernel function	$\{0.5, 1.5, 2.5, \infty\}$
RF	Splitting criterion	Error function used to find the best possible splits	{squared, absolute}
	Number of trees	Number of individual regression trees that make up the parallel ensemble	$\{h \in \mathbb{N} \mid 2 \leq h \leq 1000\}$
	Required data points at a split*	Fewest number of data points needed before a regression tree's node may be split	$\{h \in \mathbb{R} \mid 0.001 \leq h \leq 0.5\}$
	Required data points at a leaf*	Fewest number of data points allowed at the leaves of the regression trees	$\{h \in \mathbb{R} \mid 0.001 \leq h \leq 0.5\}$
	Considered features for a split[†]	Number of input parameters taken into account when searching for new splits	$\{h \in \mathbb{R} \mid 0.1 \leq h \leq 1\}$
	Bootstrap sample size*	Size of the bootstrap samples to fit the individual regression trees upon (i.e., with replacement)	$\{h \in \mathbb{R} \mid 0.01 \leq h \leq 1\}$
GB	Loss function	Error function that should be minimized by adding new trees	{squared, absolute}
	Learning rate	Shrinkage factor of the influence of newly added regression trees	$\{h \in \mathbb{R} \mid 0.01 \leq h \leq 1\}$
	Number of trees	Number of individual regression trees that make up the sequential ensemble	$\{h \in \mathbb{N} \mid 2 \leq h \leq 1000\}$
	Required data points at a split*	Fewest number of data points needed before a regression tree's node may be split	$\{h \in \mathbb{R} \mid 0.001 \leq h \leq 0.5\}$
	Required data points at a leaf*	Fewest number of data points allowed at the leaves of the regression trees	$\{h \in \mathbb{R} \mid 0.001 \leq h \leq 0.5\}$
	Considered features for a split[†]	Number of input parameters taken into account when searching for new splits	$\{h \in \mathbb{R} \mid 0.1 \leq h \leq 1\}$
	Subsample size*	Size of the subsamples to fit the individual regression trees upon (i.e., without replacement)	$\{h \in \mathbb{R} \mid 0.25 \leq h \leq 0.75\}$

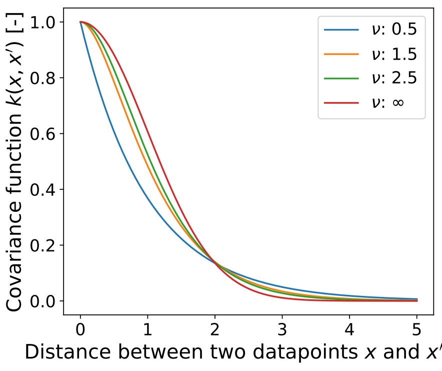
*Expressed as a fraction of the training sample

[†]Expressed as a fraction of the total number of features

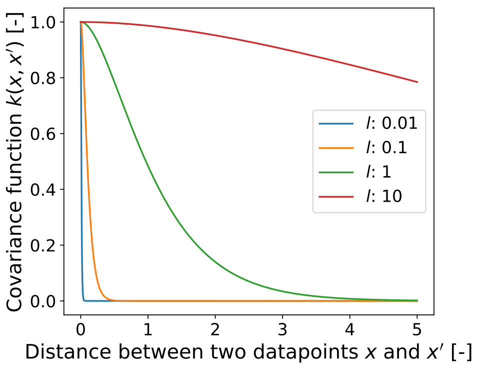
[‡]In the set-builder notation, h represents a particular hyperparameter

plemented in several ways, for LR one typically adds a weighted L1 or L2 norm of the regression coefficients to the objective function. As a result, the model prefers a simpler solution. The weight of this norm is determined by hyperparameter α, where the strength of the regularization increases with a higher value. The parameter cannot be negative for obvious reasons and the upper bound of its search space is 10. This should be more than enough for the BO algorithm to find a suitable setting. Furthermore, note that we opted for the L1 norm. The main reason is for its ability to yield sparse solutions [34]. A higher order polynomial leads to many terms, most of which are probably irrelevant to estimating the response. LASSO can easily cope with this by setting the coefficients of these terms to zero. In summary, the LR model consists of two hyperparameters: one to control the degree of the polynomial function and one to limit its complexity. The Bayesian optimizer is expected to strike a good balance between the two, taking into account the nature of the data.

The second model is Gaussian process (GP) regression. The architecture is known for being rather versatile. On the one hand, this is beneficial because users can tailor the model to their specific problem. On the other hand, such flexibility also entails a certain degree of complexity. With that in mind, the philosophy was to give the BO algorithm as much freedom as possible, but without making it too difficult. This approach is justified as various responses are considered in the current research, each of which may have different characteristics. Indeed, there is no one-size-fits-all and making prior assumptions (e.g., about the kernel function) can affect the ultimate performance of a surrogate if they turn out to be invalid. The first hyperparameter in Table D.1 is α, which incorporates the level of noise in the response. Note that most of variability in the outcome of AATOM is eliminated because of the stabilization efforts in Appendix B. Nonetheless, allowing some margin remains a good idea in case of imperfections and it also increases the robustness of the model by avoiding numerical problems. According to Pedregosa et al. [75], the tolerance for noise is implemented by means of L2 regularization, which avoids the covariance matrix to be ill-conditioned. Therefore, we include α during the optimization process with a strictly positive search space less than or equal to one. Apart from noise, subsection 3.3.2 of the Literature Study in Part II explained that GPs rely heavily on covariance functions to model a response. Several kernels were mentioned, of which the squared exponential is most commonly used in existing literature. Notwithstanding, this choice should really depend on a response and its underlying relationship with the features. A more flexible approach is therefore preferable, so that the Bayesian optimizer can try out different settings. In that regard, a better alternative is to adopt the Matérn kernel as it is more generic. Namely, the smoothness of the covariance function can be modified with a hyperparameter ν. The effect is visualized by a simple sensitivity analysis in Figure D.1a and shows that the smoothness increases with higher values of the parameter [77]. The Matérn kernel is particularly interesting because it is identical to the absolute exponential for a ν of 0.5 and approaches the squared exponential when ν becomes infinity. In between, the values of 1.5 and 2.5 are also relevant for reasons related to the differentiability [75]. With that information, we select the generic Matérn kernel and rely on the BO algorithm for tuning ν. Lastly, there is the length scale parameter l, the effect of which is shown in Figure D.1b. One can easily observe that larger values result in the covariance function being higher at a particular distance between two arbitrary data points $\boldsymbol{x}$ and $\boldsymbol{x}'$. In other words, the responses at the two data points are longer deemed to behave similar. The length scale is in fact also a hyperparameter, although it is not tuned by the Bayesian optimization algorithm. That is because the GP regression model optimizes the parameter itself by means of



(a) The effect of varying ν with a constant l of 1.



(b) The effect of varying l with a constant ν of 1.5.

Figure D.1: Tuning the Matérn kernel function [inspired by visualizations from 77].

the marginal likelihood during the training process [75, 77]. Hence, there is no need to consider it again. The search space is between $1e-5$ and $100,000$, which is the default setting in the scikit-learn package.

The next machine learning algorithm is the random forest (RF) — one of the two selected tree-based ensembles. Clearly, it consists of quite a few hyperparameters that allow users to build the ensemble according to their needs. The first one is the splitting criterion. In essence, this is an error function which determines how the individual regression trees should be split. The main principles behind it are explained in subsection 3.3.5 of the Literature Study in Part II. The default is to use the sum of squared prediction errors, although scikit-learn provides other options as well, such as minimizing the absolute error [75]. While the latter has a higher computational burden, the former may be affected by outliers due to the square. Hence, it is not just a design choice; the characteristics of the data also play a role. We therefore consider the criterion as a hyperparameter so that the BO algorithm can determine the best alternative. The second parameter is to decide on the size of the forest. This is controlled by the number of regression trees, a higher value of which results in a larger ensemble. The aforementioned subsection 3.3.5 explained that the outcome of the algorithm is essentially the average of its constituent trees. Consequently, the variance reduces with a larger forest, which in turn should improve performance [38]. Notwithstanding, the improvements usually stabilizes from a certain size onward, depending on the data. Increasing the number of trees beyond that point just makes it more computationally intensive and should be avoided. The search space is therefore limited between 2 and 1,000. The algorithm needs at least two trees to be an ensemble, while the upper bound should be well above the forest size where maximum performance is achieved. Next, there are also hyperparameters that influence how trees are to be grown. They can be fully expanded until it is no longer possible to split the nodes, although this is generally not a good idea as it degrades the generalization power of the algorithm. Therefore, it is quite common to prune or regularize regression trees, especially when overfitting is likely. One can achieve this by limiting the depth of the trees or by setting a maximum number of nodes, as well as requiring a minimum number of training data points at either the internal or leaf nodes [75]. The effect of these parameters is more or less comparable, even though they control the ensemble differently. We chose to optimize the latter two without restricting the former two, because they allow to define the search space as a function of the data. Namely, the required number of data points is expressed relative to the sample size of the training set. They can both range from 0.1% to 50%, which is broad enough for the Bayesian optimizer to experiment with different settings. Higher values lead to more regularization. Recall that meta-modeling inherently suffers from overfitting, so it may be necessary to reduce the complexity in the regression trees to ensure a satisfactory out-of-sample performance. Lastly, there are two hyperparameters that affect the randomness in the ensemble. On the one hand, the number of considered features determines how many input parameters are selected from the feature space when searching for a new split. The number is a constant during the training phase, but keep in mind that the selection itself is redone for each split. This is to prevent the ensemble from choosing the same features again and again, which fosters heterogeneity among the trees [34]. The hyperparameter is expressed as a faction of the input dimensionality and is bounded between 10% and 100%. Consequently, the tuning algorithm has complete freedom as there are only 10 features in total. On the other hand, the randomness can also be controlled by tuning the size of the bootstrap samples. The background on bootstrap aggregation is again explained in subsection 3.3.5, but in short, the subsamples for training the constituent regression trees are generated by selecting an arbitrary number of data points from the total training set. Note that this happens with replacement, so a data point may be selected multiple times [38]. The bootstrap sample size thus determines how large the subsamples should be. It is also expressed as a fraction of the training set and the search space has a wide range between 1% and 100%. Logically, the degree of randomness increases with smaller bootstrap samples. Knowing that we can influence the randomness, one may wonder why this would be beneficial in the first place. Remember that the ensemble's key principle is to use the average of numerous regression trees so that the variance diminishes. For that to be successful, it is crucial that the individual trees are sufficiently diverse. If this were not the case, there would be a considerable amount of correlation among the trees, hindering the ensemble's benefits. Hence, a degree of randomness is desirable to restrain the correlation [38]. The extent to which this should be stimulated depends on the data and is therefore tuned by the optimization algorithm.

The other tree-based ensemble is gradient boosting regression (GB); the fourth and last architecture under consideration for meta-modeling. The overview in Table D.1 shows that many of the hyperparameters of GB are in fact the same as those of RF. This may not be surprising, although there are some subtle differences nonetheless. The loss function comes first and is actually one of the more important ones. Namely, subsection 3.3.5 of the Literature Study in Part II explained that the boosting algorithm combines numerous regression trees in a sequential manner. Starting from an initial prediction, new trees are fitted onto the errors made by the previous one. The way these residuals are calculated is determined by a loss function[2]. We consider two possibilities: the squared and the absolute loss. From the theory in Hastie et al. [38], it follows that the former is more sensitive to outliers. This may or may not improve performance and depends on the data. Therefore, the function is selected by the tuning algorithm. The second hyperparameter is the learning rate, which controls the influence of newly added trees. After a tree is fitted onto the residuals of its predecessor, it is added to the sequence of the ensemble. However, it is wise to scale its impact first, as otherwise the model would overfit rather easily [34]. So in a sense, the parameter determines how quickly the machine learning model learns from its own mistakes, hence the name. It is implemented as a shrinkage factor and should be between zero and one. We have chosen to raise the lower bound slightly to 0.01, because this makes more sense from a practical point of view — the contribution of new trees would erode with values almost equal to zero and hinder the overall learning process. Of course, the ideal shrinkage also depends on the size of the sequence, as these two interact with one another. This is managed by the number of trees, whose principle and search space are similar to RFs. The only difference is that the regression trees are now boosted instead of bagged, but that has been clearly explained in subsection 3.3.5. Furthermore, note that the following three hyperparameters and their search spaces are also similar to those of random forests: the required data points at a split and at a leaf, and the considered number of input parameters at a split. We do not further elaborate on them here, as they have been extensively discussed in the previous paragraph. Lastly, there remains the subsample size. This parameter determines the number of data points sampled to train each of the individual regression trees. So in a sense, one can compare it to the bootstraps samples of the random forest. Their purpose is the same, which is to reduce the model's variance by introducing a degree of randomness, though there is one key difference. Namely, the subsamples are generated without replacement [38]. This means that a particular data point will never be in the same subsample twice or more. The parameter is usually set to 50%, but the Bayesian optimizer is allowed to select a value between 25% and 75%. Indeed, the ideal setting depends on the data. To be precise, we are thus actually performing stochastic gradient boosting [38].

There are two final notes to conclude this section. Firstly, from the explanations is clear that some of the hyperparameters interact with one another. Consequently, they should be tuned altogether, as otherwise it may lead to sub-optimal solutions. Bayesian optimization handles this quite naturally since the algorithm searches for the global optimum in the entire hyperparameter space of a model [42]. Interactions are therefore automatically taken into consideration. Secondly, it was previously mentioned that the selection in Table D.1 was arrived at after several iterations under a trial-and-error strategy. The other parameters are left at their default values from Pedregosa et al. [75] in the scikit-learn package. However, this does not imply that they have not been experimented with. During the iterations, it just became clear that a better performance was achieved by optimizing the tabulated parameters.

D.2. Results of the Optimization

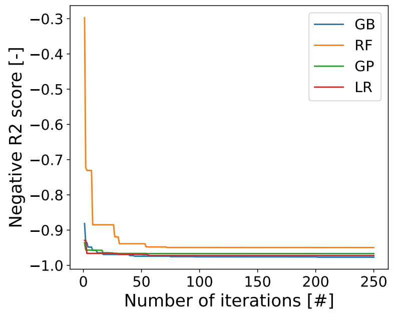
Since it is now clear which hyperparameters ought to be optimized, the next step is to prepare the Bayesian optimization algorithm. The high-level principle has been explained in section 3.4 of the Literature Study in Part II. Briefly, however, a meta-model[3] is fitted onto a prediction error metric of the machine learning model as a function of its hyperparameters. Then, an acquisition function proposes new parameters to try, so that after some iterations the combination is found that yields the highest accuracy. It will not be surprising that the algorithm itself comes with some flexibility. Hutter et al. [43] mention two important design choices: the meta-model architecture and an acquisition function. The former must be probabilistic, as this provides the basis for the latter to evaluate new candidate parameters. Hence, it follows that Gaussian process regression is the traditional alternative, because of its natural way of expressing uncertainty. Another option would be to use random forests, although these are better with higher dimensional and non-numeric parameter spaces [43]. We opt for the traditional choice, as the machine learning models in Table D.1 do not have an excessive dimensionality, while the vast majority of hyperparameters are numerical. Secondly, one should decide on the acquisition function. Archetti and Candelieri [8] discuss three common possibilities, being the probability of improvement, the expected improvement, and the lower confidence bound. The first is one of the oldest and prefers exploitation over exploration. It does not really incorporate the extent of the improvement, which is why scholars came up with the second option — the most popular alternative up to date [43]. However, the expected improvement still favors exploitation. A better balance can be achieved with the third

[2]More precisely, the trees are actually trained on the loss function's negative gradients from the perspective of the latest prediction [38].

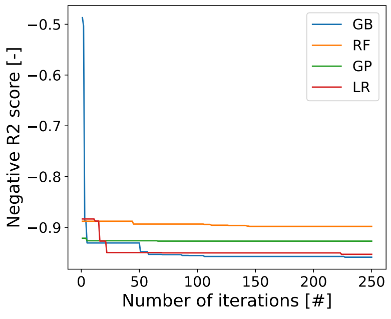
[3]Do not confuse it with the surrogates from Table D.1. This meta-model is a part of the BO algorithm to tune the hyperparameters of those from the table, which are used to mimic the responses of AATOM. To avoid ambiguity in the current section, we will refer to the surrogate models from the table as machine learning models.

function, despite it actually starts to bias exploration [8]. All three of the acquisition functions thus have their advantages and disadvantages. The creators of scikit-optimize, a popular Python implementation for tuning hyperparameters using Bayesian optimization, cleverly solved this by enabling the user to consider them altogether [39]. Namely, at every iteration, one of the three acquisition functions is chosen in a probabilistic manner. That way, there is no need to rely on only one of the alternatives. It is also the package's default setting, from which we do not intend to deviate. Furthermore, recall that the error metric to fit the meta-model on was previously set to the coefficient of determination, as argued in section 3.5 of the Literature Study in Part II. The prediction error is evaluated on the independent validation set, the sole purpose of which was to optimize the hyperparameters (see the methodology in section 3 of the Scientific Paper in Part I). Finally, there remains the number of iterations that must be performed. In principle, the sequential optimization algorithm continues until the coefficient of determination of the machine learning model does not improve further. It is thus rather difficult to decide on this in advance. Moreover, it can also differ per response and model. We therefore start with an estimate of 50 randomly selected parameter combinations for the initial set, after which 200 additional iterations are performed by the Bayesian optimizer. In the next paragraph will be determined whether or not that is sufficient.

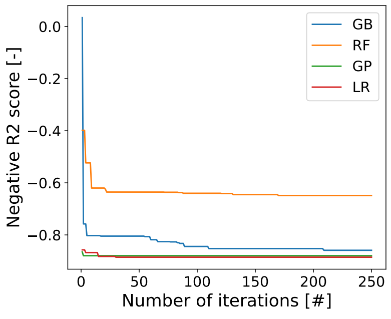
The optimization process can be easily monitored by creating convergence plots. They are presented in Figure D.2, for each of the nine responses. In essence, the graphs show how the performance of machine learning models improves over a number of iterations. Note that all four architectures are combined in one



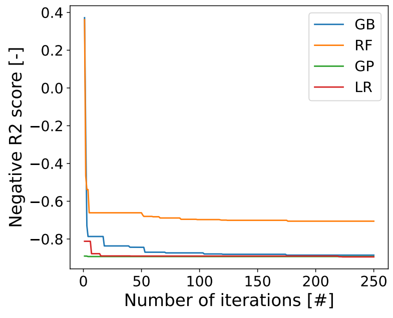
(a) Total expenditure.



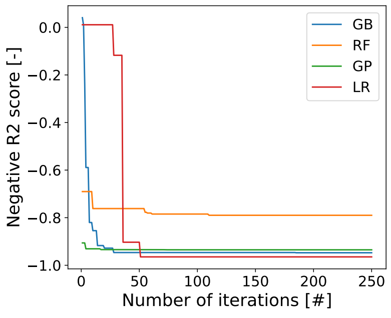
(b) Average queue time at check-in.



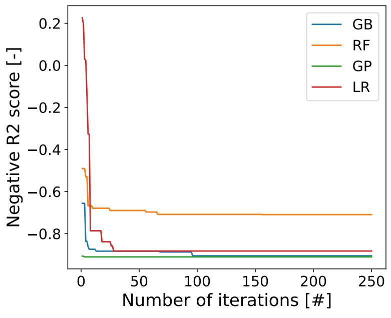
(c) Average queue time at security.



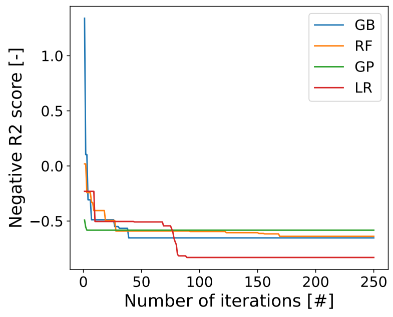
(d) Average time needed to reach the gate.



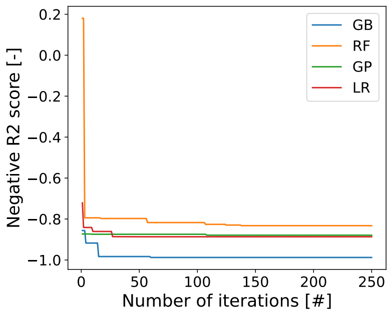
(e) Maximum queue size at check-in.



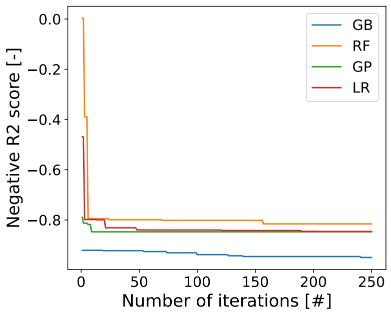
(f) Maximum queue size at security.



(g) Number of missed flights.



(h) Throughput at check-in.



(i) Throughput at security.

Figure D.2: Convergence plots of the hyperparameter tuning process towards the global optimum.

plot per response. The coefficient of determination is on the vertical axis. It is the negative score, as the implementation in scikit-optimize requires a minimization problem [39]. Consequently, accuracy increases with a lower value for the negative R^2. Sometimes it happens that the indicator is positive in the beginning of the optimization. That is, the coefficient of determination is actually negative. This is correct and does not indicate an issue with the underlying computation, but it means that the machine learning model is severely underperforming [75]. On the horizontal axis, there are the number of iterations. The first 50 are the randomly selected hyperparameter values for the initial set, followed by 200 iterations of the Bayesian optimizer. That makes a total of 250. Each trial shows the score corresponding to the best parameter combination of the attempts so far. Thus, the convergence graphs either decrease or remain constant; they never rise. From the plots is clear that the biggest improvements are realized in the beginning. However, this is not surprising, since hyperparameters are arbitrarily chosen from the search space at that time. Hence, it is likely that one of these trials turns out to be rather decent, which explains why there can be a sudden gain in accuracy at some point before 50 iterations. The Bayesian optimization algorithm takes over after that, which results in many smaller decrements. Step by step, it searches for the global optimum. Whether it actually manages to do so is hard to say, although it is evident that the curves stagnate near the end. This suggests that most of the advancements in model performance are realized. Therefore, we conclude that 50 initial random selections and 200 subsequent Bayesian updates are sufficient. Any minor improvements after that would not outweigh the additional computational requirements. Finally, it is worth noting that random forests seem to be underperforming almost consistently: the discrepancy to the other models is oftentimes rather large. This may be somewhat surprising, although it is in line with what was previously discovered in section C.3. GB, GP and LR convergence towards more or less comparable scores, except for the number of missed flights and the throughput at both check-in and security. Respectively, the higher-order polynomial and twice the gradient boosting algorithm turn out to be exceptionally better than the others in those cases. Nonetheless, keep in mind that the optimization results in Figure D.2 are evaluated on the validation and not on the test set. Hence, these insights can be misleading regarding the actual performance of the machine learning models. We do not elaborate on that here, as this is discussed in section 5 of the Scientific Paper in Part I.

After 250 iterations, the machine learning models are considered to be optimal. These are the thus the four models for each response that deliver the best possible performance. The next step is then to select only one of the four architectures per response. Of course, that is the model which can most accurately mimic the response in question. Further details behind the selection are discussed in section 5 of the Scientific Paper in Part I. Notwithstanding, one might be interested in the ultimate hyperparameter combinations to which the Bayesian optimization algorithm has converged. The result is presented in Table D.2 — only the combinations of the selected machine learning models are included for the sake of the report. A higher-order polynomial is chosen for the first seven responses. Most of them have a degree of three, except for the total expenditure and the average waiting time at security, for which four was found to be better. Also, they all seem to benefit from regularization. The vast majority of their regression coefficients are zero because of the selected values for α. The two responses with a fourth-order polynomial have only 148 and 307 non-zero terms against a total of 91,389 coefficients, while the others have respectively 136, 212, 112, 142, and 184 non-zero terms against a total of 9,138. It is thus clear that the resulting polynomials are sparse: the final number of non-zero terms does not explode, despite having rather high degrees. In other words, the models reap the benefits from adding higher-order complexity without experiencing too many of the associated drawbacks. Unnecessary terms are marginalized by regularization, which in this case can actually be seen as a form of feature (and feature interaction) selection. Finding the right balance between the degree and α is not a straightforward task, but the Bayesian optimization algorithm seems to handle it well. In fact, this approach turns out to be quite powerful for meta-modeling AATOM's responses, as it was chosen seven times out of nine. For the remaining two responses, the throughput at check-in and security, gradient boosting was the preferred machine learning architecture. The squared loss function appears to be the best option for both cases, along with fairly low learning rates and a large number of trees. So the influence of new trees in the sequence is limited, although many are added to compensate for that. Furthermore, the throughput at check-in requires at least 14.2% of the training sample before splitting a node, but otherwise there are not many requirements regarding a number of data points at internal nodes or leaves. They are all very close to or at the lower bound of their search space, which essentially means that the trees are not constrained by these hyperparameters. Hence, at first glance it seems that trees are allowed to be fully grown. This may be somewhat surprising, as it severely deteriorates the performance of gradient boosting regression in general [38]. However, there is an important caveat one should not forget. Previous section D.1 stated that Table D.1 only contains the considered parameters for the optimization, which is the result of numerous iterations under a trial-and-error strategy. The other hyperparameters were left untouched. Unlike random forests, scikit-learn limits the maximum tree depth of gradient boosting to three by default [75]. The results in Table D.2 suggest that this setting is the main constraint on tree growth, and thus not the required number of data points. A logical follow-up question is whether the performance would improve if the maximum tree depth were increased or even considered during the hyperparameter optimization. The experiments showed that this was not the case. In fact, the accuracy of the machine learning model mostly declined, which is why we decided to leave the maximum tree depth at its default setting. This choice makes sense, as Hastie et al. [38] argue that boosted tree ensembles are usually not very sensitive to depths between three and seven, with values above five rarely improving the accuracy. Note that as a consequence, the algorithm incorporates features interactions up to the third order. Finally, there are the two hyperparameters to control the ensemble's stochasticity. Both responses prefer a rather high number of features to consider for making splits, while the subsample size is close to or at the lower bound of the search space. At 69.1% and 96.9% respectively, these numbers are likely inflated by the encoding of the airport's staffing strategies. Recall that these are categorical, so more features are required to have enough information on the system (see section 4 of the Scientific Paper in Part I). It seems that the Bayesian optimizer compensated for this by selecting a smaller subsample size. That way, sufficient randomness in the ensemble is still guaranteed, leading to the best possible performance for mimicking AATOM's throughput at check-in and security. The models and associated settings in Table D.2 are those that are ultimately used in the system analysis of the airport terminal.

Table D.2: Selected surrogate model hyperparameter values per response.

Response (selected model)	**Hyperparameter**	**Optimal value**
Total expenditure (LR)	Degree	4
	α	0.494
Average queue time at check-in (LR)	Degree	3
	α	0.096
Average queue time at security (LR)	Degree	4
	α	0.217
Average time needed to reach the gate (LR)	Degree	3
	α	0.224
Maximum queue size at check-in (LR)	Degree	3
	α	0.005
Maximum queue size at security (LR)	Degree	3
	α	0.079
Number of missed flights (LR)	Degree	3
	α	0.013
Throughput at check-in (GB)	Loss function	Squared
	Learning rate	0.045
	Number of trees	990
	Required data points at a split	0.142
	Required data points at a leaf	0.006
	Considered features for a split	0.691
	Subsample size	0.257
Throughput at security (GB)	Loss function	Squared
	Learning rate	0.017
	Number of trees	1000
	Required data points at a split	0.001
	Required data points at a leaf	0.001
	Considered features for a split	0.969
	Subsample size	0.250

E
Annex to the Description of the Agent-based Model

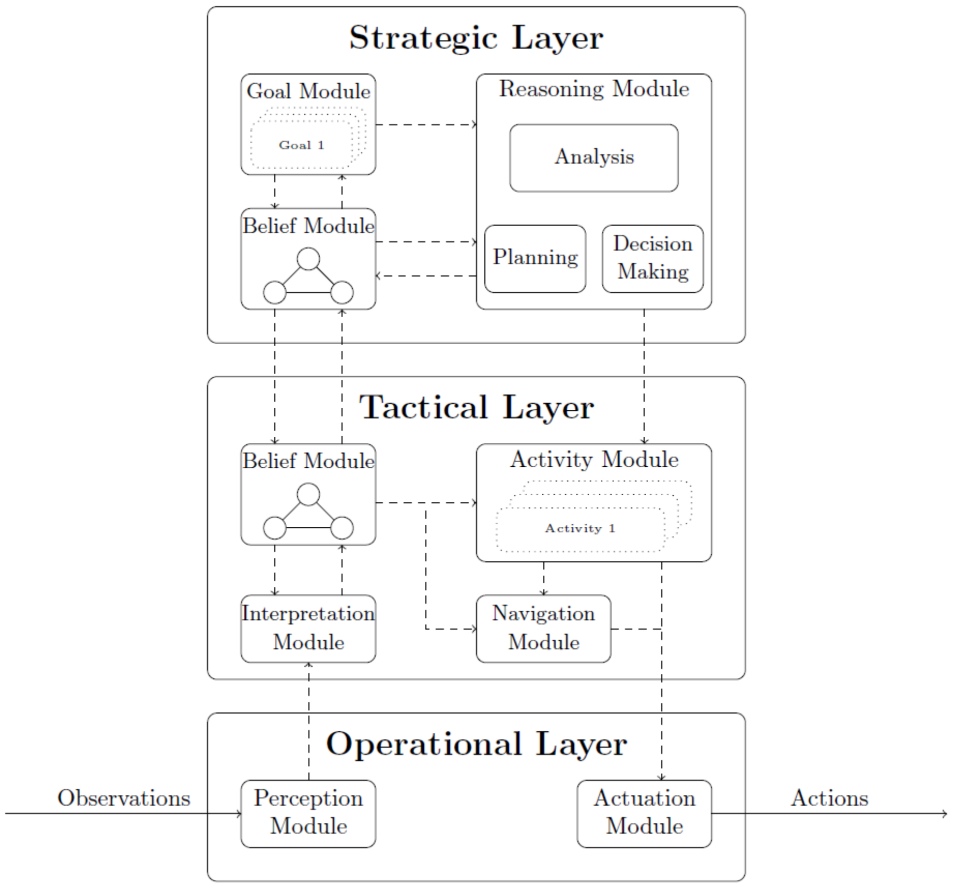


Figure E.1: Architectural layout of agents in AATOM [48]. In essence, they operate as follows: observations are perceived and interpreted, allowing agents to reason so that their activities can be set. This eventually leads to the actuation of specific actions.

Table E.1: A typical flight schedule at RTHA in the fall of 2019. While technically based on assumptions, it resembles the morning rush hour at the airport, which is considered a busy period in the terminal. We take into account 7 flights from 3 airlines to both Schengen and Non-Schengen destinations. The check-in counters are in accordance with the terminal layout in Figure 2 of the Scientific Paper; it goes from the first on the left to the 16th on the right. Furthermore, the number of passengers correspond to usual aircraft types at RTHA, with a flight being canceled if the occupancy is less than 50% of its total capacity. Finally, note that the simulation time in AATOM matches with the time slots. A simulation starts 3 hours before the first flight and ends when the last flight departs.

Nr.	**Slot (UTC)**	**Simulation time (s)**	**Airline**	**Destination**	**Gate**	**Check-in**	**Passengers**
(1)	04:55	9000	A	Schengen	6	9–12	[75, 149]
(2)	04:55	9000	B	Schengen	1	1–4	[75, 149]
(3)	05:00	9300	B	Non-Schengen	7	13–16	[75, 149]
(4)	05:05	9600	C	Non-Schengen	9	13–16	[49, 98]
(5)	05:10	9900	B	Schengen	4	5–8	[75, 149]
(6)	05:30	11100	B	Schengen	2	1–4	[75, 149]
(7)	05:45	12000	B	Schengen	5	5–8	[75, 149]

Table E.2: Presumed check-in staffing strategies at RTHA. The table shows 9 possibilities of how many desks are available. This number can change over time, which is expressed in seconds prior to the departure of a flight. Note that passengers can only check-in between 2 hours and 45 minutes before take-off. All flights of the schedule assume the same strategy over a certain simulated time frame.

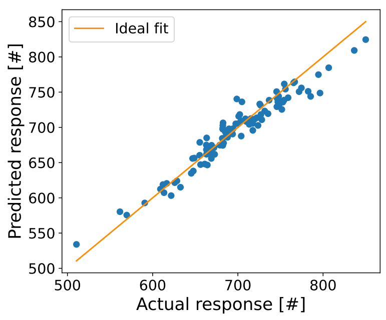
	Available check-in counters as a function of time before departure of a flight						
Strategy	>7200	7200–6300	6300–5400	5400–4500	4500–3600	3600–2700	<2700
(1)	0	1	1	1	1	1	0
(2)	0	2	2	2	2	2	0
(3)	0	1	2	2	2	1	0
(4)	0	1	1	2	2	1	0
(5)	0	1	2	2	1	1	0
(6)	0	1	2	2	2	2	0
(7)	0	1	1	2	2	2	0
(8)	0	2	2	2	1	1	0
(9)	0	2	2	2	2	1	0

Table E.3: Presumed security checkpoint staffing strategies at RTHA. The table shows 16 possibilities of how many lanes are available. This number can change over time, which is expressed in seconds over the simulation time. There is always at least 1 lane staffed, although between 30 minutes and 2 hours the minimum increases to first 2 and then 4 lanes. The reason for this is to ensure a minimum available throughout, because otherwise the terminal may become so crowded that it no longer resembles reality.

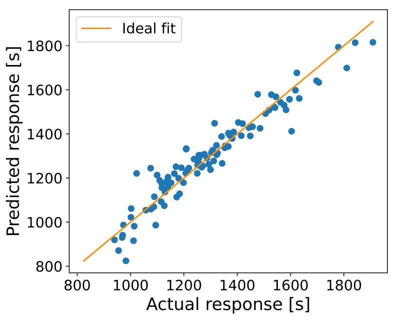
	Available lanes at security as a function of the overall simulation time						
Strategy	<1800	1800–3600	3600–5400	5400–7200	7200–9000	9000–10800	>10800
(1)	1	2	4	4	1	1	1
(2)	1	2	4	4	3	2	1
(3)	1	2	4	4	4	3	2
(4)	1	2	4	4	4	4	4
(5)	1	3	4	4	1	1	1
(6)	1	3	4	4	3	2	1
(7)	1	3	4	4	4	3	2
(8)	1	3	4	4	4	4	4
(9)	2	4	4	4	1	1	1
(10)	2	4	4	4	3	2	1
(11)	2	4	4	4	4	3	2
(12)	2	4	4	4	4	4	4
(13)	4	4	4	4	1	1	1
(14)	4	4	4	4	3	2	1
(15)	4	4	4	4	4	3	2
(16)	4	4	4	4	4	4	4

F
Annex to the Results

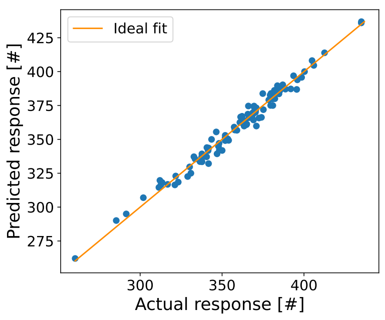
F.1. Surrogate Model Performance



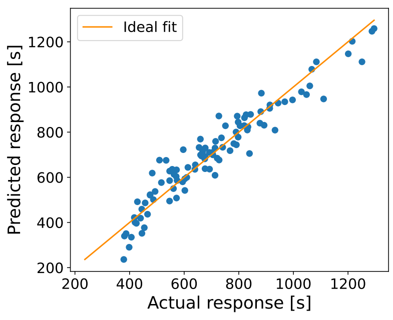
(a) Throughput at security.



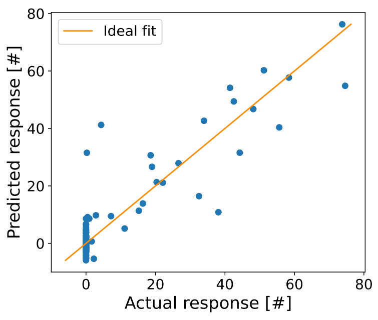
(b) Average time needed to reach the gate.



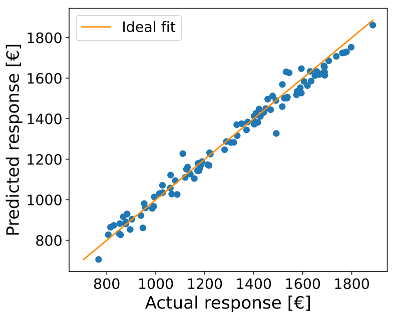
(c) Throughput at check-in.



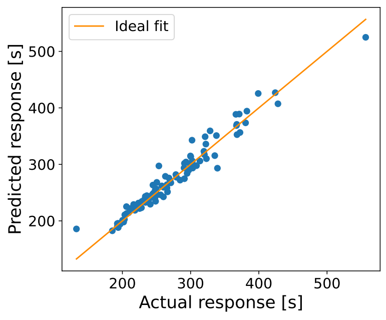
(d) Average queue time at security.



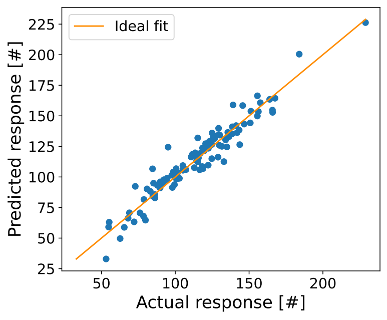
(e) Number of missed flights.



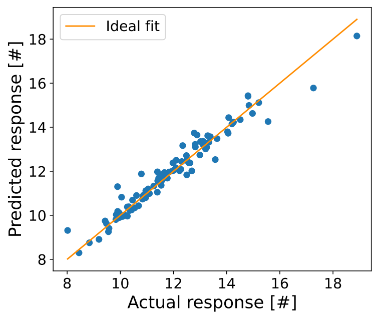
(f) Total expenditure.



(g) Average queue time at check-in.



(h) Maximum queue size at security.



(i) Maximum queue size at check-in.

Figure F.1: Predicted versus actual response values of the best performing surrogate models.

F.2. Analysis of the Total Expenditure on Discretionary Activities

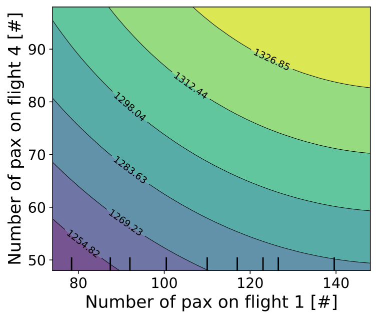


Figure F.2: Marginal effect of flights 1 and 4 on the total expenditure.

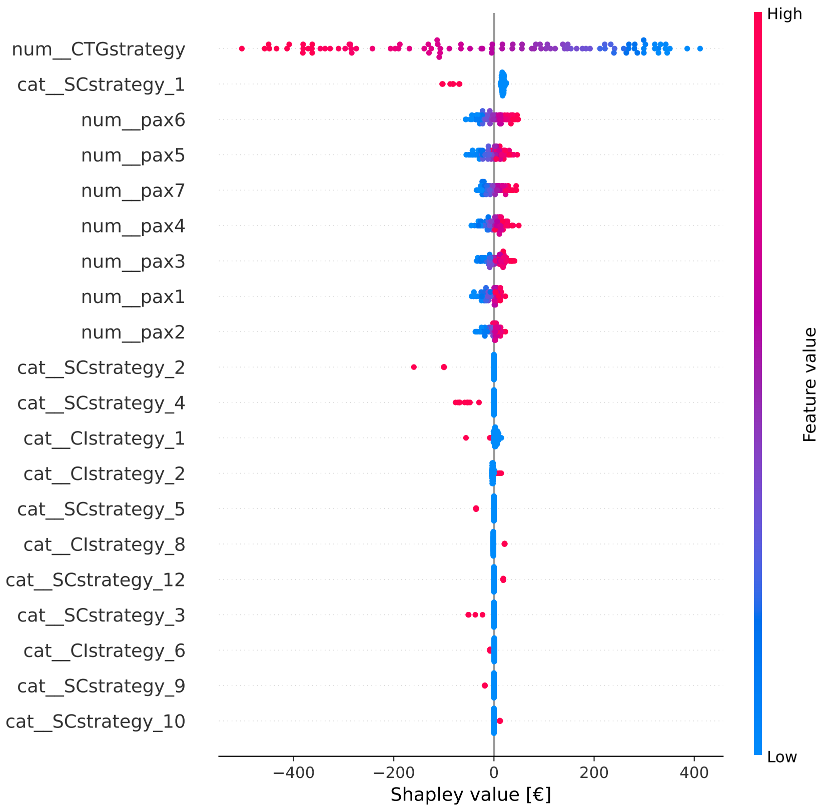


Figure F.3: Bee swarm summary plot of the total expenditure. It shows at one glance to what extent and how the surrogate model is influenced by its individual input parameters. From top to bottom, we have the most influential parameter to the least, but only the foremost 20 are plotted. From left to right, Shapley values are displayed as a function of a feature's value. They should be interpreted as the impact on a model's output and are expressed in the same unit as the response. Finally, note that categorical parameters have been one-hot encoded. Their name starts with 'cat' and ends with the respective category, while numerical ones just start with 'num'.

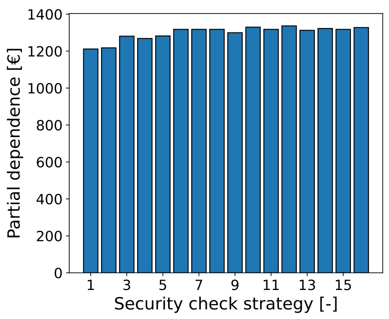


Figure F.4: Marginal effect of the security check strategy on the total expenditure.

F.3. Analysis of the Saturation at Security

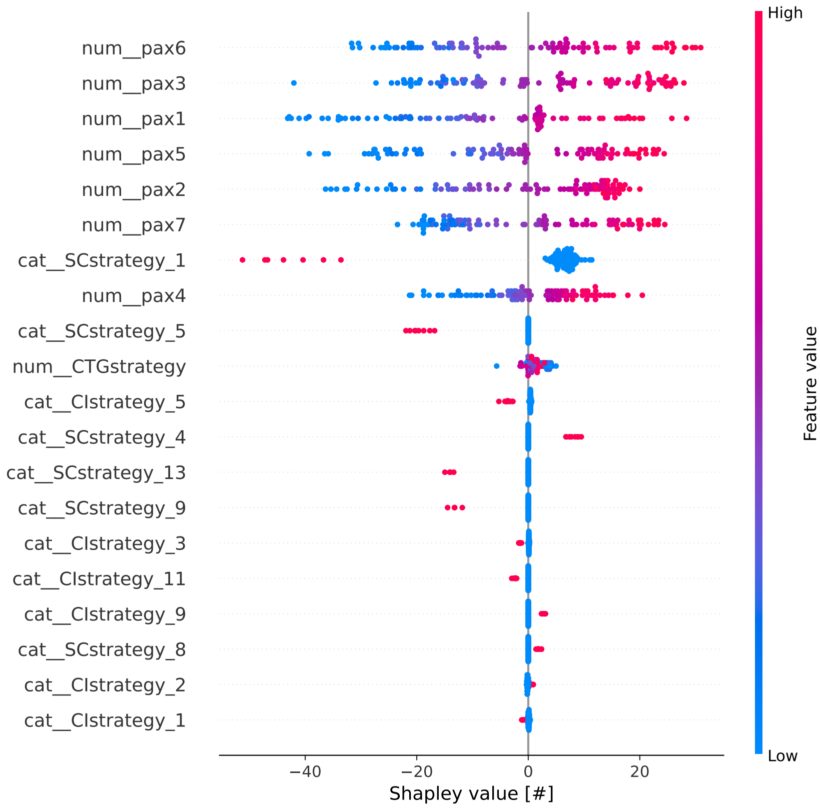
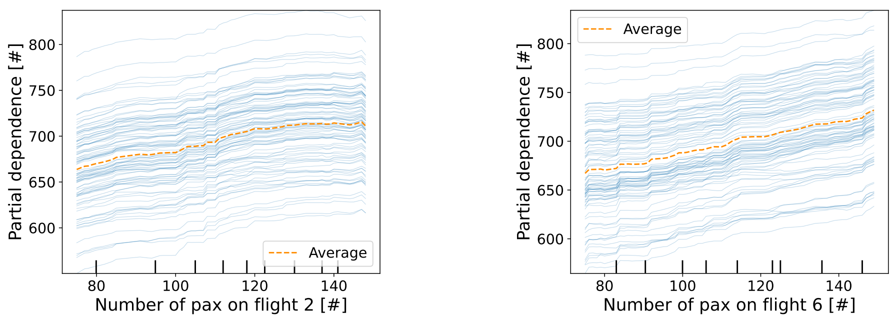


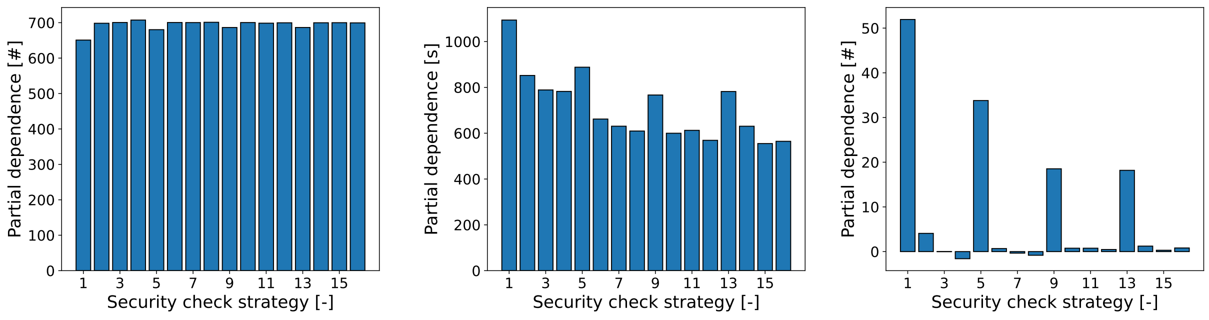
Figure F.5: Bee swarm summary plot of the throughput at security. The interpretation is the same as in Figure F.3.



(a) Marginal effect of flight 2.

(b) Marginal effect of flight 6.

Figure F.6: One-dimensional partial dependence plot of the throughput at security.



(a) Throughput at security. (b) Average queuing time at security. (c) Number of missed flights.

Figure F.7: Marginal effect of the security check strategy.

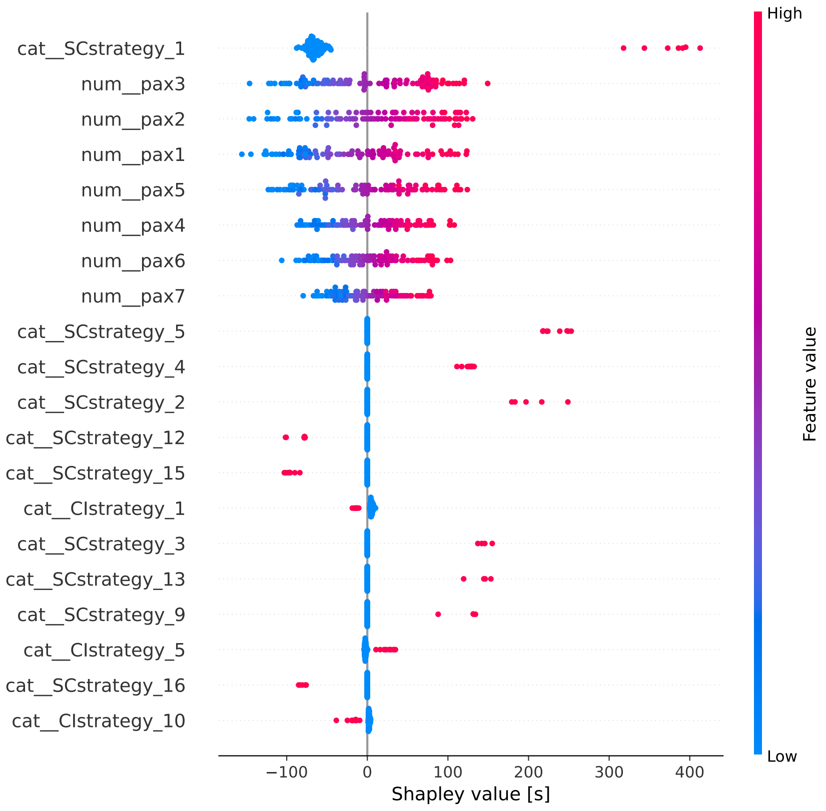


Figure F.8: Bee swarm summary plot of the average waiting time at security. The interpretation is the same as in Figure F.3.

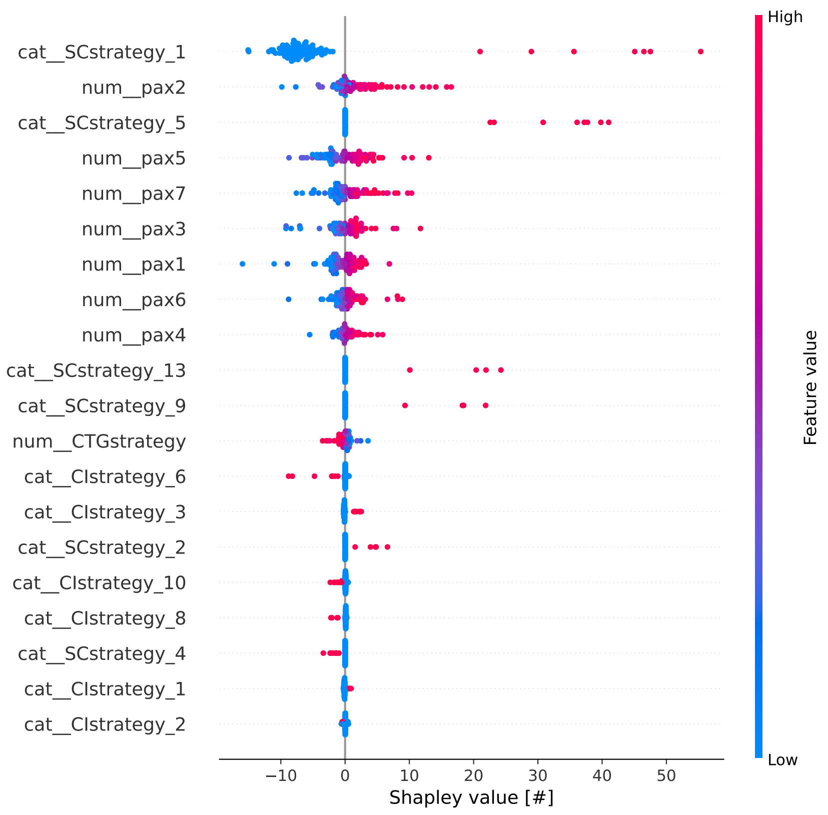


Figure F.9: Bee swarm summary plot of the number of missed flights. The interpretation is the same as in Figure F.3.

Bibliography

[1] Ludovica Adacher, Marta Flamini, Manuele Guaita, and Elpidio Romano. A model to optimize the airport terminal departure operations. *Transportation Research Procedia*, 27:53–60, 2017. ISSN 23521465. doi: 10.1016/j.trpro.2017.12.151. URL https://linkinghub.elsevier.com/retrieve/pii/S2352146517310487.

[2] Hussain Alibrahim and Simone A. Ludwig. Hyperparameter Optimization: Comparing Genetic Algorithm against Grid Search and Bayesian Optimization. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1551–1559, Kraków, Poland, June 2021. ISBN 978-1-72818-393-0. doi: 10.1109/CEC45853.2021.9504761. URL https://ieeexplore.ieee.org/document/9504761/.

[3] Reza Alizadeh, Janet K. Allen, and Farrokh Mistree. Managing computational complexity using surrogate models: a critical review. *Research in Engineering Design*, 31(3):275–298, July 2020. ISSN 0934-9839. doi: 10.1007/s00163-020-00336-7. URL http://link.springer.com/10.1007/s00163-020-00336-7.

[4] Sultan Alodhaibi, Robert L. Burdett, and Prasad KDV. Yarlagadda. Framework for Airport Outbound Passenger Flow Modelling. *Procedia Engineering*, 174:1100–1109, 2017. ISSN 1877-7058. doi: 10.1016/j.proeng.2017.01.263. URL https://linkinghub.elsevier.com/retrieve/pii/S1877705817302631.

[5] Rzvan Andonie. Hyperparameter optimization in learning systems. *Journal of Membrane Computing*, 1(4):279–291, December 2019. doi: 10.1007/s41965-019-00023-0. URL http://link.springer.com/10.1007/s41965-019-00023-0.

[6] Giovanni Andreatta, Lorenzo Brunetta, and Luca Righi. Evaluating terminal management performances using SLAM: The case of Athens International Airport. *Computers & Operations Research*, 34(6):1532–1550, June 2007. ISSN 0305-0548. doi: 10.1016/j.cor.2005.07.024. URL https://linkinghub.elsevier.com/retrieve/pii/S0305054805002352.

[7] Daniel W. Apley and Jingyu Zhu. Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models. *arXiv:1612.08468 [stat]*, August 2019. URL http://arxiv.org/abs/1612.08468.

[8] Francesco Archetti and Antonio Candelieri. *Bayesian Optimization and Data Science*. SpringerBriefs in Optimization. Springer International Publishing, Cham, 2019. ISBN 978-3-030-24494-1. doi: 10.1007/978-3-030-24494-1. URL http://link.springer.com/10.1007/978-3-030-24494-1.

[9] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, June 2020. ISSN 1566-2535. doi: 10.1016/j.inffus.2019.12.012. URL https://www.sciencedirect.com/science/article/pii/S1566253519308103.

[10] Vaishak Belle and Ioannis Papantonis. Principles and Practice of Explainable Machine Learning. *Frontiers in Big Data*, 4:39, 2021. ISSN 2624-909X. doi: 10.3389/fdata.2021.688969. URL https://www.frontiersin.org/article/10.3389/fdata.2021.688969.

[11] Atharv Bhosekar and Marianthi Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering*, 108:250–267, September 2017. ISSN 0098-1354. doi: 10.1016/j.compchemeng.2017.09.017. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135417303228.

[12] E. Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(3):7280–7287, May 2002. ISSN 0027-8424. doi: 10.1073/pnas.082080899. URL http://www.pnas.org/cgi/doi/10.1073/pnas.082080899.

[13] Emanuele Borgonovo and Elmar Plischke. Sensitivity analysis: A review of recent advances. *European Journal of Operational Research*, 248(3):869–887, February 2016. ISSN 0377-2217. doi: 10.1016/j.ejor.2015.06.032. URL https://linkinghub.elsevier.com/retrieve/pii/S0377221715005469.

[14] Kai Cheng, Zhenzhou Lu, Chunyan Ling, and Suting Zhou. Surrogate-assisted global sensitivity analysis: an overview. *Structural and Multidisciplinary Optimization*, 61(3):1187–1213, March 2020. ISSN 1615-147X. doi: 10.1007/s00158-019-02413-5. URL http://link.springer.com/10.1007/s00158-019-02413-5.

[15] Ping Nan Chiang and Kevin Taaffe. Analysis of Passenger Flow in Airport Terminal. In *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 102–105, August 2014. doi: 10.1109/IIH-MSP.2014.32.

[16] Davide Chicco, Matthijs J. Warrens, and Giuseppe Jurman. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ Computer Science*, 7, July 2021. ISSN 2376-5992. doi: 10.7717/peerj-cs.623. URL https://peerj.com/articles/cs-623.

[17] Andrew T. Crooks and Alison J. Heppenstall. Introduction to Agent-Based Modelling. In Alison J. Heppenstall, Andrew T. Crooks, Linda M. See, and Michael Batty, editors, *Agent-Based Models of Geographical Systems*, pages 85–105. Springer Netherlands, Dordrecht, 2012. ISBN 978-90-481-8926-7. doi: 10.1007/978-90-481-8927-4_5. URL http://link.springer.com/10.1007/978-90-481-8927-4_5.

[18] Xiao Cui and Hao Shi. A*-based Pathfinding in Modern Computer Games. *International Journal of Computer Science and Network Security*, 11(1):125–130, November 2010. URL http://paper.ijcsns.org/07_book/201101/20110119.pdf.

[19] D. Curcio, F. Longo, G. Mirabelli, and E. Pappoff. Passengers Flow Analysis And Security Issues In Airport Terminals Using Modeling & Simulation. In *ECMS 2007*, pages 374–379. ECMS, June 2007. ISBN 978-0-9553018-2-7. doi: 10.7148/2007-0374. URL http://www.scs-europe.net/dlib/2007/2007-0374.htm.

[20] Benyamin De Leeuw. Surrogate Modeling of Agent-based Airport Terminal Operations. Master's thesis, Delft University of Technology, July 2021. URL http://resolver.tudelft.nl/uuid:3bd67e27-6f86-4761-b899-2022d7b6c4fc.

[21] Richard De Neufville, Amedeo Odoni, Peter Belobaba, and Tom Reynolds. *Airport Systems: Planning, Design, and Management.* McGraw-Hill, New York, 2nd edition, 2013. ISBN 978-0-07-177058-3.

[22] F.M. Dekking, C. Kraaikamp, H.P. Lopuhaä, and L.E. Meester. *A modern introduction to probability and statistics: understanding why and how.* Springer texts in statistics. Springer, London, 2005. ISBN 978-1-85233-896-1.

[23] Mert Edali and Gönenç Yücel. Analysis of an individual-based influenza epidemic model using random forest metamodels and adaptive sequential sampling. *Systems Research and Behavioral Science*, 37(6):936–958, November 2020. ISSN 1092-7026, 1099-1743. doi: 10.1002/sres.2763. URL https://onlinelibrary.wiley.com/doi/10.1002/sres.2763.

[24] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, April 2004. ISSN 0090-5364. doi: 10.1214/009053604000000067. URL http://projecteuclid.org/journals/annals-of-statistics/volume-32/issue-2/Least-angle-regression/10.1214/009053604000000067.full.

[25] Radwa Elshawi, Mouaz H. Al-Mallah, and Sherif Sakr. On the interpretability of machine learning-based model for predicting hypertension. *BMC Medical Informatics and Decision Making*, 19(1):146, December 2019. ISSN 1472-6947. doi: 10.1186/s12911-019-0874-0. URL https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-019-0874-0.

[26] David S. Fay. A biologist's guide to statistical thinking and analysis. *WormBook*, pages 1–54, July 2013. ISSN 15518507. doi: 10.1895/wormbook.1.159.1. URL http://www.wormbook.org/chapters/www_statisticalanalysis/statisticalanalysis.html.

[27] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019. ISSN 1533-7928. URL http://jmlr.org/papers/v20/18-760.html.

[28] Alexander I. J. Forrester, András Sóbester, and Andy J. Keane. *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley, 1 edition, July 2008. ISBN 978-0-470-06068-1. URL https://onlinelibrary.wiley.com/doi/book/10.1002/9780470770801.

[29] Jerome H. Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1–67, March 1991. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1176347963. URL http://projecteuclid.org/journals/annals-of-statistics/volume-19/issue-1/Multivariate-Adaptive-Regression-Splines/10.1214/aos/1176347963.full.

[30] Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. ISSN 0090-5364. URL http://www.jstor.org/stable/2699986. Publisher: Institute of Mathematical Statistics.

[31] Jerome H. Friedman and Bogdan E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3), September 2008. ISSN 1932-6157. doi: 10.1214/07-AOAS148. URL https://projecteuclid.org/journals/annals-of-applied-statistics/volume-2/issue-3/Predictive-learning-via-rule-ensembles/10.1214/07-AOAS148.full.

[32] Jan N. Fuhg, Amélie Fau, and Udo Nackenhorst. State-of-the-Art and Comparative Review of Adaptive Sampling Methods for Kriging. *Archives of Computational Methods in Engineering*, 28(4):2689–2747, June 2021. ISSN 1134-3060. doi: 10.1007/s11831-020-09474-6. URL https://link.springer.com/10.1007/s11831-020-09474-6.

[33] Anne Graham. *Managing Airports: An International Perspective*. Routledge, New York, 4th edition, September 2013. ISBN 978-0-415-52941-9.

[34] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems*, volume 2th. O'Reilly Media, Sebastopol, CA, 2019. ISBN 978-1-4920-3264-9.

[35] B. Hailpern and P. Santhanam. Software debugging, testing, and verification. *IBM Systems Journal*, 41(1):4–12, 2002. ISSN 0018-8670. doi: 10.1147/sj.411.0004. URL https://ieeexplore.ieee.org/document/5386906.

[36] Daniel Harabor and Alban Grastien. Online Graph Pruning for Pathfinding on Grid Maps. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, page 6, August 2011. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3761.

[37] Toru Hasegawa, Sijia Chen, and Lan Duong. Effects of Novel Coronavirus (COVID-19) on Civil Aviation: Economic Impact Analysis. Technical report, ICAO, Montréal, Canada, September 2021. URL https://www.icao.int/sustainability/Documents/COVID-19/ICAO%20COVID%202021%2009%2022%20Economic%20Impact%20TH%20Toru.pdf.

[38] Trevor Hastie, Robert Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer series in statistics. Springer, New York, NY, 2nd edition, 2009. ISBN 978-0-387-84857-0.

[39] Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, and Iaroslav Shcherbatyi. Scikit-Optimize: Sequential model-based optimization in Python, October 2021. URL https://zenodo.org/record/5565057.

[40] Dirk Helbing, Illés Farkas, and Tamás Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, September 2000. ISSN 0028-0836. doi: 10.1038/35035023. URL http://www.nature.com/articles/35035023.

[41] Victoria Hodge and Jim Austin. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22(2):85–126, October 2004. ISSN 1573-7462. doi: 10.1023/B:AIRE.0000045502.10941.a9. URL https://doi.org/10.1023/B:AIRE.0000045502.10941.a9.

[42] Frank Hutter, Jörg Lücke, and Lars Schmidt-Thieme. Beyond Manual Tuning of Hyperparameters. *Künstliche Intelligenz*, 29(4):329–337, November 2015. ISSN 0933-1875. doi: 10.1007/s13218-015-0381-0. URL http://link.springer.com/10.1007/s13218-015-0381-0.

[43] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Springer International Publishing, Cham, 2019. ISBN 978-3-030-05317-8. doi: 10.1007/978-3-030-05318-5. URL http://link.springer.com/10.1007/978-3-030-05318-5.

[44] IATA. *Airport Development Reference Manual*. Montreal, 9th edition, 2004. ISBN 978-92-9195-086-7.

[45] K. C. James and M. Bhasi. Development of model categories for performance improvement studies related to airport terminal operations. *Journal of Simulation*, 4(2):98–108, June 2010. ISSN 1747-7778. doi: 10.1057/jos.2009.27. URL https://www.tandfonline.com/doi/full/10.1057/jos.2009.27.

[46] S. A. M. Janssen. *Capturing Agents in Security Models: Agent-based Security Risk Management using Causal Discovery*. PhD thesis, Delft University of Technology, April 2020. URL https://doi.org/10.4233/uuid:f9bbff72-b9b4-4694-a188-b2f1451449af.

[47] Stef Janssen. AATOM - An Agent-based Airport Terminal Operations Model, December 2019. URL https://github.com/StefJanssen/AATOM.

[48] Stef Janssen, Anne-Nynke Blok, and Arthur Knol. *AATOM - An Agent-based Airport Terminal Operations Model*. Delft University of Technology, April 2018. URL https://research.tudelft.nl/en/publications/aatom-an-agent-based-airport-terminal-operations-model.

[49] Stef Janssen, Alexei Sharpanskykh, and Richard Curran. AbSRiM: An Agent-Based Security Risk Management Approach for Airport Operations. *Risk Analysis*, 39(7):1582–1596, 2019. ISSN 1539-6924. doi: 10.1111/risa.13278. URL http://onlinelibrary.wiley.com/doi/abs/10.1111/risa.13278.

[50] Stef Janssen, Alexei Sharpanskykh, and Richard Curran. Agent-based modelling and analysis of security and efficiency in airport terminals. *Transportation Research Part C: Emerging Technologies*, 100: 142–160, March 2019. ISSN 0968-090X. doi: 10.1016/j.trc.2019.01.012. URL https://linkinghub.elsevier.com/retrieve/pii/S0968090X1830809X.

[51] Stef Janssen, Alexei Sharpanskykh, Richard Curran, and Koen Langendoen. AATOM: An Agent-Based Airport Terminal Operations Model Simulator. In *Proceedings of the 2019 Summer Simulation Conference (SummerSim '19)*, page 12, Berlin, July 2019. Assoc Computing Machinery.

[52] Stef Janssen, Régis van der Sommen, Alexander Dilweg, and Alexei Sharpanskykh. Data-Driven Analysis of Airport Security Checkpoint Operations. *Aerospace*, 7(6):69, May 2020. ISSN 2226-4310. doi: 10.3390/aerospace7060069. URL https://www.mdpi.com/2226-4310/7/6/69.

[53] Liangyue Jia, Reza Alizadeh, Jia Hao, Guoxin Wang, Janet K. Allen, and Farrokh Mistree. A rule-based method for automated surrogate model selection. *Advanced Engineering Informatics*, 45:101123, August 2020. ISSN 1474-0346. doi: 10.1016/j.aei.2020.101123. URL https://linkinghub.elsevier.com/retrieve/pii/S1474034620300926.

[54] Sofia Kalakou and Filipe Moura. Analyzing passenger behavior in airport terminals based on activity preferences. *Journal of Air Transport Management*, 96, September 2021. ISSN 0969-6997. doi: 10.1016/j.jairtraman.2021.102110. URL https://linkinghub.elsevier.com/retrieve/pii/S0969699721000934.

[55] Jack P. C. Kleijnen. Regression and Kriging Metamodels with Their Experimental Designs in Simulation: Review. *SSRN Electronic Journal*, 2015. ISSN 1556-5068. doi: 10.2139/ssrn.2627131. URL http://www.ssrn.com/abstract=2627131.

[56] Ladislav Kocis and William J. Whiten. Computational investigations of low-discrepancy sequences. *ACM Transactions on Mathematical Software*, 23(2):266–294, June 1997. ISSN 0098-3500. doi: 10.1145/264029.264064. URL https://dl.acm.org/doi/10.1145/264029.264064.

[57] Klemens Köstler. Simulating an Scheduling Airport Security Checkpoints: Q-Learning-Based Allocation of Operators to Security Teams at an Airport Security Checkpoint. Master's thesis, Delft University of Technology, November 2021. URL https://repository.tudelft.nl/islandora/object/uuid%3A4269ed22-debe-432c-8c0f-ac7127341001.

[58] Deepthi Praveenlal Kuttichira, Sunil Gupta, Cheng Li, Santu Rana, and Svetha Venkatesh. Explaining Black-Box Models Using Interpretable Surrogates. In *PRICAI 2019: Trends in Artificial Intelligence*, volume 11670, pages 3–15. Springer International Publishing, Cham, 2019. ISBN 978-3-030-29907-1. doi: 10.1007/978-3-030-29908-8_1. URL http://link.springer.com/10.1007/978-3-030-29908-8_1.

[59] Chen Quin Lam. *Sequential Adaptive Designs In Computer Experiments For Response Surface Model Fit*. PhD thesis, Ohio State University, 2008. URL http://rave.ohiolink.edu/etdc/view?acc_num=osu1211911211.

[60] Francesco Lamperti, Andrea Roventini, and Amir Sani. Agent-based model calibration using machine learning surrogates. *Journal of Economic Dynamics and Control*, 90:366–389, May 2018. ISSN 0165-1889. doi: 10.1016/j.jedc.2018.03.011. URL https://linkinghub.elsevier.com/retrieve/pii/S0165188918301088.

[61] Haitao Liu, Yew-Soon Ong, and Jianfei Cai. A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design. *Structural and Multidisciplinary Optimization*, 57(1):393–416, January 2018. ISSN 1615-147X. doi: 10.1007/s00158-017-1739-8. URL http://link.springer.com/10.1007/s00158-017-1739-8.

[62] Jason L. Loeppky, Jerome Sacks, and William J. Welch. Choosing the Sample Size of a Computer Experiment: A Practical Guide. *Technometrics*, 51(4):366–376, November 2009. ISSN 0040-1706. doi: 10.1198/TECH.2009.08040. URL http://www.tandfonline.com/doi/abs/10.1198/TECH.2009.08040.

[63] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, volume 30, Long Beach, CA, USA, 2017. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

[64] C.M. Macal and M.J. North. Tutorial on agent-based modeling and simulation. In *Proceedings of the Winter Simulation Conference, 2005.*, pages 2–15, December 2005. doi: 10.1109/WSC.2005.1574234. ISSN: 1558-4305.

[65] Liliana Magalhães, Vasco Reis, and Rosário Macário. A new methodological framework for evaluating flexible options at airport passenger terminals. *Case Studies on Transport Policy*, 8(1):76–84, March 2020. ISSN 2213-624X. doi: 10.1016/j.cstp.2018.03.003. URL https://linkinghub.elsevier.com/retrieve/pii/S2213624X18300749.

[66] Ioanna E. Manataki and Konstantinos G. Zografos. Development and Demonstration of a Modeling Framework for Airport Terminal Planning and Performance Evaluation. *Transportation Research Record: Journal of the Transportation Research Board*, 2106(1):66–75, January 2009. ISSN 0361-1981. doi: 10.3141/2106-08. URL http://journals.sagepub.com/doi/10.3141/2106-08.

[67] M. D. McKay, R. J. Beckman, and W. J. Conover. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21(2):239–245, 1979. ISSN 0040-1706. doi: 10.2307/1268522. URL http://www.jstor.org/stable/1268522.

[68] Adin Mekic, Seyed Sahand Mohammadi Ziabari, and Alexei Sharpanskykh. Systemic Agent-Based Modeling and Analysis of Passenger Discretionary Activities in Airport Terminals. *Aerospace*, 8(6):162, June 2021. doi: 10.3390/aerospace8060162. URL https://www.mdpi.com/2226-4310/8/6/162.

[69] Maike Meyer. Flight delayed? What airports are doing in order to avoid delays caused by boarding, November 2019. URL https://www.flight-delayed.co.uk/blog/2019/11/05/flight-delayed-what-airports-are-doing-in-order-to-avoid-delays-caused-by-boarding.

[70] Christoph Molnar. *Interpretable Machine Learning*. Lulu, 2nd edition, February 2019. ISBN 978-0-244-76852-2. URL https://christophm.github.io/interpretable-ml-book/.

[71] M.Z. Naser. An engineer's guide to eXplainable Artificial Intelligence and Interpretable Machine Learning: Navigating causality, forced goodness, and the false perception of inference. *Automation in Construction*, 129, September 2021. ISSN 0926-5805. doi: 10.1016/j.autcon.2021.103821. URL https://linkinghub.elsevier.com/retrieve/pii/S0926580521002727.

[72] Caio Nóbrega and Leandro Marinho. Towards explaining recommendations through local surrogate models. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1671–1678, Limassol Cyprus, April 2019. ACM. ISBN 978-1-4503-5933-7. doi: 10.1145/3297280.3297443. URL https://dl.acm.org/doi/10.1145/3297280.3297443.

[73] Allan Nõmmik and Dago Antov. Modelling Regional Airport Terminal Capacity. *Procedia Engineering*, 178:427–434, 2017. ISSN 1877-7058. doi: 10.1016/j.proeng.2017.01.083. URL https://linkinghub.elsevier.com/retrieve/pii/S1877705817300838.

[74] Paul Pao-Yen Wu and Kerrie Mengersen. A review of models and model usage scenarios for an airport complex system. *Transportation Research Part A: Policy and Practice*, 47:124–140, January 2013. ISSN 0965-8564. doi: 10.1016/j.tra.2012.10.015. URL https://linkinghub.elsevier.com/retrieve/pii/S0965856412001541.

[75] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[76] Bruno Pietzsch, Sebastian Fiedler, Kai G. Mertens, Markus Richter, Cédric Scherer, Kirana Widyastuti, Marie-Christin Wimmler, Liubov Zakharova, and Uta Berger. Metamodels for Evaluating, Calibrating and Applying Agent-Based Models: A Review. *Journal of Artificial Societies and Social Simulation*, 23 (2):9, 2020. ISSN 1460-7425. doi: 10.18564/jasss.4274. URL http://jasss.soc.surrey.ac.uk/23/2/9.html.

[77] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2006. ISBN 978-0-262-18253-9.

[78] Saman Razavi, Bryan A. Tolson, and Donald H. Burn. Review of surrogate modeling in water resources. *Water Resources Research*, 48(7), 2012. ISSN 1944-7973. doi: 10.1029/2011WR011527. URL http://onlinelibrary.wiley.com/doi/abs/10.1029/2011WR011527.

[79] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-Agnostic Interpretability of Machine Learning. *arXiv:1606.05386 [cs, stat]*, June 2016. URL http://arxiv.org/abs/1606.05386.

[80] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA, August 2016. Association for Computing Machinery. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939778. URL http://doi.org/10.1145/2939672.2939778.

[81] Ribana Roscher, Bastian Bohn, Marco F. Duarte, and Jochen Garcke. Explainable Machine Learning for Scientific Insights and Discoveries. *IEEE Access*, 8:42200–42216, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2976199.

[82] Rotterdam The Hague Airport. Over ons. URL https://www.rotterdamthehagueairport.nl/luchthaven-en-ik/organisatie/over-ons/.

[83] Rotterdam The Hague Airport. Welkom in de vernieuwde vertrekhal!, November 2020. URL https://www.rotterdamthehagueairport.nl/vernieuwde-vertrekhal/.

[84] Peter J. Rousseeuw and Mia Hubert. Robust statistics for outlier detection. *WIREs Data Mining and Knowledge Discovery*, 1(1):73–79, January 2011. ISSN 1942-4787, 1942-4795. doi: 10.1002/widm.2. URL https://onlinelibrary.wiley.com/doi/10.1002/widm.2.

[85] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Internal Representations by Error Propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, September 1985. URL https://apps.dtic.mil/sti/citations/ADA164453.

[86] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global Sensitivity Analysis: The Primer.* John Wiley, Chichester, England, 2008. ISBN 978-0-470-05997-5.

[87] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, August 2018. ISSN 0022-2496. doi: 10.1016/j.jmp.2018.03.001. URL https://linkinghub.elsevier.com/retrieve/pii/S0022249617302158.

[88] Timothy W. Simpson, Dennis K. J. Lin, and Wei Chen. Sampling Strategies for Computer Experiments: Design and Analysis. *International Journal of Reliability and Applications*, 2(3):209–240, 2001. URL http://www.personal.psu.edu/users/j/x/jxz203/lin/Lin_pub/2001_IJRA.pdf.

[89] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, August 2004. ISSN 1573-1375. doi: 10.1023/B:STCO.0000035301.49549.88. URL https://doi.org/10.1023/B:STCO.0000035301.49549.88.

[90] Ilya M Sobol. Sensitivity analysis for non-linear mathematical models. *Mathematical modelling and computational experiment*, 1(4):407–414, 1993.

[91] Xueguan Song, Guangyong Sun, Guangyao Li, Weizhao Gao, and Qing Li. Crashworthiness optimization of foam-filled tapered thin-walled structure using multiple surrogate models. *Structural and Multidisciplinary Optimization*, 47(2):221–231, February 2013. ISSN 1615-147X. doi: 10.1007/s00158-012-0820-6. URL http://link.springer.com/10.1007/s00158-012-0820-6.

[92] James Stewart. *Calculus: early transcendentals.* Thomson Brooks/Cole, Belmont, CA, 6th edition, 2008. ISBN 978-0-495-01166-8.

[93] Rick L. Sturdivant and Edwin K. P. Chong. Systems Engineering Baseline Concept of a Multispectral Drone Detection Solution for Airports. *IEEE Access*, 5:7123–7138, 2017. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2697979. URL https://ieeexplore.ieee.org/document/7911176/.

[94] Vojin Tosic. A review of airport passenger terminal operations analysis and modelling. *Transportation Research Part A: Policy and Practice*, 26(1):3–26, January 1992. ISSN 0965-8564. doi: 10.1016/0965-8564(92)90041-5. URL https://linkinghub.elsevier.com/retrieve/pii/0965856492900415.

[95] Didier van der Horst. An improved Tabu Search for optimising the configuration of an agent-based simulation model of a novel security checkpoint. Master's thesis, Delft University of Technology, November 2021. URL https://repository.tudelft.nl/islandora/object/uuid%3A1dac59fd-8682-4856-9ef1-c57519fb9bbf.

[96] Tom Van Steenkiste, Joachim van der Herten, Ivo Couckuyt, and Tom Dhaene. Data-Efficient Sensitivity Analysis with Surrogate Modeling. In *Uncertainty Modeling for Engineering Applications*, PoliTO Springer Series, pages 55–69. Springer International Publishing, Cham, 2019. ISBN 978-3-030-04870-9. URL https://doi.org/10.1007/978-3-030-04870-9_4.

[97] Felipe A.C. Viana. Things you wanted to know about the Latin hypercube design and were afraid to ask. In *10th World Congress on Structural and Multidisciplinary Optimization*, page 9, Orlando, USA, May 2013. URL https://mae.ufl.edu/mdo/Papers/5176.pdf.

[98] Sandra Vieira, Walter H.L. Pinaya, and Andrea Mechelli. Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications. *Neuroscience & Biobehavioral Reviews*, 74:58–75, March 2017. ISSN 0149-7634. doi: 10.1016/j.neubiorev.2017.01.002. URL https://linkinghub.elsevier.com/retrieve/pii/S0149763416305176.

[99] Bowen Wang, Biao Xie, Jin Xuan, and Kui Jiao. AI-based optimization of PEM fuel cell catalyst layers for maximum power density via data-driven surrogate modeling. *Energy Conversion and Management*, 205:112460, February 2020. ISSN 0196-8904. doi: 10.1016/j.enconman.2019.112460. URL https://linkinghub.elsevier.com/retrieve/pii/S0196890419314682.

[100] G. Gary Wang and S. Shan. Review of Metamodeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical Design*, 129(4):370–380, April 2007. ISSN 1050-0472. doi: 10.1115/1.2429697. URL https://asmedigitalcollection.asme.org/mechanicaldesign/article/129/4/370/466824/Review-of-Metamodeling-Techniques-in-Support-of.

[101] Paul Westermann and Ralph Evins. Surrogate modelling for sustainable building design A review. *Energy and Buildings*, 198:170–186, September 2019. ISSN 0378-7788. doi: 10.1016/j.enbuild.2019.05.057. URL https://www.sciencedirect.com/science/article/pii/S0378778819302877.

[102] Uri Wilensky and William Rand. *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo*. MIT Press, Cambridge, MA, USA, April 2015. ISBN 978-0-262-73189-8.

[103] Bianca Williams and Selen Cremaschi. Selection of surrogate modeling techniques for surface approximation and surrogate-based optimization. *Chemical Engineering Research and Design*, 170:76–89, June 2021. ISSN 0263-8762. doi: 10.1016/j.cherd.2021.03.028. URL https://www.sciencedirect.com/science/article/pii/S0263876221001465.

[104] David H. Wolpert. What Is Important About the No Free Lunch Theorems? In Panos M. Pardalos, Varvara Rasskazova, and Michael N. Vrahatis, editors, *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, Springer Optimization and Its Applications, pages 373–388. Springer International Publishing, Cham, 2021. ISBN 978-3-030-66515-9. URL https://doi.org/10.1007/978-3-030-66515-9_13.

[105] Tien-Tsin Wong, Wai-Shing Luk, and Pheng-Ann Heng. Sampling with Hammersley and Halton Points. *Journal of Graphics Tools*, 2(2):9–24, January 1997. ISSN 1086-7651. doi: 10.1080/10867651.1997.10487471. URL http://www.tandfonline.com/doi/abs/10.1080/10867651.1997.10487471.

[106] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, November 2020. ISSN 0925-2312. doi: 10.1016/j.neucom.2020.07.061. URL https://linkinghub.elsevier.com/retrieve/pii/S0925231220311693.

[107] Raul Yondo, Esther Andrés, and Eusebio Valero. A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses. *Progress in Aerospace Sciences*, 96:23–61, January 2018. ISSN 0376-0421. doi: 10.1016/j.paerosci.2017.11.003. URL https://www.sciencedirect.com/science/article/pii/S0376042117300611.

[108] Yuya Yoshikawa and Tomoharu Iwata. Gaussian Process Regression With Interpretable Sample-Wise Feature Weights. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2021. ISSN 2162-2388. doi: 10.1109/TNNLS.2021.3131234.

[109] Tong Yu and Hong Zhu. Hyper-Parameter Optimization: A Review of Algorithms and Applications. *arXiv:2003.05689 [cs, stat]*, March 2020. URL http://arxiv.org/abs/2003.05689.

[110] Elena D. Zidarova and Konstantinos G. Zografos. Measuring Quality of Service in Airport Passenger Terminals. *Transportation Research Record*, 2214(1):69–76, January 2011. ISSN 0361-1981. doi: 10.3141/2214-09. URL https://doi.org/10.3141/2214-09. Publisher: SAGE Publications Inc.