Surrogate Modeling of Agent-based Airport Terminal Operations

Master of Science Thesis

Benyamin De Leeuw

Surrogate Modeling of Agent-based Airport Terminal Operations

Master of Science Thesis

by

Benyamin De Leeuw

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on July 28th, 2021.

Student number:	4471121
Project duration:	April 1st 2020 – July 28th 2021
Thesis committee:	Prof. Dr. ir. H.A.P. Blom — TU Deflt, Chairman
	Dr. ir. O.A. Sharpans'kykh — TU Delft, Supervisor
	Dr. B. Chen — TU Delft, External Examiner

TU Delft

Acknowledgements

Dear reader,

This research has been led in the context of my Aerospace Engineering Master thesis at TU Delft, following the master track in Control & Operations with a specialisation in Air Transport Operations. The study was centered around the surrogate modeling of Agent-based Airport Terminal Operations models.

It has been a long learning experience for finishing my studies at the faculty of Aerospace Engineering at the TU Delft. I would like to thank my supervisor Dr. Alexei Sharpanskykh for his patience during this long journey and for all his support. I also want to thank Sahand and Adin for meeting with me on a regular basis to guide me through several challenges that I faced and for Sahand's contribution to the thesis paper with the addition of an ANN-based surrogate model. Thank you as well to Gregory, Klemens and Didier for accompanying me through the AATOM thesis journey.

Finally, I want to thank my friends and family for being there at every moment of my adventure at the TU Delft. This is my last project that concludes 6 years of joy and work at the TU Delft.

Benyamin De Leeuw
Delft, July 2021

Contents

List of Figures

List of Tables

List of Abbreviations

AATOM	Agent-based Airport Terminal Operation Model
ABM	Agent-Based Model
ANN	Artificial Neural Network
CART	Classification And Regression Tree
CV	Cross-Validation
DOE	Design Of Experiment
KR	Kriging
LOOCV	Leave-On-Out Cross-Validation
MAPE	Mean Absolute Percentage Error
MARS	Multi Adaptive Regression Spline
PR	Polynomial Regression
RBF	Radial Basis Function
RF	Random Forest
VIM	Variable Importance Measure
WTMD	Walk-Through Metal Detector

I

Scientific Paper

Surrogate Modeling of Agent-based Airport Terminal Operations

Benyamin De Leeuw,[*]

Delft University of Technology, Delft, The Netherlands

Abstract

Airport terminal operations are playing a crucial role in the air transportation system. The airport terminals are complex sociotechnical systems, which are difficult to understand and their behaviour is hard to predict. Hence, an agent-based model, the Agent-based Airport Terminal Operation Model (AATOM), has been designed to represent and analyse diverse airport terminal processes, actors, their behavior and interactions at a detailed level. However, the main issue with complex agent-based models is the large computational requirements for simulating detailed processes, making it a timely inefficient simulation method. Therefore, the goal of this research is to approximate the dynamics of AATOM by a surrogate model, while preserving the important system properties and to better understand the model underlying functions. A methodology is suggested for training and validating a surrogate model, based on the Random Forest algorithm. The trained surrogate model is capable of approximating the AATOM and identifying relative importance of the model variables w.r.t the model indicators. Firstly, the results obtained contain an evaluation of the surrogate model accuracy performance, indicating that the surrogate model can achieve an average accuracy of 93%. Nonetheless, one indicator, the number of missed flights, has shown to be more difficult to predict, with an average accuracy of 83%. Secondly, the results show that the resource allocation has an important impact on the efficiency of the airport terminal, with the two most important variables being the number of desks at the check-in and the number of lanes at the checkpoint. Last, the developed surrogate model was compared with a second Artificial Neural Network surrogate model. The comparison demonstrated the same order of prediction accuracy and the same two most important variables. Ultimately, the results have shown that the developed surrogate model can serve as an accurate approximation of the AATOM, which can be part of a broader framework for airport terminal decision-making.

Index Terms - Surrogate modeling, Agent-based model, Random Forest, Airport terminal

1 Introduction

The airport terminal plays a crucial role in the modern air transportation system. Previous studies have focused on modelling and simulating the airport terminal operations, concentrating mainly on security analysis [Janssen, 2020, Janssen et al., 2019]. For this purpose a terminal operation model (AATOM) has been developed for modelling and analysis of complex sociotechnical systems with diverse interacting actors and highly dynamic and uncertain processes. Agent-based modeling has proven to be a reliable method [Lee et al., 2015]. The model represents a multi-agent system where every agent is considered an autonomous entity that can observe and act with its environment in order to achieve certain goals or solve complex problems. The main issue with agent-based models, however, is the stochastic nature and the complexity of the systems. The model dynamics are hard to understand [Lee et al., 2015]. Approaches exist to grasp a better understanding of complex systems. One of the approaches is surrogate modeling (also called metamodelling), used to better understand systemic properties of the original model. In essence, it consists of generating a "model of the model", obtaining an approximation of the original model. Surrogate modeling is used through various domains with the objective to emulate/surrogate an existing agent-based model [van der Hoog, 2018]. Subsequently, the surrogate models are used for calibration [Lamperti et al., 2018], validation [Zhang et al., 2020] or behaviour space exploration [Edali and Yücel, 2019]. The behaviour exploration is a valuable application of the surrogate model, in addition to the advantage of the model being more timely efficient than the original agent-based model.

Hence, two elements form the basis of the research, being the original AATOM model and surrogate modelling to abstract this model. The former, from its agent-based nature, poses challenges for understanding the behaviour of the system under given conditions. The latter is an approach for approximating the AATOM. Thus, leading to the following research objective/question: *"How to obtain an approximation of an agent-based model for simulating airport terminal operations while preserving the important dynamic (emergent) properties of the model and getting an insight into the underlying function?"* The objective can be seen as two-folded:

[*]Msc Student, Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology

approximating AATOM while preserving the properties and getting a better understanding of these properties. The first part underlines that by applying the surrogate modelling method (Random Forest) on the AATOM model, an approximation of the model can be obtained, preserving the system properties by accurately predicting the model indicators under given conditions. The last part is the ability of the approximation obtained from the original AATOM model to reveal the relations between the model inputs and outputs, leading to a better understanding of the system behaviour. The main contributions of the study are the generation of a surrogate model from AATOM, the evaluation and validation of the surrogate model and the comparison of the developed surrogate model (the Random Forest model) with another surrogate model (the Artificial Neural Network model).

In the following in this paper, background elements regarding AATOM and the surrogate model used for approximation are given in Section 2. Furthermore, the methodology developed for generating the surrogate model based on the AATOM is described in Section 3. The building blocks of the methodology are: the model (AATOM) configuration, the simulation, the training of the surrogate model and the evaluation of the surrogate model. Section 4 presents the results obtained by applying the methodology, focusing on the performance of the surrogate model and the input-output relationships. The results are further discussed in Section 5. Last, the conclusions are drawn in Section 6 and suggestions are given for future work.

2 Background

In this Section some background is given on two aspects of the research: the model used (AATOM) and the surrogate modelling method applied on the AATOM. Both are specific to this research, and define the nature of the original system and the method used for approximating the system. Furthermore, the methodology presented in the next Section is developed around the two elements presented in this Section.

2.1 AATOM Model

A brief overview of AATOM is given in this Section in order to provide a better understanding of its working and capabilities [Janssen, 2020]. It is considered in this research as a building block of the methodology and the source of data for both training and validation of the surrogate model, thus crucial to elaborate.
The AATOM is an agent-based simulation model used to represent passengers and airport terminal staff in the context of airport terminal operations. It comprises the specification of agents constructed on the principles of agent-based modeling and simulation, developed on the elements that define agent-based modeling: the agent (and its properties), the environment and the interaction between both (and among agents). The agent has a three-layered architecture, with each level adding a layer of abstraction, given in Figure 1. The three layers are: operational, tactical and strategical. In essence, the three-layered architecture dictates how the agent observes the environment and the other agents, and the way the agents interacts with both, based on the observations. Ultimately the agent is able to make certain decisions based on the beliefs about the environment and the other agents.

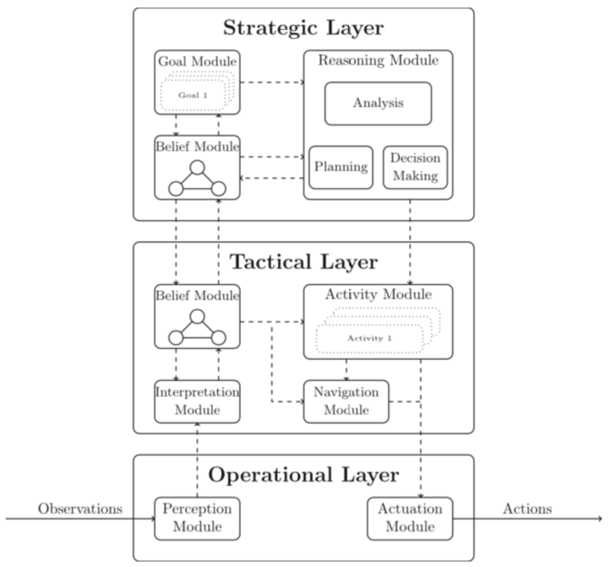


Figure 1: The AATOM architecture and its different modules [Janssen et al., 2019].

The environment is composed of three objects: the areas, the flights and the physical objects. The first being two-dimensional polygons that delimit the different terminal areas (check-in, checkpoint, entrance, gate and facility). Every point in the model is either inside or outside of an area. Thus, given an agent's location (a point in the model), it can always be determined in which specific area an agent is located. The second environmental object, the flight, is an abstract object. The flight is defined by the flight time and the flight type. The former indicating the scheduled time and the latter indicating whether the flight is an arrival or a departure. The last environmental object is the physical object. The object is a vector with three properties: transparency, blocking and sensor.

Thus, the AATOM module is used for the research as a supporting structure to model an airport terminal. The specifications of the AATOM used in this study are further presented in Section 3.1.

2.2 Surrogate Modelling

Surrogate modeling is the approach of generating an approximation of the model in order to reduce the complexity while maintaining the dynamic properties of the original model. The surrogate model is obtained by using a learning algorithm to obtain an abstraction of the model. The study includes two surrogate models. The first model presented is developed using the methodology described in Section 3. The second is used for comparison purposes and a brief explanation of its elaboration is given. Hence, both models are described in this Section.

The algorithm chosen for generating the first surrogate model, based on the AATOM simulation data, is the **Random Forest**. The learning algorithm has been proven to be a reliable and efficient method for working with large data-sets, achieving low computational costs [Villa-Vialaneix et al., 2012]. In addition, the implementation of the algorithm for surrogate modelling is relatively simple. Random Forest relies on the bagging principle and decision trees. Bagging (bootstrap-aggregating) is a scheme in which bootstrap samples are generated from a given data-set. Subsequently, a predictor is constructed from the sample and overall predictions are made by averaging over the predictors' results [Biau and Scornet, 2016]. In other words, Random Forest can be seen as an ensemble of trees that determine, by means of majority vote between the trees [Boulesteix et al., 2012], the resulting output from a given combination of input values. The trees, called decision trees, are constructed on a number of splits that generate nodes until the terminal node (leaf) is attained [James et al., 2013]. The splitting criterion is the CART-split criterion [Biau and Scornet, 2016]. At each split in the decision tree, based on CART-split, the best cut is selected by optimizing the prediction squared error. Thus, given a set of observations, the decision tree determines the region to which each observation of the set belongs.

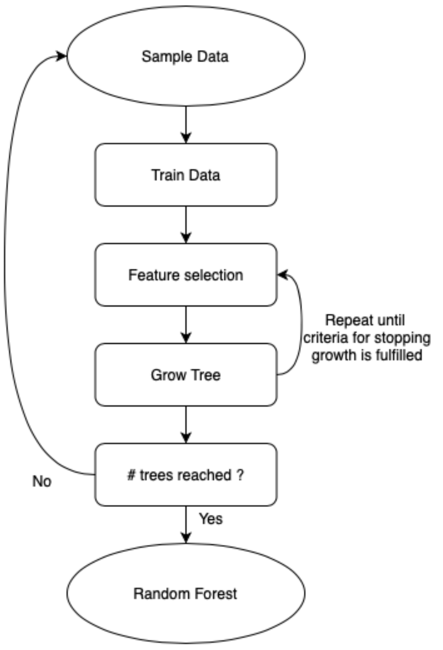


Figure 2: Schematic representation of the Random Forest Algorithm.

Furthermore, there are classification and regression trees, differing on the nature of the target variable, being qualitative or quantitative respectively. In this research only regression trees are considered, having only continuous AATOM target variables. The target variables, elaborated in Section 3, are all continuous as they represent different airport terminal indicators measuring processing times or counting the passenger flux. In addition, the criterion for stopping the growth of a tree, with other parameters inherent to tree growth and Random Forest are further elaborated in Section 3.

In Figure 2, the flow diagram for the Random Forest algorithm is given. As can be seen, the growth of the decision trees is an internal iterative process of the Random Forest, indicated by the two steps: *Feature selection* and *Grow tree*. The first step is the selection of the variable for splitting the node. The second step, *Grow tree*, is the growth of the decision tree by splitting the node. Moreover, the overall process consists of sampling training data to generate individual trees until the total number of trees criterion is reached. Subsequently, the grown trees are used as an ensemble to make predictions on given input value combinations.

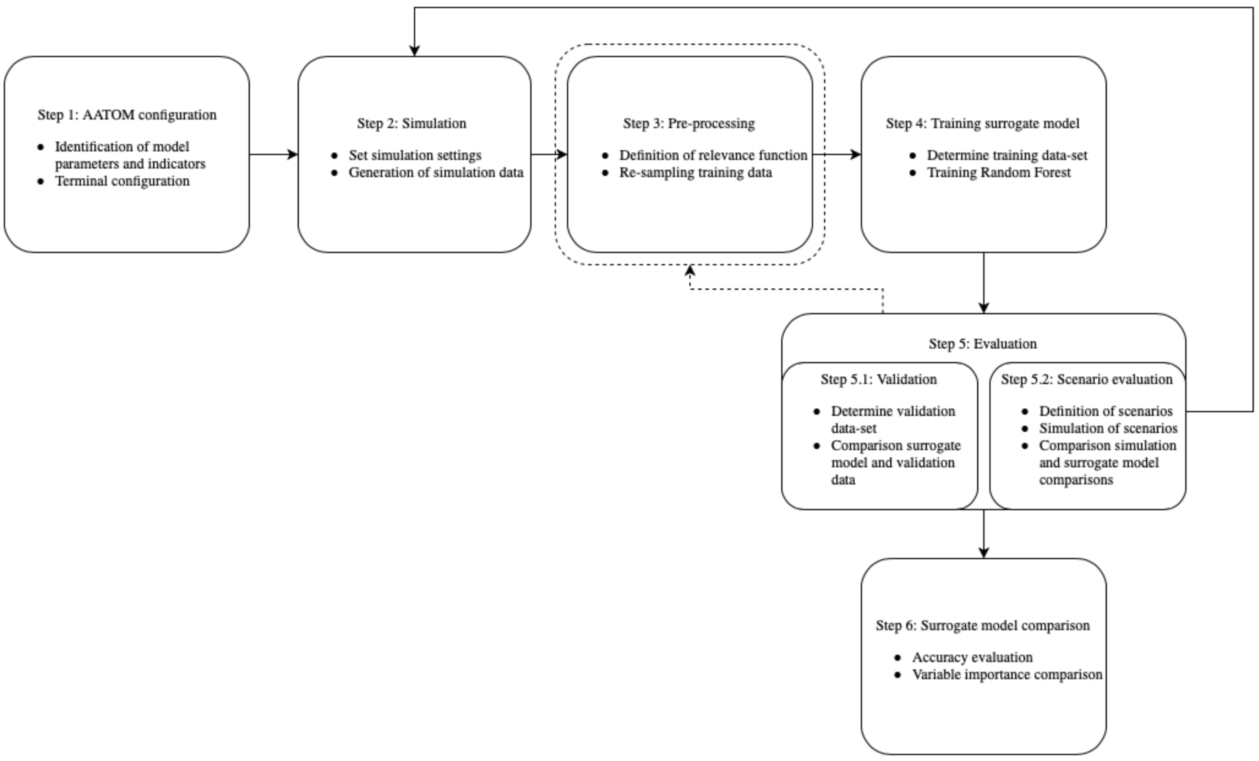
In addition, the Random Forest method contains a specific measure for ranking the different parameters on their relative importance, called the variable importance measure (VIM) [Behnamian et al., 2017]. Measuring the relative importance of the variables can identify specific relations between the model variables and indicators. The VIM is a measure taken on the change of prediction accuracy of the RF model by applying a random permutation to each variable. Hence, a relative ranking can be determined of the different model parameters, given additional insights into the underlying relationships of the original (AATOM) model. The application of VIM on the trained surrogate model is given in Section 4. Last, the implementation of both the Random forest algorithm and the VIM is done in a Python environment, using the Scikit-learn library [Pedregosa et al., 2011].

The second model is based on **Artificial Neural Network** (ANN).The key motivation to use ANN is the low computational cost and the capability of approximation. ANNs have good generalization properties, can deal with large datasets and are able to represent nonlinearity. However, vanishing gradient is the main problem of ANN in the back propagation. The number of hidden layers and hyperparameters have to be defined properly while the inputs and output layers are set according to the amount of input and output variables. In order to measure the sensitivity of parameters, different algorithms, mainly focusing on the connection weights in artificial neural networks, have been proposed. In [Kira and Rendell,] feature selection is described as 'the problem of choosing as small subset of features that ideally is necessary and sufficient to describe the target concept'. The main issue with the previous algorithm [Garson, 1991] was that when the training finished, it naturally assumed that the higher the amount of weights, the more important a parameter while that is already known that regularization techniques make weights to not increase and become smaller [Goodfellow et al., 2016]. The method which we have used here to measure the relative importance of parameters in the AATOM simulation's output is called variance-based feature importance by use of neural networks [Welford, 1962]. This is based on the principle that the more important is a parameter, the higher the weights, which are connected to the corresponding input neurons, vary during the training of the model. To measure the relative importance of each parameter, variances of each weight connected to the input layer is calculated in the training time [Sadeghyan, 2018]. The algorithm that we have used here is an adaptation of Welford's online algorithm [Welford, 1962] for computing these variances. This method is used for calculating corrected sums of squares that is the sum of squares of the deviations of the values about their mean.

The methodology proposed for analyzing the sensitivity of parameters is a combination of the result of AATOM from simulation together with the neural network, this occurs in two phases. In the first phase, the security check-point parameters of the AATOM are used as an input for the neural network. The influence of each parameter uncertainty on output uncertainty is considered through a series of forced perturbations on the parameters. The variation of the six parameters of interest in the security checkpoint in the airport terminal in AATOM, and in the second phase, the result of changes with respect to the variation of each parameter is considered for the output layer in the neural network. In our case, the inputs of the neural network are the parameters of the AATOM where parameters (e.g. number of open lanes in the security check-point) are changing, and the output of the neural network is the simulation result of average queue time in security check-point. The training and validation of the neural network for the prediction part are performed and eventually, the more important the weight from each input (Welford's method) is considered as the importance of that parameter. To measure the sensitivity of each parameter such as average queue time in the security check-point, we have used the method proposed in [Welford, 1962]. The obtained running variance that has been proposed by Welford [Welford, 1962] is used for analyzing sensitivity in the neural network that has been proposed in [Sadeghyan, 2018]. The importance for each input in the neural network is defined by the variances of the weights combined with the corresponding last connection weight. Due to the problem with storing all weights during training (with hundreds of epochs to be trained and changing weights at each batch), the running variance of each weight which is connected to the input layer is saved instead [Welford, 1962]. Eventually, all variances connected to each input layers are aggregated to figure out the most important inputs. We have used regression mode with two activation functions, RELU and Linear, in which the loss function was Mean Squared Error.

3 Methodology

A step-wise systematic iterative procedure is defined in order to obtain an accurate approximation of AATOM by surrogate modelling. The schematic representation is given in Figure 3. Configuring the AATOM model is the first step of the procedure. The important model parameters and indicators are identified and the model configuration is determined. The specifications of the AATOM model are further explained in this section. The well-defined AATOM can subsequently be used for simulation in order to generate simulation data for training and validating the surrogate model. Furthermore, an intermediate step, preceding the surrogate model training, is implemented. The simulation data is pre-processed with the result that the surrogate model can achieve better performances. It consists of defining the relevant parameter values and re-sampling the simulation data to better train the surrogate model. Pre-processing is included after the first iteration of generating the surrogate model. The following step is training the surrogate model. Training is dependent on the surrogate algorithm, the Random Forest. The training data-set is provided from the simulation data of the second step. The simulation data is randomly divided for two thirds in a training data-set and one third in a validation data-set [Kleijnen and Sargent, 2000]. The fifth methodology step, the evaluation of the surrogate model, is subdivided into the validation of the surrogate model and a scenario-based evaluation. The validation is a comparison between the simulation data obtained from the AATOM model and the prediction results obtained from the trained surrogate model. The evaluation scenarios represent different terminal operation conditions w.r.t the passenger demand and flight schedule. For the evaluation, the scenario simulations of the AATOM model are compared with the surrogate model predictions on the same scenarios. Referring to the research objectives in Section 1, the methodology would allow preserving the important emergent properties through an accurate approximation of the model performance indicators. The evaluation step highlights the surrogate model's ability to preserve the AATOM model properties (indicators). The last step of the methodology is the comparison of the two surrogate models, both explained in Section 2. The Random Forest model is developed in this study with the given methodology. The Artificial Neural Network model is only used as a model for comparison with the developed Random Forest model. The models' accuracy and measure of variable importance are compared in order to identify relationships between the model parameters and indicators. All the different methodology steps are further elaborated in this Section.



Step 1: AATOM configuration
Identification of model parameters and indicators
Terminal configuration
Step 2: Simulation
Set simulation settings
Generation of simulation data
Step 3: Pre-processing
Definition of relevance function
Re-sampling training data
Step 4: Training surrogate model
Determine training data-set
Training Random Forest
Step 5: Evaluation
Step 5.1: Validation
Determine validation data-set
Comparison surrogate model and validation data
Step 5.2: Scenario evaluation
Definition of scenarios
Simulation of scenarios
Comparison simulation and surrogate model comparisons
Step 6: Surrogate model comparison
Accuracy evaluation
Variable importance comparison

Figure 3: The schematic representation of the methodology for obtaining a trained surrogate model, based on AATOM.

3.1 AATOM & Simulation

The AATOM has a modular architecture, as mentioned in Section 2, which requires a specific configuration for simulating airport terminal operations. In Figure 4, the chosen configuration for conducting the surrogate model generation is visually represented. The model consists of a check-in and security checkpoint with each having a queuing system in place. Moreover, the gate area is included in the model. The three systems (check-in, checkpoint and gate) are marked in Figure 4. Crucial for further steps in the surrogate modeling procedure are the defined model parameters and indicators. Both are given in Table 1. As can be seen in Table 1, parameters have been chosen such that both the check-in (n_{desks}) and security checkpoint (n_{lanes}, $n_{dropoff}$, $n_{collect}$) are regarded as the inputs of the model. The remaining input parameters in Table 1 are representing a one hour flight schedule, with the first time slot scheduled 400 seconds after the start of the scheduling hour and the last slot scheduled 3400 seconds after the start. Each slot is assigned either zero (if no flight is scheduled) or the number of passengers for the scheduled flight. Furthermore, regarding the indicators given in Table 1, both queuing time and throughput for security checkpoint and check-in are included. The last two indicators, the time to gate and the number of missed flights are more holistic, giving indications on the entire terminal.

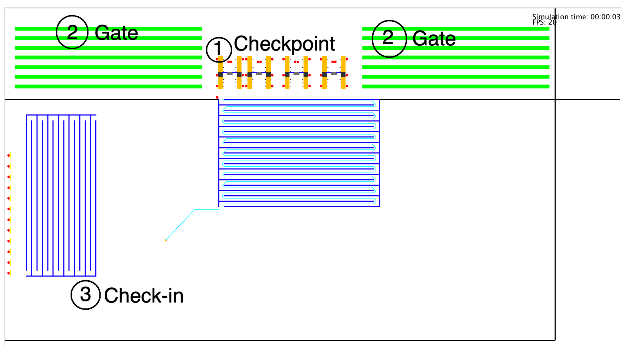


Figure 4: The AATOM lay-out inlcuding the check-in, security checkpoint and gate area (adapted from [Janssen, 2020]).

Hence, with all parameter specifications defined, the AATOM can be used for simulation purposes. In this research, the number of simulations needed for obtaining stable results is around 500, based on the coefficient

Table 1: The defined model parameters and indicators, according to the AATOM architecture, chosen for simulation purposes.

Parameters	Description	Unit
n_{lanes}	number of lanes at the security checkpoint	-
$n_{dropoff}$	number of luggage drop-off points	-
$n_{collect}$	number of luggage collection points	-
n_{desks}	number of check-in desks	-
$slot_{400}$	schedule slot at time 400(sec)	[s]
$slot_{1000}$	schedule slot at time 1000(sec)	[s]
$slot_{1600}$	schedule slot at time 1600(sec)	[s]
$slot_{2200}$	schedule slot at time 2200(sec)	[s]
$slot_{2800}$	schedule slot at time 2800(sec)	[s]
$slot_{3400}$	schedule slot at time 3400(sec)	[s]
Indicators		
$scQueue_{avg}$	average security checkpoint queuing time	[s]
$checkinQueue_{avg}$	average check-in queuing time	[s]
$throughput_{checkin}$	number of passenger that passed the check-in	-
$throughput_{sc}$	number of passenger that passed the security checkpoint	-
$TimeToGate_{avg}$	average time to the gate	[s]
$n_{missedFlights}$	number of missed flights	-

of variation of the indicators. The number of simulations are determined through the coefficient of variation, given in the Supporting Work. However, with the limited computational capacity, not every data point has been simulated 500 times. Nevertheless, for both the training and validation data-set, a relative large pool of sample points is drawn from simulation using the Latin Hypercube sampling method [Kleijnen et al., 2005].

3.2 Training & Validation

From the simulations, two data-sets are obtained, one for training and one for validating the surrogate model. The methodology includes prior to the training, however only considered after a first evaluation of the surrogate model, an additional pre-processing step. The AATOM, presented earlier in this Section, consists of multiple performance indicators related to distinct aspects of the terminal operations. The surrogate model, given a combination of input values, is required to predict the values of the target variables. However, the accuracy of the surrogate model may differ for the different target variables. It is possible that the surrogate is unable to achieve the validity criteria for one or multiple target variables. There exists different strategies for coping with it [Branco et al., 2015, Torgo et al., 2015]. The stra-

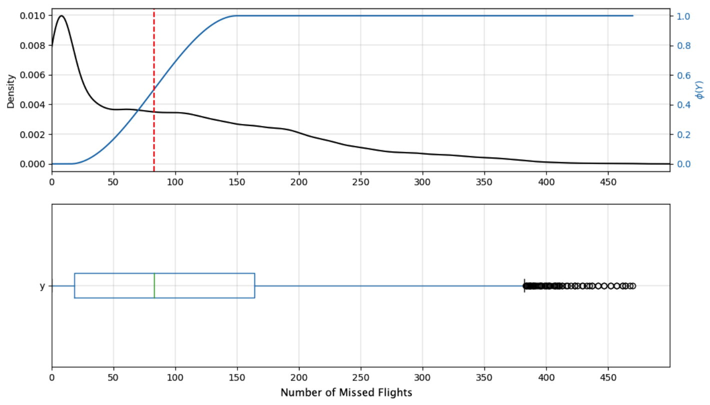


Figure 5: The relevance function and mapping of relevance for the $n_{missedFlights}$.

tegy used in this research, for its simplicity
of implementation and application, is the
utility-based regression and re-sampling approach. The relevance of the target variable values is identified through a relevance function, represented in Figure 5. The lower plot depicts the box-plot of the $n_{missedFlights}$ and the upper plot illustrates the mapping of the $n_{missedFlights}$ values on a scale from zero to one. Subsequently, the irrelevant data is under-sampled, redistributing the simulation data. The pre-processing method is further explained in the Supporting Work.

In this research, the surrogate model is generated with the Random Forest algorithm (elaborated in Section 2). The training data points consist of value combination of inputs and outputs as defined in Table 1. For the Random Forest model, several parameters need to be determined prior to the training. The parameters are called hyperparameters and are described in Table 2[Probst et al., 2019, Boulesteix et al., 2012, Díaz-Uriarte and Alvarez de Andrés, 2006].

Table 2: The hyperparameters of the Random Forest algorithm.

Hyperparameters	Description
mtry	number of drawn candidate variables at each split
samplesize	number of sample points drawn for each tree
replacement	draw sample points with or without replacement
nodesize	minimum number of points in the terminal node
ntree	number of trees in the RF

Given the hyperparameters in Table 2, several methods exist for determining each parameter value. The most simple is to base the values on expert domain knowledge. However, in the context of the research, no previous knowledge on the use of Random Forest approximations of AATOM exist. Hence, hyperparameter tuning algorithms can be applied for choosing the optimal hyperparameter values . For the purpose of the research, the Bayesian optimization algorithm has been chosen, proven to be reliable and efficiently applied in previous studies [Bergstra et al., 2011, Wu et al., 2019]. Accordingly, the results on prediction accuracy obtained by tuning the hyperparameters and by using a default set of hyperparameter values are compared. If an improvement is observed by using the optimization method then the optimized set is chosen for the hyperparameters, or else the default set is chosen. Further details about the Bayesian optimization and hyperparameter tuning are given in the Supporting Work.
Lastly, subsequent to training the surrogate model, validating the model on a determined measure is required. Ideally, real-world data is used and compared with the predictions made by the surrogate model on the same conditions (inputs). However, no real-world data is available for comparison on the AATOM described earlier. Hence, the AATOM is validated on a validation set obtained by simulating the AATOM. The results of the AATOM on the validation set are compared with the outputs from the validation set on the Mean Absolute Percentage Error (MAPE). The measure is the absolute difference between the prediction of the surrogate model and the AATOM simulation output on the same input values, given in percentage form. Moreover, the criteria of validation of AATOM is hard to determine. From literature [Kleijnen and Sargent, 2000], Random Forest surrogate models often achieve accuracies of more than 90%. Given that the nature of problems differ from the problem at hand (approximating the AATOM), a "softer" threshold of 80% is established for considering the obtained surrogate model as an acceptable approximation of the AATOM.

3.3 Evaluation

The research objective, mentioned in Section 1, underlines the importance of preserving the model properties for a model approximation. Hence, the surrogate model obtained in the research is evaluated on its accuracy of approximation and its generalisation. The validation, explained in the previous subsection, is evaluating the surrogate model on prediction accuracy. A second evaluation is included for evaluating the generalisation ability of the surrogate model. Both evaluations are given below:

- **Validation**: The surrogate model, obtained after training on a given data-set, is evaluated by comparison on a data-set unknown to the model prior to validation. The validation reveals the ability of the model to make accurate predictions on unknown data, hence investigating whether the surrogate model is an accurate approximation.
- **Scenarios**: An additional evaluation, next to the validation, is considered. It consists of a set of scenarios representing realistic airport terminal conditions. The main purpose, of using these scenarios as an evaluation approach on the surrogate model, is to determine the degree of generalisation of the surrogate model or i.e. the applicability of the model.

The scenario evaluation consists of a set of scenarios that represent real-world airport terminal cases. Scenarios are given in Table 3. The scenario is a combination of a set of resources, related to the check-in and checkpoint, and a given (one hour) flight schedule. The parameters for the resource allocation and flights schedule are the same parmaters as explained previously in Table 1.

Table 3: The set of scenarios for fixed resource allocation for the different flight schedules, with four resource variables (n_{lanes}, n_{drop}, $n_{collect}$ and $n_{checkin}$) and six time slots (from 400 to 3400) .

ID	Resource Allocation				Flight Schedule					
	n_{lanes}	n_{drop}	$n_{collect}$	$n_{checkin}$	**400**	**1000**	**1600**	**2200**	**2800**	**3400**
LOW	3	3	3	7	186			186		
MED	5	3	3	12	197		142	197		164
HIGH	7	3	3	19	186	197	186	142	197	197

Hence, three different scenarios are presented in Table 3 representing three different levels of passenger demand: low demand (LOW), medium demand (MED) and high demand (HIGH). The levels are based on a regular flight schedule at Rotterdam-The Hague Airport (RTHA). In each scenario, the fixed resource allocation is based on prior research analyzing check-in and checkpoint systems in the context of airport security [Janssen et al., 2019, Janssen et al., 2020]. One security checkpoint lane has an approximate capability of 160 passenger per hour and one check-in desk is able to process on average 60 passengers per hour.
The flight schedules, presented in Table 3, are derived from the flight schedule of RTHA, having on a peak hour an average of six flights scheduled. The representation is an hour of time slots where the value indicates the number of passengers scheduled for that time slot (empty if no flight is scheduled). Moreover, the number of passengers correspond to the Boeing 737 and 738 that are common aircraft at RTHA.

4 Results

In this Section results are presented, using the methodology described in Section 3 to generate the surrogate model. The parameter ranges for training the surrogate model are given in Table 4. The range of parameter values are based on the limitation of simulation and standard values for RTHA. The $n_{flights}$ are divided into time of scheduling and number of passengers. The former is explained in Section 3 and the latter can take the following values: [0, 142, 153, 164, 175, 186, 197]. Both time slots are combined in an array, e.g. [[1600, 142], [3400, 142]]. The value ranges for the other parameters are determined by the limitation of the simulation model.

Table 4: Value ranges of the AATOM parameters for training.

AATOM parameter	Range
n_{lanes}	[1,2,3,4,6,7,8]
$n_{dropoff}$	[1,2,3]
$n_{collect}$	[1,2,3]
n_{desks}	[1,2,3,4,5,6,7,8,9,10,11,12]
$n_{flights}$	[1,2,3,4,5,6]

The remainder of the Section is divided into an evaluation of the surrogate model (elaborated in Section 3) and an analysis of the input-output relations. The results also include a continuous comparison between the researched model (Random Forest model), obtained with the presented methodology, and the second comparison model (Artificial Neural Network model) described in Section 2.

4.1 Evaluation of the surrogate model

The evaluation results comprise validation using simulation data and scenarios, with the evaluation given in Table 5. It contains the time performance and accuracy measurements for both the Random Forest model and the Artificial Neural Network model. The accuracy measure with the Mean Absolute Percentage Error (MAPE), previously defined in Section 3.

Importantly, the optimized Random Forest model is not included in the results. The reason for this is that the performances obtained by applying the Bayesian optimization for hyperparameter tuning did not generate any significant improvement on one or multiple performance indicators. Presumably, the accuracy obtained without optimization reaches a ceiling, and applying hyperparameter optimization cannot increase the accuracy beyond the ceiling limit. Hence, there is no need to consider the optimised model for further evaluation or analysis.

Table 5: Comparison of performances between the Random Forest surrogate model with default hyperparameters, the optimized Random Forest surrogate model and the Artificial Neural Network surrogate model.

Performances	**RF**	**ANN**
Training time [sec]	3.82	12.54
Execution time [sec]	0.12	0.91
Accuracy		
$checkinQueue_{avg}$	91.66%	93.71%
$scQueue_{avg}$	96.02%	94.39%
$TimeToGate$	97.13%	98.01%
$n_{missedFlights}$	83.44%	80.12%
$throughput_{checkin}$	94.71%	91.29%
$throughput_{sc}$	94.44%	95.61%
Mean Accuracy	92.90%	93.87%

From Table 5, the Random Forest model has a low training and execution time. In addition, the surrogate model is significantly faster than the AATOM, with the original model having a simulation running time of approximately 5 minutes. Regarding the accuracy, the overall performance of the Random Forest model is attaining values of 90% and above, with an average accuracy of 92.90%. Moreover, the most accurate prediction, being the *TimeToGate*, is reaching 97.13%. It can be reasoned by considering that the TimeToGate is the indicator based on the most information, it is a measure combining both systems present in the model (security checkpoint and check-in). Nonetheless, the indicator for the number of missed flights is less accurately predicted, only achieving 83.44% accuracy. The reason for the lower accuracy is thoroughly discussed in Section 4.2.

Furthermore, comparing the Random Forest and Artificial Neural Network model, it can be seen that the Artificial Neural Network model is slower on both the training and execution time. However, the Artificial Neural Network model is still largely faster than the AATOM. On the accuracy, both models have similar performances. There are only small differences, with a mean accuracy slightly higher (+1%). The $TimeToGate$ is also the most accurate prediction for the Artificial Neural Network model. Lastly, the same distinction between the accuracy of the $n_{missedFlights}$ and the other indicators is present, only reaching 80.12%.

Successively, the second evaluation is given in Table 6, based on the scenarios from Section 3. The accuracy of the surrogate model is given for each indicator at every level of passenger demand. First, the model has on overall poor performance on the LOW scenario compared to the two other scenarios, with the largest differences on the accuracy for the two checkpoint indicators and the $n_{missedFlights}$. The most apparent reason is the lack of training data sampled around the region of the variable values defined in the LOW scenario. With the current methodology, it would require more simulation data generated for training by running again the AATOM simulation. Additionally, the $TimeToGate$ performance is a direct consequence of the poor performance on the checkpoint indicators, measuring the behaviour.

Furthermore, the surrogate model low accuracy on the $n_{missedFlights}$ is also visible in the results of the two other scenarios. However, there is a distinct increase in accuracy with increasing passenger demand, going 0% to 61.53%. There are possibly several causes, which are discussed in Section 4.2. However, it can be concluded from Table 6 that the lower values are harder to predict, mainly because lower values are present at lower demand of passengers and higher values at higher demand. Moreover, the surrogate model is trained on a wide

Table 6: The prediction for different scenarios from the trained RF surrogate model.

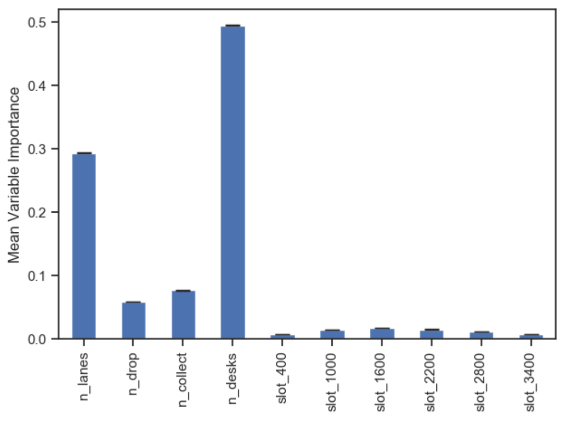
ID	$checkinQueue_{avg}$	$scQueue_{avg}$	$TimeToGate_{avg}$	n_{missed}	$throughput_{checkin}$	$throughput_{sc}$
LOW	89.13%	60.56%	80.01%	0%	94.1%	75.88%
MED	93.58%	88.62%	92.64%	8.81%	92.82%	98.21%
HIGH	88.17%	94.95%	95.38%	61.53%	91.45%	94.94%

range of cases with highly varying $n_{missedFlights}$ values. The nature of the phenomenon is further discussed at the end of the results Section.

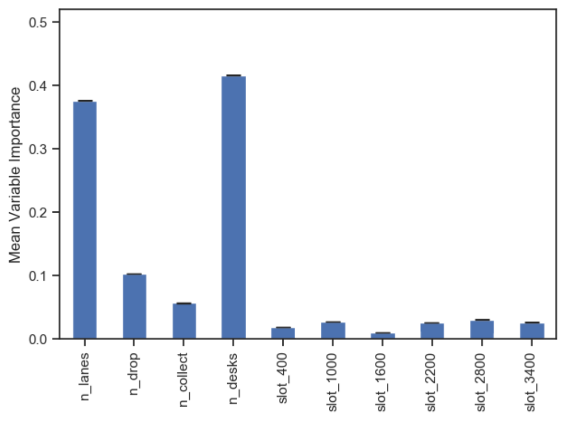
4.2 Variable Importance and $n_{missedFlights}$ Analysis

In addition to the evaluation of the surrogate model on running time and accuracy performances, in this section an analysis is given on the mean variable importance. Additional insight is given into the underlying relationships between the parameters and indicators of the original model (the AATOM).

The first analysis is the measure of importance of the different variables. The variable importance measure (VIM) is calculated on both surrogate models (Random Forest and Artificial Neural Network), given in Figure 6. The explanation and determination of the VIM is given in Section 2. In both plots the mean VIM is given for all model variables.



(a) RF model



(b) ANN model

Figure 6: The variable importance measure for the AATOM model parameters from the Defuault RF surrogate model and the ANN surrogate model.

As can be seen in Figure 6(a), there are two distinct variables, identified by VIM as the most important variables: the number of lanes of the security checkpoint and the number of desks at the check-in. Both variables are regulating the processing capacity of passengers at the two systems (checkpoint and check-in), therefore the indication from the VIM conforms with the representation of the two variables in the airport terminal operations. Moreover, from Figure 6(a), the four resource allocation variables (n_{lanes}, $n_{dropoff}$, $n_{collect}$, n_{desks}) are relatively more important than the flight schedule variables (time-slots). All indicators, except for the $n_{missedFlights}$, are measuring time performance through queueing or throughput. Hence, the resource allocation is more directly related to the time performance of the terminal than the flight schedule, dictating the check-in and checkpoint behaviour. The last observation in Figure 6(a) is the trend of the time-slot importance, with the first and last slot being the least important slots. Presumably, the importance of the middle slots stems from the effect of reducing the time between flights by adding more slots in the scheduling hour.

The VIM of the ANN model is given in Figure 6(b) for comparison purposes with the RF model. Multiple similarities are perceivable between the two models, with the most recognisable that the same two most important variables (n_{desks} and n_{lanes}) are identified with the VIM. Additionally, the resource allocation variables are prominently more important than the flight time-slots. Besides, there are several differences between the RF model's VIM and the ANN model's VIM. First, the importance measure difference between the two most important variables is lower. The lower difference is possibly due to the use of a different surrogate model, hence both models are not using the variable information in exactly the same manner to make predictions on the different target variables. Second, the ranking of the collect and drop-off location variables is reversed. Nonetheless, the order of magnitude is the similar for both variables in both models. Lastly, the trend of importance between the time-slots in the RF model is not present for the ANN model. Thus, the trend of the time-slot variables in the RF model is plausibly related to the choice of the surrogate model.

Furthermore, an analysis of the surrogate model accuracy (on the validation data-set) is given by systematically adding variables to the model. At each addition the predictions made by the surrogate model are compared with the validation data-set. The different mean accuracy levels obtained for both models are given in Table 7. Figure 7 visualizes the accuracy change by adding model variables.

From Table 7 and Figure 7, it can be seen that the four resource variables are supporting the RF model to make predictions with an averaged accuracy of 86.82%. Adding the remaining variables increases the mean accuracy by 6%. Hence, clearly observable in Figure 7, the resource allocation variables have the largest contribution to the increase in prediction accuracy of the surrogate model. Moreover, for the RF model, similar elements can be observed as in the VIM (from Figure 6(a)). The most important variable is in both cases the n_{desks}. Additionally, in VIM the resource allocation is considered more important than the flight schedule and the two extreme time-slots (first and last) are less important than the other slots. Lastly, for the RF model, there is the small reduction of accuracy after the addition of the first time-slot variable. However, the difference is sufficiently small (0.07%) to be neglected.

Furthermore, by comparing the RF and ANN model in Table 7, several similarities can be observed. First, the resource variables are sufficient for the ANN model to reach an accuracy of 89.77 %. The remaining variables, similar to the RF model, are only slightly increasing the prediction accuracy by 4%. Second, the most important variable is the n_{desks}, related to similar VIM (given in Figure 6(b)). Nonetheless, there is an observable difference between the contribution of the difference time-slot variables in the RF model and the ANN model. The differences coincide with the VIM of the variables in Figure 6.

Table 7: Mean accuracy of the surrogate models based on Random Forest and Artificial Neural Network with increasing numbers of variables, visually represented in Figure 7, for both the RF model and the ANN model.

ID	Parameters	Included Parameters	RF	ANN
1	$[n_{lanes}]$	[1]	62.53%	68.03%
2	$[n_{dropoff}]$	[1,2]	65.50%	72.24%
3	$[n_{collect}]$	[1,2,3]	70.97%	76.91%
4	$[n_{desks}]$	[1,2,3,4]	86.82%	89.77%
5	$[slot_{400}]$	[1,2,3,4,5]	86.75%	90.01%
6	$[slot_{1000}]$	[1,2,3,4,5,6]	87.74%	90.80%
7	$[slot_{1600}]$	[1,2,3,4,5,6,7]	89.89%	92.26%
8	$[slot_{2200}]$	[1,2,3,4,5,6,7,8]	91.32%	92.89%
9	$[slot_{2800}]$	[1,2,3,4,5,6,7,8,9]	92.38%	93.15%
10	$[slot_{3400}]$	[1,2,3,4,5,6,7,8,9,10]	92.88%	93.76%

$n_{missedFlights}$

Mentioned earlier in the evaluation results, the $n_{missedFlights}$ is the one target variable for which the surrogate model illustrates difficulties from the surrogate model to make accurate predictions. The accuracy is lower for the specific scenarios than for an evaluation on the validation data, overall the accuracy on the $n_{missedFlights}$ is lower than the other indicators.

In essence, the nature of the $n_{missedFlights}$ is different from all other indicators chosen for the AATOM simulation. All other indicators are measuring the efficiency in time or passenger count of the check-in and/or checkpoint. The $n_{missedFlights}$, however, is not directly bounded two one of the systems. The target variable is more evenly dependent on all the operational elements in the terminal. This can be observed in the VIM for predicting the $n_{missedFlights}$ and the incremental accuracy by variable addition, given in Figure 8.

According to Figure 8(a), the n_{lanes} is the most important variable. The average time to check a passenger at a checkpoint lane is higher than for the check-in, mentioned in Section 3. Hence, the number of lanes present at the checkpoint largely determine the number of passengers that can arrive on time for the flight. Furthermore, from Figure 8(a), the importance of all remaining variables is approximately evenly distributed. Thus, it indicates the necessity of all variable information to achieve adequate accuracy (above 80%) on the

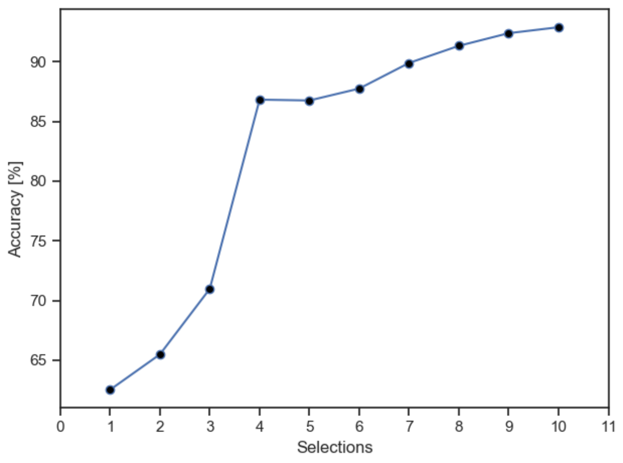
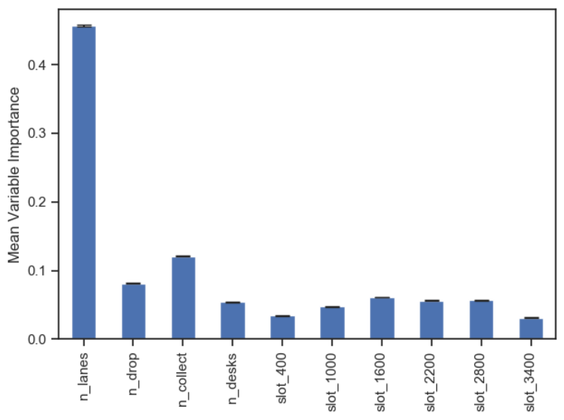
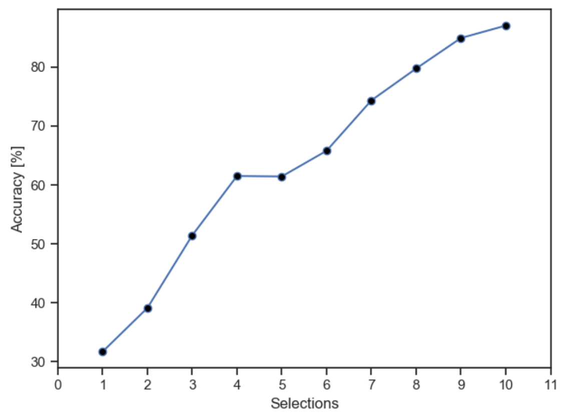


Figure 7: The model accuracy with addition of variables from the RF surrogate model.



(a) VIM



(b) Accuracy

Figure 8: The variable importance and model accuracy with addition of variables from the RF surrogate model for predicting the number of missed flights.

validation data. This can also be seen from Figure 8(b). The figure depicts the change in accuracy by step-wise addition of the variables. In contrast with the same plot from Figure 7, the accuracy is linearly increasing with every variable addition. Thus, this supports that large amount of accurate information about model variables is needed to predict the behaviour of the terminal on the number of missed flights.

Moreover, referring to the accuracy obtained for the scenarios, the surrogate model is predicting less accurately the smaller number of missed flights. The chosen measure for accuracy (MAPE) determines the absolute differences between the prediction of the surrogate model and the simulation result. Hence, for smaller numbers of $n_{missedFlights}$ the difference in percentage are larger, for example predicting 4 instead of 6 reveals an inaccuracy of 40%. Second, the number of missed flights occurring over the flight hour schedule is a rare event, especially rare for the lower values. The infrequent occurrence complicates the task of predicting this indicator. Last, the range of values for the $n_{missedFlights}$ is wide, visualised in Figure 9. Hence, it is more complex for the surrogate model to achieve high accuracy on the whole spectrum.

In addition, Figure 9 indicates the imbalanced distribution of the training data for the $n_{missedFlights}$. However, utility-based regression and the method for re-sampling, explained in Section 3, did not improve the accuracy of the surrogate model. The method has been applied on re-sampling the data to under-sample the higher values of the number of missed flights. Hence, it emphasizes that the current information contained in the training data is not sufficient for the surrogate model to attain highly accurate predictions on the $n_{missedFlights}$.

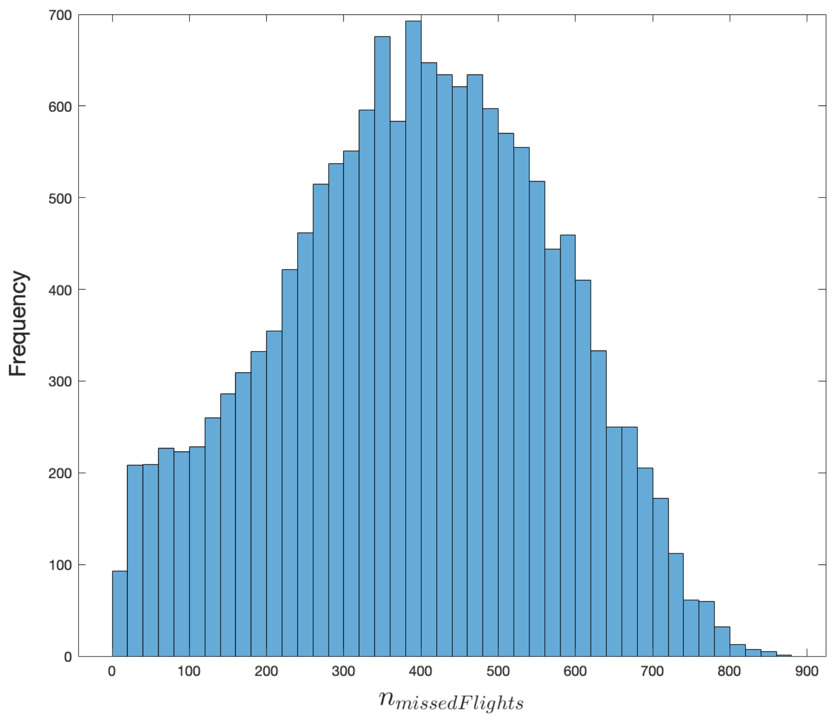


Figure 9: Distribution of the $n_{missedFlights}$ from the training data-set, with the number of missed flights over the number of occurrences in the data-set.

5 Discussion

First, the obtained results illustrate the approximation ability of the surrogate model. The model achieves high overall accuracy in both the validation and the scenario evaluation. However, the evaluation of the surrogate model is only considering a single accuracy measure, containing the measure of absolute difference between prediction and simulation result. Hence, only this difference is studied and it has shown its limitations when considering the poorly performing model on predicting the $n_{missedFlights}$. Other metrics could be considered, like the F1 score (for regression) or the Root Mean Squared Error.

Second, regarding the $n_{missedFlights}$, the proposed methodology requires some adaptations to cope with the target variable. The results show the difficulty to make accurate predictions on the smaller values. The main reasons for the lower prediction accuracy are the nature of the $n_{missedFlights}$ being the only indicator not directly measuring time performance, current information provided by the model variables is not complete and the wide range of possible values for the $n_{missedFlights}$. Thus, the current framework of generating the data and training the surrogate model is not sufficient to improve on the $n_{missedFlights}$. For the difficulty of predicting lower values, a solution could be to generate separate models for different conditions, having for example a trained surrogate model on conditions with low $n_{missedFlights}$. However, the issue with separate models is the risk of covering only a narrow range of conditions. Another option could be to use an adaptive method for generating the surrogate model, taking into account the model response. The adaptive method can include an adaptive sampling technique. Regarding the different nature of the $n_{missedFlights}$, additional model variables can bring different information for the surrogate model to support its predictions on a wider collection of inputs.

Furthermore, the comparison between the RF model and the ANN model reveals some similarities of results on both the evaluation and the model input-output relationships. The accuracy on the different indicator predictions is similar. Additionally, the problem of the $n_{missedFlights}$ is also present for the ANN model. Hence, the comparison of both surrogate models indicates similar assets and difficulties. Regarding the variables, the analysis of the ANN model identified the same most important variables and an equal difference in importance between the resource allocation and the flight schedule.

The variable importance results expose the ranking of importance of the different variables and their contribution to prediction accuracy, with the resource allocation being more important than the flight schedule. However, the importance measure and the accuracy contribution cannot capture the individual impact of every model variable. Moreover, the inter-dependency between the resource variables and the time-slots is not represented with the given analysis in the results. Hence, the surrogate model makes complex relations between the model variables and indicators that are only partially depicted in the given results.

Last, the application of the surrogate model on specific (real-world) scenarios for evaluation illustrates the possibility of the surrogate model as being part of a larger framework for future research. The surrogate model can serve several purposes, from validation and calibration to optimization. The trained surrogate model, the Random Forest model, can be considered for different applications, like terminal operation optimization challenges or AATOM calibrations.

6 Conclusion & Future Work

The research purpose was to study an approach for using an approximation of the Agent-based Airport Terminal Operation Model (AATOM) in order to make predictions on the behaviour of airport terminal operations. The study included two additional elements: preserving the properties of the model and getting an insight in the underlying dynamics of the AATOM. The method consists of an iterative process in which the AATOM generates simulation data for training and validation of the surrogate model. The surrogate model used in the method is the Random Forest (RF). The trained surrogate model was evaluated on the absolute difference between the predictions and the simulations results from the validation data-set. An additional evaluation was made on specific scenarios for different levels of passenger demands. Furthermore, an additional surrogate model is used for comparison, based on Artificial Neural Network (ANN). The ANN model is also evaluated on the validation data-set and exposed additional insight on the underlying relationships in the AATOM.

Regarding the approximation, the obtained surrogate model is able to make accurate predictions on all indicators except one, the number of missed flights. It achieves an average accuracy of 93%. For the number of missed flights, the accuracy is reaching 83%. Further analysis highlights the wide range of number of missed flights simulated and the difficulty for the surrogate to make accurate predictions on the lower numbers. Moreover, the evaluation on the scenarios has shown the generalisation capability of the surrogate model, with the same difficulty to make accurate predictions on the number of missed flights. The ANN model attain similar accuracy on all indicators and encounters the same phenomenon with the predictions on the number of missed flights. In essence, the results have shown through accuracy evaluation that the emergent properties, represented by the model indicators, are preserved by achieving acceptable accuracy levels.

Furthermore, the importance measure is determined on the different variables for both the RF model and the ANN model. They both highlight that the two most important variables are the number of desks at the check-in and the number of lanes. In addition, in both cases the resource allocation variables (related to the check-in and checkpoint) are considered more important than the variables of the flight schedule.

Several recommendations are suggested for future work on approximating the AATOM with a surrogate model. First, the process of generating the surrogate, based on the simulation data, can be further elaborated. Specifically, a more adaptive framework can be considered, in which the evaluation of the surrogate model is taken into consideration by the sampling method. Secondly, another method for processing the training data can be studied, more adequate with the complexity of predicting the number of missed flights. Lastly, additional metrics can be investigated for evaluating the surrogate model that could benefit the overall surrogate modelling framework.

References

[Behnamian et al., 2017] Behnamian, A., Millard, K., Banks, S. N., White, L., Richardson, M., and Pasher, J. (2017). A systematic approach for variable selection with random forests: achieving stable variable importance values. *IEEE Geoscience and Remote Sensing Letters*, 14(11):1988–1992.

[Bergstra et al., 2011] Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24:2546–2554.

[Biau and Scornet, 2016] Biau, G. and Scornet, E. (2016). A random forest guided tour. *Test*, 25(2):197–227.

[Boulesteix et al., 2012] Boulesteix, A., Janitza, S., Kruppa, J., and König, I. R. (2012). Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):493–507.

[Branco et al., 2015] Branco, P., Torgo, L., and Ribeiro, R. (2015). A survey of predictive modelling under imbalanced distributions. *arXiv preprint arXiv:1505.01658.*

[Díaz-Uriarte and Alvarez de Andrés, 2006] Díaz-Uriarte, R. and Alvarez de Andrés, S. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3.

[Edali and Yücel, 2019] Edali, M. and Yücel, G. (2019). Exploring the behavior space of agent-based simulation models using random forest metamodels and sequential sampling. *Simulation Modelling Practice and Theory*, 92:62–81.

[Garson, 1991] Garson, D. G. (1991). Interpreting neural network connection weights.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

[James et al., 2013] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning*.

[Janssen, 2020] Janssen, S. (2020). Capturing agents in security models: Agent-based security risk management using causal discovery.

[Janssen et al., 2019] Janssen, S., Sharpanskykh, A., and Curran, R. (2019). Agent-based modelling and analysis of security and efficiency in airport terminals. *Transportation research part C: emerging technologies*, 100:142–160.

[Janssen et al., 2020] Janssen, S., van der Sommen, R., Dilweg, A., and Sharpanskykh, A. (2020). Data-driven analysis of airport security checkpoint operations. *Aerospace*, 7(6):69.

[Kira and Rendell,] Kira, K. and Rendell, L. A. The feature selection problem: Traditional methods and a new algorithm. In *Aaai*, volume 2, pages 129–134.

[Kleijnen et al., 2005] Kleijnen, J. P., Sanchez, S. M., Lucas, T. W., and Cioppa, T. M. (2005). State-of-the-art review: a user's guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing*, 17(3):263–289.

[Kleijnen and Sargent, 2000] Kleijnen, J. P. C. and Sargent, R. G. (2000). A methodology for fitting and validating metamodels in simulation. *European Journal of Operational Research*, 120(1):14–29.

[Lamperti et al., 2018] Lamperti, F., Roventini, A., and Sani, A. (2018). Agent-based model calibration using machine learning surrogates. *Journal of Economic Dynamics and Control*, 90:366–389.

[Lee et al., 2015] Lee, J.-S., Filatova, T., Ligmann-Zielinska, A., Hassani-Mahmooei, B., Stonedahl, F., Lorscheid, I., Voinov, A., Polhill, J. G., Sun, Z., and Parker, D. C. (2015). The complexities of agent-based modeling output analysis. *Journal of Artificial Societies and Social Simulation*, 18(4):4.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Probst et al., 2019] Probst, P., Wright, M. N., and Boulesteix, A. (2019). Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1301.

[Sadeghyan, 2018] Sadeghyan, S. (2018). A new robust feature selection method using variance-based sensitivity analysis. *arXiv preprint arXiv:1804.05092*.

[Torgo et al., 2015] Torgo, L., Branco, P., Ribeiro, R. P., and Pfahringer, B. (2015). Resampling strategies for regression. *Expert Systems*, 32(3):465–476.

[van der Hoog, 2018] van der Hoog, S. (2018). Surrogate modelling in (and of) agent-based models: A prospectus. *Computational Economics*.

[Villa-Vialaneix et al., 2012] Villa-Vialaneix, N., Follador, M., Ratto, M., and Leip, A. (2012). A comparison of eight metamodeling techniques for the simulation of n2o fluxes and n leaching from corn crops. *Environmental Modelling Software*, 34:51–66.

[Welford, 1962] Welford, B. (1962). Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420.

[Wu et al., 2019] Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., and Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40.

[Zhang et al., 2020] Zhang, Y., Li, Z., and Zhang, Y. (2020). Validation and calibration of an agent-based model: A surrogate approach. *Discrete Dynamics in Nature and Society*, 2020:6946370.

II

Literature Study

previously graded under AE4020

1 Introduction

This report is a collection of literature in order to support the thesis research on surrogate modeling of Agent-based Airport Terminal Operation Models. Previous to the study, an extensive research has been led by (prior) Phd candidate Stef Janssen on the topic of modeling airport terminal operations for security analysis. The model developed by Stef, Agent-based Airport Terminal Operation Model (AATOM), is also the basis of this thesis research. The objective is to generate a surrogate model from AATOM. Currently, AATOM is used in a wider collaboration project between the TU Delft, Rotterdam - The Hague Airport and other parties. Both AATOM and the surrogate model could be used in many applications such as optimisation processes of terminal operations.

Regarding the structure of the report, in Chapter 2 the AATOM is explained in detail. The architecture of the agents composing AATOM is elaborated. In addition, the environment of the model is described. All the different environmental objects are depicted in detail. Last for Chapter 2, the parameters and indicators that can be defined in AATOM are listed and their relevance is discussed. These variables will also be essential in the simulations following the literature study.
Chapter 3 contains the theory on surrogate modeling. An overview of the different aspect of abstracting an existing model are explained: the sampling approach, the metamodeling technique, the validation procedure and the overall strategy. Additionally, an evaluation of the metamodeling techniques is given for making an initial choice on the technique used for the surrogate model. The understanding of all concepts of surrogate modeling is crucial for correctly applying the abstraction of AATOM in the later stages of the research.
In the last chapter, Chapter 4, the research objectives are set. The main research question is elaborated to fix the scope and the aims of the research. In addition, sub-questions are enunciated, helping to divide the objective into different sub-tasks. This categorisation is also used for presenting the thesis plan at the end of this report. All the different tasks needed to achieve the objective are listed in relation with deadlines to contain the project into a limited timeframe. After the literature study, the planning will be used for tracking the progress of the research until graduation.

2 Model: AATOM

The acronym AATOM stands for Agent-based Airport Terminal Operations Model [44]. The AATOM is an agent-based model (ABM) for simulation of the terminal of airports. Both the movements in a terminal and the operations are modeled. In this chapter the underlying model for the surrogate purposes of this research is explained. In section 2.1 the architecture used for the agents present in the simulation is described. Furthermore, the environment of the model is given in section 2.2. Last, the model parameters and indicators that can be used for surrogate modeling are given in section 2.3.

2.1. Agent architecture

The architecture of the agents is a three-layered architecture with three levels of abstraction, namely operational, tactical and strategical. The lowest level of abstraction is the operational layer. In this layer two distinct interactions of the agent with its environment (and other agents) are specified: the perception module and the actuation module. The perception module specifies the observation that the agent can observe. These observations are further used in the two other layers. The actuation module is the performing of the actions of the agent in the environment. Examples are the walking behavior of the agent in the environment or the communication of the agent with other agents.

The middle level layer is the tactical layer. Which interprets the observation, updates the belief of the agent, executes activities for preparing the actions and determines the routes for the agent. The interpretation module interprets the observation made in the lower level and passes them to the belief module. Often, it serves only as an intermediate step for passing on the observations. However, for some situations observations are combined for interpreting the situation (like being stuck at a location). The belief module can be seen as the memory of the agent. Four different types of information are saved in the belief module: the characteristics (info) of the agent, the interpretations made by the agent, the activities that the agent can execute and their current state and the agent's plan (when to do what). The belief is also exchanged with the strategical layer. Hence, the beliefs are necessary for other modules and beliefs are constantly updated. The two last modules contained in this layer are the activities and the navigation. Activities are based on the beliefs and the reasoning (from the higher level of abstraction). Activities are related to the physical area of execution and the activity's state. The navigation is the determination of a free-collision route for the agent in order to reach its target area. Observations are used for determining the route and activities determine the target area (end point) of the route.

The strategical layer is the highest level of abstraction. This layer contains a goal module and a reasoning module. Both exchange information with the belief module in order to update the beliefs in both the strategical and tactical layer. The goal module controls the goals of an agent. The goal for an agent is the aim of the agent to finish an activity before a certain time or before another activity. Last, the reasoning module analyses the current state of the agent, generates the agent's planning and makes decisions that are not covered in other modules. The planning is considered to be a sequence of activities that is based on the goals of the agent. For the decision-making, an example would be the choice of location where the activity is performed (which open security lane).

2.2. Environment

The agents from AATOM, with their architecture mentioned in section 2.1, interact with and within a specific environment. The environment in AATOM represents an airport terminal. Moreover, the environment consist of different objects given in Figure 2.1.

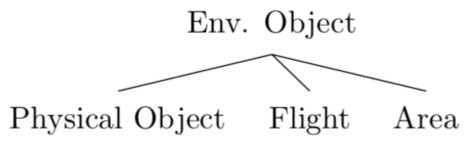


Figure 2.1: The different environment object types. [44]

From Figure 2.1, three environment objects can be distinguished: the areas, the flights and the physical objects. First, the areas are defined as two-dimensional polygons, every point in the environment is either inside or outside an area. With the area, the different airport terminal specific locations can be implemented such as the check-in and the security checkpoint. An overview of the different areas present in AATOM are given in Figure 2.2.

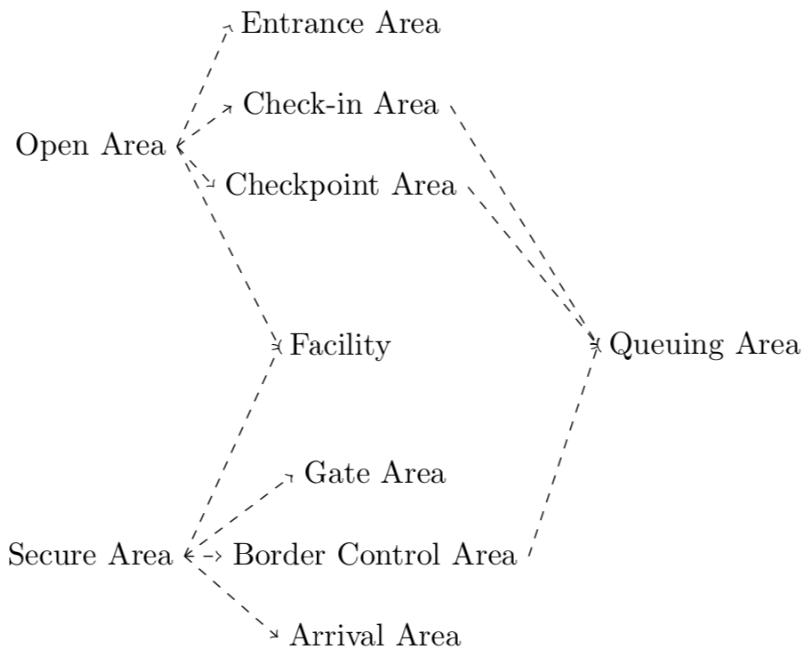


Figure 2.2: The 10 different areas present in AATOM. The arrows indicate that an area is located within another area. [44]

As can be seen in Figure 2.2, ten different areas are defined in AATOM. The Open Area and Secure Area divide areas between space where all human agents can be present (Open Area) and the areas where only human agents having passed the security checkpoint can go (Secure Area). Furthermore, Queuing areas include a queuing system where agents wait to be served, relevant for the check-in, checkpoint and border control.
Secondly, flights are considered as abstract environment objects within AATOM. They contain two parameters: the flight type and the flight time. The latter is the time at which a flight is scheduled. The former indicates whether the flight is a departure or an arrival. The flights are important objects for the functioning of the check-in system and related to the properties of passengers.
Last, the physical object, in contrast with the flights, are physical representations of the terminal. They represent different objects that are physically present in the airport terminal systems such as the luggage ,the seats

or the queue separator. The physical object is implemented as vector with three properties: transparency, blocking and sensor. The latter indicates whether or not the object is a sensor. Transparency indicates if the object blocks the observation of sensors (such as walls) and blocking indicates if another object or agent can be located at the same location as the object. These three type of objects form the entire environment of the airport terminal. In the next section, the parameters and indicators related to the environment and agents are described in detail.

2.3. Parameters and Indicators

In practice, the model presented in previous sections can be set up in various ways for simulation purposes, based on the source code [45]. The set up mainly relies on the set of model parameters and indicators defined for simulations. In this section, a first glance on the parameters and indicators of AATOM is given. In addition, the premises of the scope for the experiments set-up is suggested at the end of this section.

Parameters

There exists a large number of possible parameters for the AATOM simulation. An overview of possible parameters are given below. The parameters are categorized under three categories: the security checkpoint, the check-in the the flight.

- **Security Checkpoint**
  - Number of open security lanes
  - Number of drop-off points
  - Number of collection points
  - Distribution time for the different sensors (X-RAY, WTMD, ...)
- **Check-in**
  - Number of open check-in desks
  - Distribution time at desk
- **Flight specific**
  - Number of flights
  - Flight Times
  - Number of passengers for a scheduled flight
  - Passenger arrival distribution per flight

As can be seen in the list, the parameters related to the security checkpoint are closely related to the functioning of the lanes. The parameters define the scale and capacity of the security checkpoint. For the check-in desks, a small set of parameters define the check-in process. Lastly, the parameters associated to scheduled flights are various and will define the density of passengers in the simulation. Based on the given list of parameters, it can be concluded that their range of values are closely related, meaning that based on a certain schedule of flight a minimum of resources needs to be allocated in order to cope with the resulting passenger flow from the flights. Furthermore, parameters of a simulation model are defined in association with a set of indicators for measuring the performance of the model. Hence, in the following subsection the indicators of AATOM are given.

Indicators

Indicators of the model output the performance of the model during the simulation, focusing on a certain aspect of the model. Given below is a list of indicators that cover the different aspects of the model. The indicators are divided into system specific outputs (check-in or security checkpoint) and general outputs (overall performance).

- **System specific**
  - Average Queue Time: this is the average time that an agent (passenger) spends in a queue.
  - Queue Length: the length is determined by the number of passengers present in the queue.
  - Throughput: the rate of passengers passing the system
- **General**
  - **Missed Flights:** the number of passengers that missed a flight.
  - **Agent Number**: number of passengers in the terminal
  - **Number of employees:** Number of employees present in the terminal.
  - **Activity distribution:** the distribution of the number of passengers for certain activities, e.g. check-in or or gate.
  - **Time to Gate**: average time to the gate for all passengers.
  - **Distance**: average distance covered by passengers in the terminal

Regarding the division of indicators, additionally indicators can be seen as agent-related indicators (e.g. queue time of a passenger) and global outputs (e.g. number of passengers in terminal). Hence, the indicators and parameters are constraining the simulation configuration. However, in order to opt for certain parameters or indicators, the scope of the model needs to be defined beforehand.

Scope

The simulation model can capture only one part of the terminal or cover multiple parts. As a starting point, it is advised to limit the scope of the model to the **security checkpoint**. Past research around AATOM focused on the security of the terminal. Therefore, the parameters and indicators are also limited by the scope. A first draft for the starting parameters and indicators is given below:

- **Parameters**
  - Number of open security lanes
  - Number of drop-off points
  - Number of collection points
  - Distribution time for the different sensors (X-RAY, WTMD, ...)
- **Indicators**
  - Average Queue Time

Following the results based on this of parameter and indicators, an extended scope can be simulated with AATOM. In addition to the security checkpoint, the **check-in** and the **gate** can be included in the simulation. Moreover, for experimentation, specific **flight schedules** can be defined for evaluating the performance of the model and its ability to cope with "real-world" problems.

3 Model Abstraction

In the previous chapter, Chapter 2, the agent-based model (ABM) is presented. It models the operations and processes in a terminal airport. In general, analyzing the output of such an ABM, in order to better understand it, is relatively complex [35]. Understanding the ABM requires an exploration of its behavior under different settings, different input values. The challenge of analysis for ABMs is the nonlinearity of the interactions and the unknown distribution of its output.

The explorations for ABMs of the solution space, better called input-output exploration, is a broad topic [35]. There exists different methods for exploring this space. These methods include sampling techniques. This is a systematic approach to obtain parameters with certain statistical characteristics. Moreover, other techniques exists to explore the solution space, more recent techniques. One of them is machine learning [37] [2]. This enables deep analysis of the output data of the ABM. Data can be clustered or patterns can be detected by applying these techniques on the data set of input-output combinations. Moreover, there is an important specific approach to exploration of the solution space, sensitivity analysis.

In sensitivity analysis the question asked is which variables of the model are causing the most variation in the output [35]. The objective is an attempt to grasp a better understanding of the influence of variables and consequently investigate the possible reasons. The method is to analysis the response of the model by changing the values of the parameters.

There are various sensitivity analysis techniques. The simplest form of sensitivity analysis is the local sensitivity analysis, or also called one-parameter-at-the-time. As its name indicates, changes are applied to one model parameter at the time and the effect of these changes on the model response are analyzed. The disadvantage is that interactions between parameters are not captured with this method, it is difficult to obtain a holistic picture of the model. Global sensitivity analysis changes multiple variables and investigates their influence on the model response. However, the coverage of complex interdependencies is limited. Hence, another method, metamodeling addresses this issues in sensitivity analysis. The metamodel approximates the ABM, fitted and validated on model output, and facilitates the sensitivity analysis of the ABM by reducing the computational time. This approach is also the focus of this study in the following sections.

Furthermore, for the analysis of ABM output a very large number of combinations exists of input parameter that generate different model responses. Hence, the analysis requires collecting data from the model. The collection of data or sampling is an important aspect of the analysis [43]. Different approaches result to different model responses. Hence, as mentioned earlier, for the analyst the design of experiment is an interesting aspect of the analysis that is discussed in more detail in section 3.1.

For the remaining of the chapter the analysis of the model is further elaborated and the abstraction of this model with the use of metamodeling is explored. In section 3.1 the design of experiments are discussed for collecting the data of the model for further analysis. Section 3.2 depicts the use of metamodeling for ABMs. The metamodeling techniques are various, section 3.3 gives an overview of these techniques and their applications. Finally, section 3.5 discusses different strategies for approximating the ABM.

3.1. Experimental designs

In this section the design of experiment for analysis is further discussed. For analyst it is crucial to choose an appropriate design of experiment (DOE) for simulations [33]. Some clarification of terminology in DOE is helpful. The input or parameter of the model is called factor in DOE. These factors can take two or more values. The setting of the number of values for a factor is called factor level. Moreover, a combination of factor levels for all the factors is called a scenario or design point. In addition, for models of stochastic nature, the replicates are the simulation of same scenarios with different pseudo random numbers. Furthermore, the simulation code in DOE is viewed as a black box where inputs come in and outputs come out.

The choice of a particular design depends on the settings of the simulation. In Kleijnen et al. [33] an overview is given of the different designs based on two simulation aspects: the number of factors and the response surface. This overview can be seen in Figure 3.1. Hence, on the horizontal axis a spectrum is presented going going from simple to complex response surfaces. On the vertical axis the number of factors are represented.

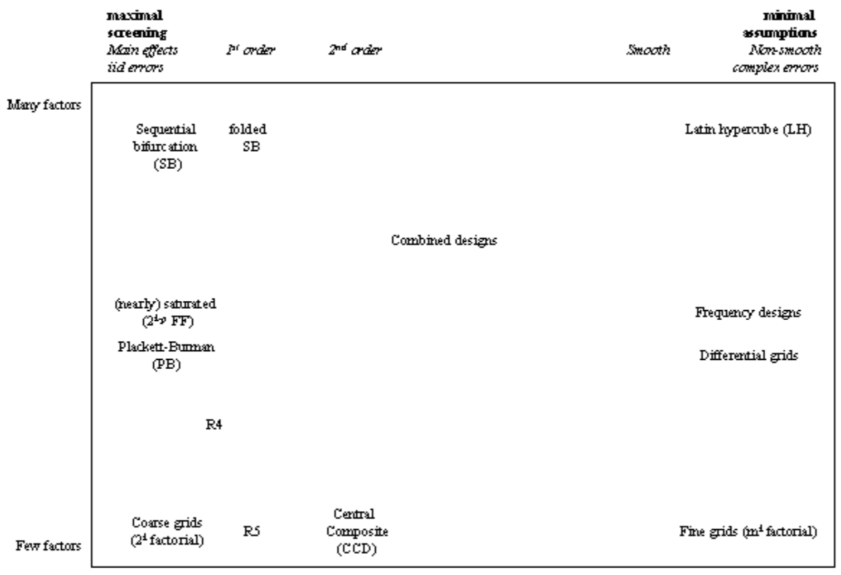


Figure 3.1: Recommended Designs According to the Number of Factors and System Complexity Assumptions. [33]

All sampling methods presented in Figure 3.1, however, are corresponding to the same category of experiments, namely the static design of experiments. Two categories can be distinguished within DOE: the static DOE and adaptive DOE. A list of common **static DOE** is given below. In addition to the distinction made in Figure 3.1, the distinction can also be made between classic and space-filling designs. [53].

- Classic designs
  - Random sampling [1]
  - Factorial design [33] [43]
  - Central composite design [33] [43]
  - Plackett-Burman design [40]
- Space filling designs
  - Latin Hypercube design [33] [43]
  - Orthogonal arrays [42]
  - Hammersley sequences [39]
  - Uniform design [19]
  - Maximum entropy design [12]
  - Minimax and Maximin design [27]

Moreover, Figure 3.2 depicts the categorisation of sampling methods. As can be seen, static methods can be either one-shot or sequential. However, some methods are sequential adaptive [38]. The main difference with static methods is chosen information from the metamodel included in the sampling process. Hence, adaptive sequential sampling leads to similar results as static methods with fewer points. However, two goals, proper to adaptive sampling, are necessary to balance in order to efficiently apply the sampling approach. The two key concepts are: **exploration** and **exploitation** [15].

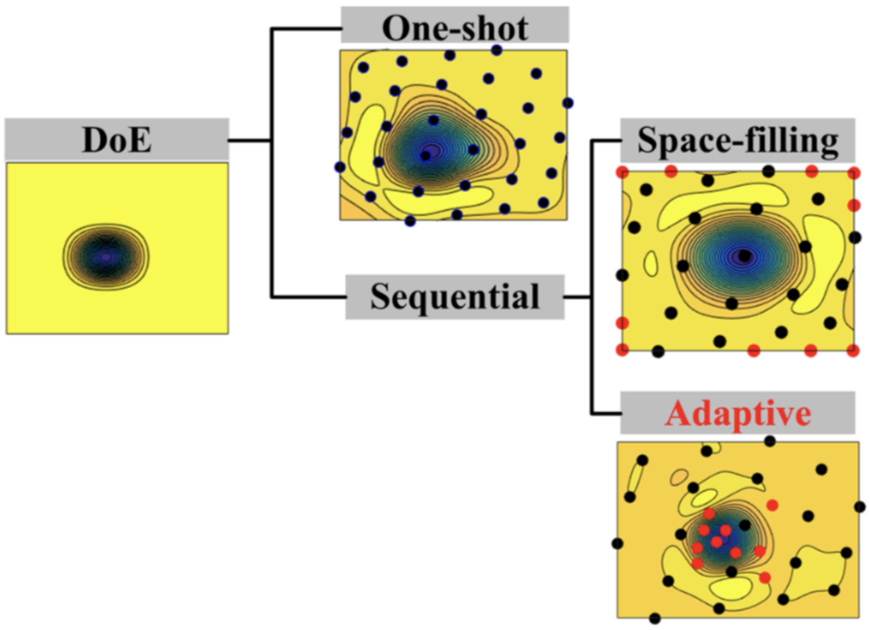


Figure 3.2: Sampling categories. [38]

The exploration and exploitation balance is best illustrated in Figure 3.3. Figure 3.3a depicts an initial evenly distributed sample. As can be seen, this approach is not certain of capturing any "interested" part of the underlying function. Figure 3.3b, however, is filling the parameter space and can therefore also capture the relevant function domain. The function is better explored in the case of Figure 3.3b. In contrast, Figure 3.3c exploits the relevant function domain at the cost of exploring the whole function domain. Thus, the adaptive methods represent a challenge on determining the adequate balance between exploring the parameter space and exploiting the most interesting parts.

3.1.1. Discussion

Both static and adaptive are applicable. However, the static method are simpler to implement within the surrogate modelling framework. A similar approach as for the model scope in Chapter 2 is considered for sampling the data. Initially static sampling is considered for obtaining data points in the parameter space. Only if, after evaluation of the surrogate, the training performance is considered insufficient then an adaptive method could be applied. The training performance is including both the training time and the accuracy obtained for the surrogate model. Furthermore, defining a sampling approach is not sufficient for establishing a surrogate modelling framework. In section 3.5 the strategy of generating the surrogate model is further elaborated.

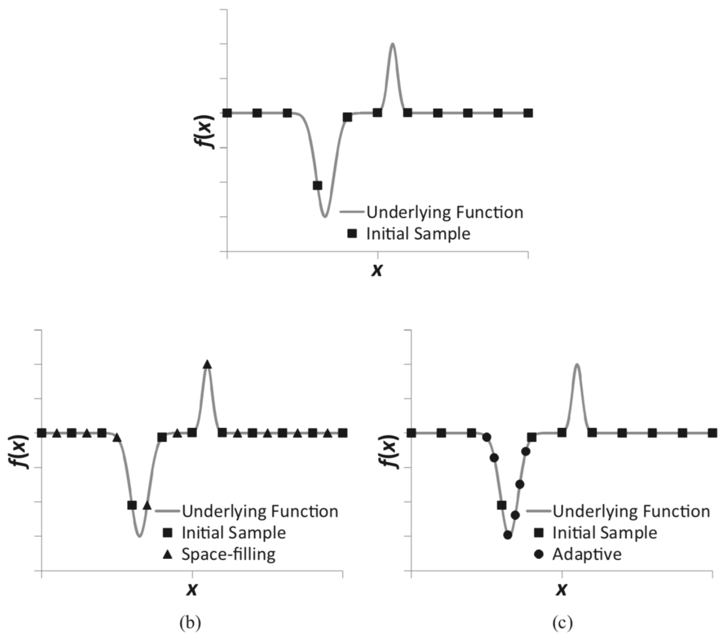


Figure 3.3: Exploration vs. exploitation problem: (a) underlying function with initial sample, (b) space-filling sampling, (c) adaptive sampling. [11]

3.2. ABM & Metamodeling

As mentioned in Chapter 2, the model that serves as basis for the surrogate modeling is the AATOM. The AATOM is an agent-based model for simulation purposes of terminals. Hence, a special attention is given on examples from literature that apply surrogate modeling on agent-based models. Several examples are given in the following paragraphs:

Surrogate Modelling in (and of) Agent-Based Models: A Prospectus[50] - The research focuses on the use of **agent-based models for economic** simulations. The main constraint in this domain is time. The agent-based model is very time consuming. Hence, **Artificial Neural Network (ANN)** and **Deep Learning** are applied for coping with this issue. Moreover, as indicated in the title, surrogate modeling is used in two ways: "in" and "of". The former relates on the use of a surrogate model to emulate the behaviour of agents within the model. The latter represents the use of a surrogate model to emulate the entire agent-based model. The paper underlines the interest of using neural networks for predicting the behaviour of agents in the model. Moreover, the research shows that ANN can be used to predict agent-based model behaviour in an efficient and accurate way. Last, the **complexity of the ANN** is noted as a challenging point for future applications of ANN in surrogate modeling of agent-based models.

Agent-based model calibration using machine learning surrogates[34] - Surrogate models can be used for different purposes. This paper considers the use of a surrogate model for **calibrating** the surrogate model, with a machine learning based surrogate. Calibration of agent-based models contains expensive steps, whether it is simply running model or identifying parameters of interest. Hence, the papers shows the interest of using machine learning surrogates for **diminishing the computational expenses**. Machine learning is an adequate technique for parameter space exploration and sensitivity analysis of the different parameters. The research shows that using a machine learning surrogate model is accurate and more efficient than the original model. Moreover, it also works for larger scales of agent-based models and performs better than classical techniques (Kriging).

Validation and Calibration of an Agent-Based Model: A Surrogate Approach[55] - This is another example of the application of a machine learning surrogate model for calibrating agent-based models. The approach consist of a supervised machine learning algorithm (CatBoost) combined with an iterative sampling procedure, reducing the number of samples needed to train the surrogate model. The results obtained in this paper show that using the approach delivers good performance results both on accuracy and time efficiency, again underlining that machine learning based surrogate can accurately predict the behaviour of an entire agent-based model.

Exploring the behavior space of agent-based simulation models using random forest metamodels and sequential sampling[18] - The research attempts on exploring the agent-based model based on a combined approach of sampling with a **random forest** surrogate model. There are different objectives for using a surrogate. In this research the objective is to get a better understanding of the behaviour of the agent-based model. The surrogate model used is the random forest and it is trained by the means of **sequentially sampling data points** from the original model simulation. An addition is made on the approach, a two-step procedure is implemented for determining whether to keep the target variables continuous or transform them into classifications. Hence, the paper shows that iterative sampling is more efficient than using random sampling for obtaining the data points. Moreover, the approach is shown to be both efficient and supporting a better **understanding of the original model**.

Comparative Analysis Of Metamodeling Techniques Based On An Agent-Based Supply Chain Model[17] - The objective of this research is to compare different techniques for obtaining a surrogate model. Four machine learning algorithms are considered: Decision Trees, Random Forest, K-Nearest Neighbour Regression and Support Vector Regression. The techniques are compared on their accuracy on the system-level output and the individual-level output. The study shows that all techniques are overall more efficient than the original agent-based model. For every approach the accuracy increases with increasing sample size. Furthermore, there is no single best technique for every conditions. The technique with the overall best accuracy performance is the Support Vector Regression.

All the examples shown above indicate the interesting elements of using a surrogate model for approximating the agent-based model. With the complexity associated to detailed agent-based models, surrogate models are an effective approach for predicting more efficiently the behaviour of the original model. In the following sections, the different techniques and approaches for obtaining surrogate models are elaborated.

3.3. Metamodeling techniques

In this section the metamodeling techniques for fitting the data are discussed. The goal is to use a statistical learning method on the training data for estimating the underlying function between the inputs and outputs. These methods can be categorized in two distinct categories, namely parametric and non-parametric methods [24].

The first category, parametric methods, is based on two steps. The first step is to make an assumption on the function's form, e.g. assuming the function is linear. In the following step the model is fitted on the training data by estimating the parameters of the previously defined function. Hence, as the name indicates, in parametric approaches the parameters are estimated instead of estimating the function. Non-parametric methods, on the other hand, do not make any assumptions on the functional form. They only try to estimate the output by getting close to the data points. The advantage of non-parametric over parametric methods is that it can cope with a broader range of forms for the estimated function. However, the issue with non-parametric methods is that by making no assumption about the function and therefore not reducing the number to a small number of parameters a larger amount of observations is needed than for parametric methods [24].

In the remainder of this section different metamodeling techniques are presented and discusses. It is not an exhaustive selection of techniques, the technqiues presented are the most frequently encountered in literature. At the end of this section an evaluation and comparison is given about all the techniques covered.

3.3.1. Linear regression

The simplest form of linear abstraction of a simulation model is the linear regression. This technique has been used extensively for approximating underlying models. An exmample of the linear regression model is the second-order regression model determined by [30],

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + e$$

where β is the regression parameter and x the factor. This is a one-factor model, meaning that it is a single input metamodel. However, linear regression metamodels can have multiple inputs and outputs in order to approximate a more complex underlying model. The general linear regression model can be formulated as follows [30],

$$y = X\beta + e$$

where y is the vector of predictors , X the matrix of regression variables and e the residuals. The criterion for estimating the regression parameter β is the criterion of Least squares. The parameter is chosen such that the sum of squared residuals is minimized[32]. The least squares estimator β' of the regression parameter β can be given by [29][30],

$$\beta' = \left(X^T X\right)^{-1} X^T w$$

where w is the output of the underlying simulation model. It is an average taken over the number of replicated simulation responses. Linear regression is a well-known tool for approximation. However, linear regression has its limitation towards more complex multivariate simulation models. An overview of its strength and weaknesses is given in Table 3.1.

Review of linear regression	
Strength	Weakness
• Widely used surrogate model • Effect of variables are easily identifiable	• Unsuitable for non-linear, multi-modal, multi-dimensional designs • Known properties of the I/O functions, like monotonicity, are not preserved.

Table 3.1: Strength and weaknesses of the linear regression method.

From Table 3.1 two clear advantages of using linear regression models are given. Being a more traditional metamodeling technique, it is a widely used technique under analysts. The second advantage relies on the effects of variables that can be deducted from their coefficients [21]. However, the linear regression is limited when looking at more complex non-linear designs [21]. Moreover, the known properties of the underlying function are not preserved when applying linear regression or Kriging [29]. In the following subsection an alternative technique to linear regression is discussed, namely Kriging.

3.3.2. Kriging

Kriging models mostly assume a single simulation output [32]. When the simulation has multiple outputs a univariate Kriging model is used for every output separately. The simplest type of Kriging is the Ordinary Kriging (will be called Kriging for the remainder of this section). The model assumes the following [32]:

$$w(d) = \mu + \delta(d) \tag{3.1}$$

where μ is the simulation output averaged over the experimental area, $\delta(d)$ is the additive noise that forms a stationary covariance process, d is the vector of input variables and w the output of the simulation. Furthermore, the following linear predictor is used in Kriging:

$$y(d) = \lambda' w$$

where the weights λ are not constant but decrease with the distance between the inputs that are predicted and the points in the design matrix. In contrast with linear regression, where the criterion for optimal weights is the sum of squared residuals, for Kriging it selects the Best Linear Unbiased Predictor which minimizes the mean square error.

The optimal weights depend on the covariances (correlations) of the simulation outputs in the Kriging model [32]. The correlations are determined by the distance between the inputs. With the assumption of the stationary covariance processes made in equation 3.1, several types of correlation function are possible to determine the correlation. Three popular correlation functions are the linear, exponential and Gaussian function. A more general popular function (that includes the exponential and Gaussian function function) is the following[32]:

$$\rho(h) = \prod_{j=1}^{k} exp\left[-\theta_j h_j^{p_j}\right]$$

where θ_j indicates the importance of the factor j, h the distance (mentioned earlier) for factor j and p_j is the smoothness of the function. In addition, other types of Kriging method exists, like Universal Kriging and Blind Kriging[21]. The adjustment made in Universal Kriging, compared to Ordinary Kriging, is the mean term μ in equation 3.1. The mean term is a function of the inputs and determined through polynomial regression. In Blind Kriging the mean term is determined through a more complex data-analytic procedure.

Review of Kriging	
Strength	Weakness
• Successfully applied on deterministic simulation models.	• Rare on random simulation models. • Weights depend on correlation function

Table 3.2: Strength and weaknesses of the Kriging method.

The Kriging method has its strengths and weaknesses. These are given in Table 3.2. It has been shown to be successful in applications on deterministic simulation models[29]. However, its use for random simulation models is less frequent than for deterministic models. Moreover, the weights that determine the predictor rely on the correlation function of the underlying model which is assumed in Kriging. However, this correlation function is unknown and makes the estimation of the parameters harder[32].

3.3.3. Radial Basis Function

The method for Radial Basis Functions (RBF) is a linear model, given by [41]:

$$f(x) = \sum_{j=1}^{k} w_j h_j(x)$$

where y is the estimated output, x is the input vector, k is the number of functions h_j and w_j are the weights. It is a combination of fixed k basis functions. The typical basis function is the Gaussian basis function[51]. The basis function is determined by the following:

$$h_j(x) = exp\left(-\frac{||x - c_j||^2}{r_j^2}\right)$$

where c_j is the center and r_j is the radius of the basis function. Other basis functions can be used instead for the model. These functions include the Cauchy, the multiquadratic or the inverse function[51]. In this model, using the radial basis function, there are the weights, the center and the radii that need be determined. The input matrix is used for the centers of the basis functions. The radii are equal to the span of the training set inputs. However, in order to avoid underfitting, radii are scaled and the Bayesian Information Criterion is used for that purpose. Moreover, weights are of the model are determined based on the sum squared error.

Review of Radial Basis Functions	
Strength	Weakness
• Can interpolate sample points. • Easy to construct.	• Need more training samples. • Sensitive to numerical noise.

Table 3.3: Strength and weaknesses of the RBF method.

Table 3.3 presents the overall strengths and weaknesses of RBF. In contrast with polynomial regression, RBF can interpolate sample points and is easy to construct[53]. However, RBF needs more samples for the same accuracy as other techniques[51]. In addition, it is found that RBF is more sensitive to numerical noise[53].

3.3.4. Multiple Adaptive Regression Splines

The Multi Adaptive Regression Splines method relies on a set of basis functions. The MARS model is determined by [22]:

$$\hat{y} = \sum_{i=1}^{k} c_i B_i(x)$$

where $\hat{y}$ is the estimated output, x is the input vector, k is the number of of basis functions B_i and c_i the weight factors. The non-linearity between input and output is taken into account by the hinge function as the basis function[51]. The basis function is a constant, a hinge function or the product of hinge functions (for interactions). The approach for approximating the response function is a two-step iterative approach: the forward selection step and the backward pruning step [51] [36]. In the forward selection basis functions are added based on the error reduction, functions with the largest reduction are added. During the backward selection, the model is pruned based on trade-off between the goodness-of-fit and the complexity of the model [51].

3.3.5. Artificial Neural Networks

An interesting metamodeling technique from the machine learning domain is Artifical Neural Networks (ANN). The basic idea behind ANN is to imitate the way neurons in human brains preform certain functions [23]. Similar to human brain, the ANN is composed of neurons that from the nodes of the network. Subsequently, these neurons can be connected to a neural network. The model of such a neuron is depicted in Figure 3.4. Three elements can be distinguished in this model: the synapses, the adder and the activation function. The synapses are the connecting links of the neuron with its input. Every synapse has assigned a certain weight. Furthermore, all the inputs received from the synapses are summed up in the second element of the neuron: the adder. The last element, the activation function, serves as a limit on the output of the neuron output. In addition, in Figure 3.4 a bias can be seen. Its function is to increase or decrease the input for the activation function.

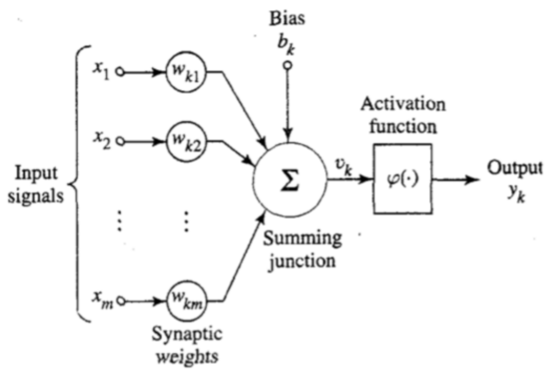


Figure 3.4: Non-linear model of a neuron. [23]

Hence, neurons can be connected to a network of neurons. The architecture of these networks differs on the way neurons are connected. Moreover, the design of the neural network architecture is also related to the choice of learning algorithm to train the ANN with. Figure 3.5, Figure 3.6, Figure 3.7 and Figure 3.8 all represent different types of neural network architectures.

There is a first distinction to make between feedforward networks and recurrent networks. Figure 3.5 and Figure 3.6 are both feedforward networks, the two other networks are recurrent. In a feedforward network the input signal is taken by the input layer and transmitted forward to the next layer until the output layer. The difference is that Figure 3.5 only contains an input layer and a layer of neurons whereas Figure 3.6 contains a hidden layer of neurons between the input layer and the the layer of output neurons. The additive value of a hidden layer is the ability to capture higher order statistics which is useful when having a large input layer.

Figure 3.7 and Figure 3.8 represent recurrent networks. The term recurrent is referring to networks where the output of a neuron is fed back into the input layer of the same network. The feedback loops have the benefit of enabling additional learning capabilities and impacting the performance of the network.

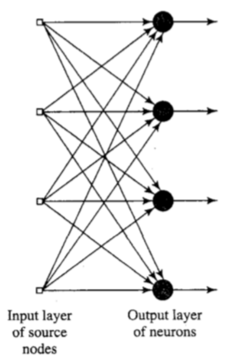


Figure 3.5: Feedforward network with a single layer of neutrons.[23]

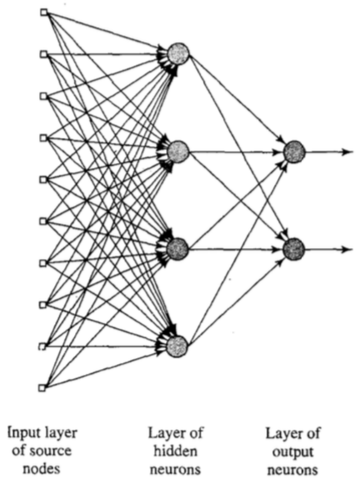


Figure 3.6: Fully connected feedforward network with one hidden layer and one output layer.[23]

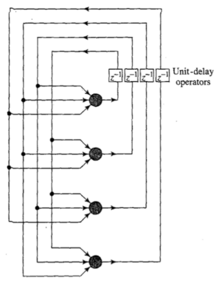


Figure 3.7: Recurrent network with no self-feedback loops and no hidden neurons.[23]

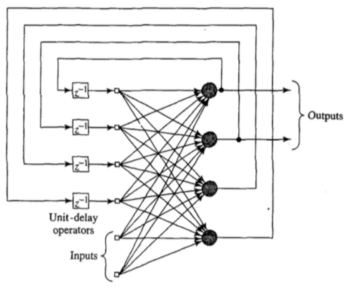


Figure 3.8: Recurrent network with hidden neurons.[23]

A first example of ANN for metamodeling is a study by ***Fonseca et al.*** [20].

The authors present in their study the use of ANN for metamodeling and provide guidelines for developing these metamodels. As mentioned earlier, for designing an ANN there are different architecture possibilities and learning algorithms. Regarding the architecture Fonseca et al. mention that multilayered, feedforward (non-linear) networks, also called Multilayer Perceptions (Maps), are commonly utilized for generalization purposes. These networks have been proven successful in a supervised training environment. Hence, the authors chose for this architecture combined with the back error propagation algorithm (BEEP)[13]. The BEEP works through a first generation of the output based on the given input vector. If the generated output differs from the target output then weights are changed in order to reduce this difference. The general rule for the weight change is given by the following equation[13],

$$\Delta_p w_{ij} = \eta(t_{pj} - o_{pj})i_{pi}$$

where $\Delta_p w_{ij}$ is the change to be made to the weight of the ith and jth element of the input/output pair p, (t_{pj} and o_{pj} are the target input and output if the jth element of the output pattern and i_{pi} is the ith element of the input pattern. In other words, for every training pattern (input/output) the weights of the neurons are adjusted to reduce the signal error.

Furthermore, the authors emphasize the importance of a methodological approach to develop ANN metamodels, providing general guidelines. Several points are stressed: sufficient input, smooth response (small change input results in small change in output), avoid overfitting. In addition, the authors suggest a step-by-step experimental approach to determine the numbers of neurons, input/output and hidden layers. The paper concludes on a promising future for the use of ANN metamodels given that they are time efficient and simple to adapt for new scenarios. It is regarded as an interesting contribution to the simplifying decision- making processes.

This is an early study on ANN metamodels for an emergency department simulation by ***Kilmer et al.*** [28].
In this paper an application of an ANN metamodel for simulating the emergency department of a hospital is presented. The underlying model simulates the department processes and allows for testing of the patient flow, staff, layout and equipment. Subsequently, the objective is to take more optimal strategies for the emergency department based on simulations. The role of the metamodel is to accurately approximate the simulation and be be more time efficient than the underlying model. The stochastic nature of the simulation causes queues and possible congestion if decisions are not taken quick enough.

Traditional metamodeling techniques are discarded in this study due to their limitation regarding the simulation domain. A more novel technique is chosen in this study, neural networks. The architecture of the network can be seen in Figure 3.9. The authors opted for two parallel neural networks, one that predicts the mean time while the other predicts the variance of that mean time. The are used in combination for obtaining the prediction intervals over the simulation domain in order to compare with the underlying model. As is depicted in Figure 3.9, both networks contain two hidden layers. No argumentation is given for the number of hidden layers and for the number of hidden neurons the authors mention determination by brief experimentation. Moreover, similar to Fonseca et al. [20], a feedforward, multilayered network is chosen and the same learning algorithm is used, namely back-propagation algorithm. Regarding the training, two methods were used, one where all replications are combined into the expected value and one where all replications are used individually. The network with the minimum mean absolute error was ultimately chosen.

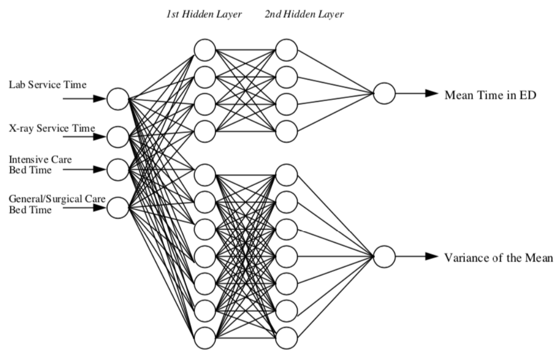


Figure 3.9: Architecture of the parallel neural metamodel. [28]

Hence, the authors reflect on the authors on their results obtained by indicating that the skewness (asymmetry) of the distribution output is not reflected through the metamodel. Moreover, the metamodel is deterministic, meaning that the stochastic nature of the simulation is lost. In addtion, changing the simulation by changing variables would require a new metamodel. Interesting points highlighted in the paper is that the metamodel could be updated by extra observation of the system. This training aspect is also considered by the authors as an important point for further research.

Here a combat model is used for studying different experimental designs , by ***Fasihul et al*** .[1].
The authors are using a combat model based on system dynamics as the underlying model. This type of model, based on a dynamic system, is rather complex and has a relatively high dimensionality. Hence, simulation is computation costly and the difficult to interpret. Therefore, a metamodel is suggested for coping with these simulation issues.

The metamodel is a feedforward neural network containing three layers. The design choices for every approach (design experiment) have been made through experimentation, the number of hidden neurons are determined through trial-and-error. The architecture of the network is determined with the minimum mean square error. The general development can be seen in Figure 3.10. A simple three step process: obtain data from original model, train ANN on this input/output dataset and test metamodel on new input data.

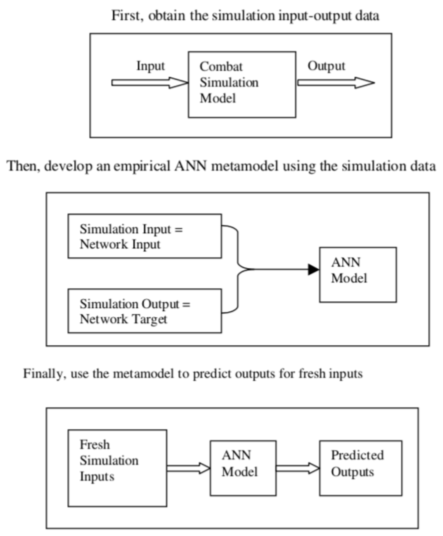

First, obtain the simulation input-output data
Input
Combat Simulation Model
Output
Then, develop an empirical ANN metamodel using the simulation data
Simulation Input = Network Input
Simulation Output = Network Target
ANN Model
Finally, use the metamodel to predict outputs for fresh inputs
Fresh Simulation Inputs
ANN Model
Predicted Outputs

Figure 3.10: Process of developing ANN metamodels from a combat simulation model. [1]

Furthermore, the ANN metamodels are developed with six different experimental design approaches, meaning that the training set for the ANN is obtained in six different ways. The six approaches are: full factorial design, random sampling design, central composite design, modified Latin Hypercube design and two of them supplemented with domain knowledge. The Latin Hypercube design is modified in order to counteract the randomness for covering the variable space. This is done by applying the MAXIMIN distance criterion, meaning that the minimum distance between points of the configuration is maximized.

The purpose of the study is to show the effect of sampling on the ANN metamodel accuracy. The findings are that the modified Latin Hypercube design supplemented by domain knowledge could be more effective and robust for the development of the metamodel. It obtains better predictions than the conventional approaches and random sampling. The recommendation from the authors is to use traditional experimental designs for relatively smooth response surfaces and avoid using them when more non-linear metamodels are required.

3.3.6. Decision trees & Random Forest

Considered as a simple method for interpreting the underlying model, the metamodeling techniques based on decision trees are interesting for decision support tools. The method is based on the analogy of a tree [24]. The objective of this method is to predict, based on observations, the region where this observations belong. The observations are split, reaching the terminal nodes (or leaves). These nodes are the regions. For all the observations that fall in the same region, the same prediction is made.

There is a distinction to make between regression and classification trees. Both are using the same decision tree method. However, the regression tree predicts a quantitative response where the classification tree predicts a qualitative response. Hence, for the regression tree, the prediction of a region is the mean responses of the observations in that region. In classification trees the prediction is determined by the most commonly occurring class in that region.

Figure 3.11 and Figure 3.12 depict two different ways to visualize the results from the method. Figure 3.11 is an upside-down tree representation where every split generates new branches in the tree until the leaves at the bottom. Figure 3.12 is another approach of visualizing the splits and regions, where the lines split the observations and form regions on the grid.

For the regression tree, the splits, that form the decision tree, are high-dimensional rectangles. The objective is to find the boxes for which the residual square error is minimum. However, due its large number of possibilities, the recursive binary splitting is used. At each step, the best split is made without considering future steps. This process is limited by a predefined stopping criterion. Important to note, similar to other statistical learning methods, there is a risk of overfitting the data. This can be avoided by pruning the tree. It is the process in which initially a large tree is grown and from this tree an adequate subtree is selected, the

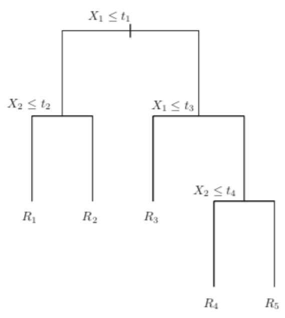


Figure 3.11: Result of the decision tree method represented as tree.[24]

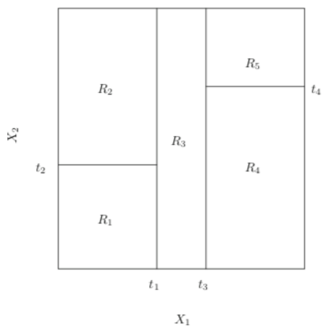


Figure 3.12: Result of the decision tree method represented in two dimensions.[24]

subtree with the smallest cross-validation error.

In classification trees, the splits and errors are determined differently. The binary split is based on the classification error rate, in contrast with the residual square error. The error is the fraction of observations that do not belong to the most common class.

Thus, decision trees are relatively easy to interpret and even with limited expertise in the topic some relations can be interpreted. The trees are also compatible with predicting qualitative responses without the need to use additional variables. However, there are some drawbacks with decision trees. The decision tree is not the most accurate statistical learning method. In addition, the tree can potentially be non-robust. Meaning that small changes in the data highly affect the final decision tree. Hence, using a single decision tree is not always beneficial. Therefore a more reliable solution is to use ensemble methods that make use of the decision trees, namely bagging or random forests.

Bagging or bootstrap aggregation relies on training sets obtained through the bootstrap sampling method. In essence, bootstrap is a statistical tool that produces multiple distinct samples from a single data set. The objective for using this tool is to reduce the variance of decision trees and increase their accuracy. This is done by generating B bootstrapped training data sets from the same population and applying the decision tree method on every training data sets. Hence, B decision trees or obtained from this approach and their predictions are averaged. The difference between regression and classification trees in this approach is the quantitative result that can be averaged for the regression tree where the result for the classification is averaged by means of the majority vote. Subsequently, the error of the bagged model can be determined with the fraction of observation that are not used by the tree. These are named Out-of-bag (OOB) observations. For each OOB observation a predictive response can be made and from these an overall OOB error can be computed. Moreover, one of the benefits of using a decision tree is the relative easiness to interpret the results. However when using many trees, like bagging and the two following ensemble methods, it becomes more difficult to interpret their results. In these situation an overall importance can be computed for all the variables, this is called the variable importance.

Another method, similar to bagging, is random forests. Similarly to bagging, random forests generate B distinct amount of trees on B bootstrapped training data sets. However, where it differs from bagging, is in the split made when building a tree. Instead of considering all predictors as possible split candidates, only a subset of predictors is considered and the candidate is randomly chosen from this subset. The goal with random forests, compared to bagging, is to reduce the correlation between the bagged trees and therefore reducing the variance when averaging the obtained bagged trees.

The individual trees are aggregated based on the majority vote[7]. The CART-split criterion is used in the construction of individual trees in order to determine the best split at each node by by optimizing the criterion. Generally, the Gini impurity is used for classification and the prediction squared error is used for regression[6]. The construction of the individual trees end the RF is schematized in Figure 3.13

Two important aspect of the RF algorithm can be distinguished in Figure 3.13: the tree growth with feature selection and the **Out-of-bag** (OOB) error. The first indicates that the feature selection during tree growth can play an important role. Similarly, the number of trees are also influencing the RF algorithm. Regarding the OOB error, it is an alternative to get a first error prediction by estimating the error of a tree on the data that has not been used to grow this tree (as mentioned for bagging). Lastly, an interesting feature of the RF is that this method does not **overfit** [14]. More nuanced, individual trees can overfit. However, this overfitting can

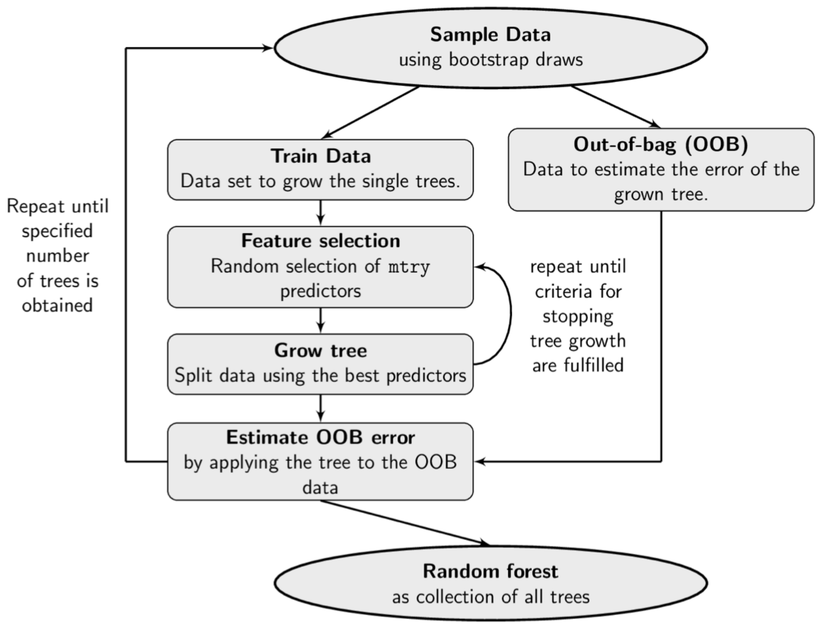

Sample Data
using bootstrap draws
Train Data
Data set to grow the single trees.
Out-of-bag (OOB)
Data to estimate the error of the grown tree.
Repeat until specified number of trees is obtained
Feature selection
Random selection of mtry predictors
repeat until criteria for stopping tree growth are fulfilled
Grow tree
Split data using the best predictors
Estimate OOB error
by applying the tree to the OOB data
Random forest
as collection of all trees

Figure 3.13: Random Forest Algorithm. [7]

be compensated by the number of trees and the feature selection for tree growth.

The third approach is boosting. Again, similar to bagging, boosting uses many trees to obtain a final prediction response. However, in contrast with bagging and random forests, boosting does not use bootstrap data sets. Instead, in boosting trees are generated in a sequential manner. For regression trees, every decision tree is fitted to residuals of the model. A similar approach is applied for classification trees, only more complex. In general, boosting is determined by three tuning parameters: the number of trees B, the shrinkage parameter λ and the number of splits for each tree d. If B is too large, overfitting the data is possible. Hence, cross-validation is used to determine B. The parameter λ controls the learning rate of boosting. Opting for a large B requires a small λ. The last parameter d is the number of splits for each tree, meaning that choosing a large d increases the complexity of the final boosted decision tree. Using small values for d can increase the interpretability of the final tree.

3.3.7. Evaluation

In this section, the techniques presented earlier are evaluated, leading to a unique choice for the surrogate model of AATOM. The choice is based on a comparison of the techniques. However, comparing them is challenging. The performance of a technique is strongly related to the nature of the problem at hand and the criteria of evaluation. Furthermore, existing comparisons in literature are not exhaustive, both on the criteria and on the techniques considered. Hence, an attempt is made to compare all the techniques presented in the previous section based on three criteria by cross-referencing five different papes [51] [36] [16] [26] [52]. The three criteria are the following:

- Time efficiency: the computational time of the surrogate model, for the construction (training) and the execution (prediction).
- Accuracy: the results obtained by the surrogate model compared with the results of the original model.
- Transparency: the difficulty of interpreting the relations of the inputs and outputs.

First, the time needed for training a model and making predictions with the trained model is an essential asset when using surrogate models. One of the objectives of substituting the original model with a surrogate is decreasing the computational time. Hence, time is one of the criteria of comparison.

Table 3.4 gives the comparison on time efficiency, for both the training and predictions. The cross-referencing for time comparison is not absolute due to the different sample sizes used in every research. As can be seen, PR is the fastest technique mainly due to its simplicity and ANN the slowest. ANN is mainly slower on the time needed to train the surrogate model. Overall all techniques are significantly faster than the original model on the execution (making predictions. Thus, using any technique for surrogate modeling.

1 (fastest)	2	3	4 (slowest)
• PR	• RBF • MARS	• RF • KR	• ANN

Table 3.4: Ranking of metamodeling techniques based on time efficiency.

Second, a primordial criteria for any surrogate model is the accuracy with which it can predict the behaviour of the original model given specific conditions (inputs). In the different papers, different metrics are used for evaluating the accuracy of the metamodeling techniques, making it challenging to cross-reference. Moreover, similar to time efficiency, different sample sizes are used in every research.
Table 3.5 depicts the comparison on accuracy of the different techniques. The larger sample sizes of the different research are considered for this comparison. RF achieves the best accuracy. PR is making less accurate results mainly due to its difficulty to model complex (non-linear) relations.

1 (most)	2	3	4 (least)
• RF	• ANN	• RBF • MARS	• PR • KR

Table 3.5: Ranking of metamodeling techniques based on accuracy.

Last, the transparency of the different models are evaluated. In contrast with time efficiency and accuracy, less studies consider also the transparency as an evaluation criteria for surrogate models. In addition, interpreting relations and importance factors is non-trivial. However, a comparison is given in Table 3.6 based on the available studies. As can be seen, PR is the most interpretable model. In PR the different coefficients directly indicate the importance of the different parameters. Moreover, RF is still given insights on the relations with its ability to determine variable importance. However, ANN is the least interpretable method. It is widely considered as a "black box" algorithm, where the relations between input and output are very difficult to interpret.

1 (most)	2	3	4	5 (least)
• PR	• RF	• MARS • RBF	• KR	• ANN

Table 3.6: Ranking of metamodeling techniques based on transparency.

Thus, based on the different comparisons given on the three criteria (time, accuracy and transparency), a choice is made on the technique to use for obtaining a surrogate model of AATOM. **RF** is chosen as the most appropriate technique. It is a relatively fast surrogate model, both on the training and the predictions. Furthermore, RF is very accurate, especially on larger sample sizes. Regarding transparency, RF enables to get an insight into relations of the inputs and outputs. Hence, RF is regarded as an adequate surrogate model for approximating AATOM while maintaining the AATOM properties.

3.4. Model validation

After fitting the metamodel with a specific technique on the training data, the model needs to be evaluated. This is called validation. Validation of the model is a challenging step, there is no unique method for validating the model[24]. Validation of the model includes measuring the performance of the model and obtaining the validation data set. The latter is possible to obtain in two ways: using the existing data (from fitting) or sampling additional data[39]. The additional data is generally obtained from using the same sampling technique as for the training data set. However, this option can be computationally expensive for certain simulation models [21] [39]. Hence, in these cases, it is more interesting to use the existing data. This is discussed in the next subsection. Subsequently, the performance measurements are discussed.

3.4.1. Cross-validation

When measuring the error of the metamodel, the training error or test error can determined[24]. The first is the error of the metamodel responses compared with the training data responses. The test error is determined on unseen data. However, in cross-validation the test error is estimated using the training data. The methods in cross-validation estimate the test error by "holding out" a subset of the training data from the fitting process [24]. These methods can be applied to both qualitative and quantitative responses. These are several techniques from cross-validation:

Validation Set

The validation set is a simple approach. The observations obtained from sampling are randomly divided into two sets, the training set and the validation set. Hence, this strategy is simple and easy ti implement. However, there are two main drawbacks [24]. The test error can highly vary due to the randomness of selecting the training and validation set. Moreover, by reducing the number of observations for fitting the model, the performance of the metamodel is also potentially lowered.

Leave-One-Out Cross-Validation

The Leave-One-Out Cross-Validation (LOOCV) is an technique the compensates the drawbacks from the validation set approach. For LOOCV, from the n observations obtained $n-1$ are used for fitting and one is used for estimating the test error. Only using one error estimate would make it very variable. Hence, this is repeated n times (for every observation), meaning that the the model is also fitted n times. The test error is averaged over the n estimates. The advantages of using LOOCV is that there is less bias than the validation set approach. The other advantage is that there is no randomness in LOOCV, the same result is always obtained. However, LOOCV is a very expensive technique as the model has to be fitted n times to the data.

k-Fold Cross-Validation

The k-fold Cross-Validation (k-fold CV) is similar to LOOCV. IN k-fold CV the data set is randomly divided in k groups (or folds) of approximately equal size. One group is used for validation and the remaining $k-1$ groups are used for training the model. This is repeated k times. Hence, if k is equal to n k-fold is identical to LOOCV. The advantage of k-fold CV over LOOCV is the computational reduction. by reducing the number of groups ($k < n$). Another, less obvious advantage over LOOCV is the lower variance. In LOOCV the n training data sets are almost identical, leading to highly correlated outputs. LOOCV, however, has a lower bias than k-fold CV.

These three cross-validation have their benefits and drawbacks. A common characteristic to all techniques is the computational efficiency. The validation set is more efficient than LOOCV and k-fold CV. However, the validation set approach is also highly random. There is no unique solution for cross-validation. Furthermore, once a validation set is obtained, multiple methods exist for estimating the test error. These techniques are discussed in the following subsection.

3.4.2. Performance evaluation

The model is evaluated on its goodness of fit. The performance measure is a way to quantify how close the predicted response is from the true response [24]. The performance measures are used for both model and metamodel validation[31]. The following equations are different measures of the performance of a model:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad \text{Mean Squared Error [24]} \qquad (3.2)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \qquad \text{Root Mean Squared Error [51] [53]} \qquad (3.3)$$

$$R^2 = 1 - \frac{\sum_{i=1} n(\hat{y}_i - y_i)^2}{\sum_{i=1} n(\overline{y} - y_i)^2} \qquad \text{Coefficient of Determination [51] [53]} \qquad (3.4)$$

$$MAE = max\left(|\hat{y}_1 - y_1|, ..., |\hat{y}_n - y_n|\right) \quad \text{Maximum Absolute Error [51] [53]} \tag{3.5}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i} \quad \text{Mean Absolute Percentage Error [17]} \tag{3.6}$$

$$RPE = \frac{\hat{y}_i}{y_i} \quad \text{Relative Prediction Error [1]} \tag{3.7}$$

$$F_1 = \frac{2 \cdot true\ positive\ labels}{2 \cdot true\ positive\ labels + false\ positive\ labels + false\ negative\ labels} \quad \text{[55]} \tag{3.8}$$

$$TPR = \frac{Number\ of\ correctly\ predicted\ positives}{Number\ of\ positives\ in\ the\ pool} \quad \text{True Positive Rate [55]} \tag{3.9}$$

where y_i is the original model output, $\hat{y}_i$ is the metamodel output and $\overline{y}$ is the mean simulation output value. The MSE and the $RMSE$ (equations 3.2 and 3.3) are small if the predicted response is close to the true one, and large otherwise. The coefficient of determination R^2 (equation 3.4) indicates the correlation between the original model and the metamodel, being 1 if perfectly correlated. The MAE and $MAPE$ (equations 3.5 and 3.6) are indicators of the local and global approximation abilities of the metamodel. The RPE (equation 3.7 is errorless when equal. Interesting with this measure is its distribution. The last two measures, F_1 and TPR (equations 3.8 and 3.9), are interesting for machine learning (like boosting) metamodeling techniques. F_1 is for binary outputs, the larger F_1 the better. TPR gives an indication of the capicity of the model to find positive labels (for both binary and real-value outputs).

3.4.3. Validation of surrogate model

In the case of AATOM, there is no choice between using real data or simulation data for validating the surrogate model as no real data is available for conducting the generation of a surrogate model. Hence, simulation data from AATOM can be used as validation data for validating the performance of the surrogate. One of the three methods can be used for validation: validation set, LOOCV or k-fold CV. Furthermore, any of the measures presented can be used for the validation process, considering the nature of the target variables (continuous or not). With the indicators given in Chapter 2, MAPE can be a first appropriate measure of performance for the surrogate. If needed, other measure can be used in a later stage. In the following section the general strategies for obtaining a surrogate model are given, including the validation process.

3.5. General strategies

In the previous sections different aspect of the generation of a metamodel are discussed: the designs (section 3.1), the techniques (section 3.3) and the model validation (section 3.4). However, there is no single strategy to generate the metamodel starting from these building blocks. Hence, in this section possible approaches for generating the metamodel are presented. The simplest common process to create the metamodel is through a "one-shot" design, meaning that an single training set is sampled prior to the training of the metamodel[1]. It is a straightforward approach to obtain a metamodel. An alternative is by introducing a loop process for the generation [53] [21]. This process can be seen in Figure 3.14.

In contrast with a "one-shot" strategy, a small sample is obtained from the original model. The metamodel is fitted on this training data and validation determines whether additional training data is needed. In the case of Figure 3.14 cross-validation is applied through k-fold CV (see section 3.4). Both strategies are not specific to certain metamodeling techniques. However, for certain techniques, it is interesting to expand the iterative strategy. This is the case for decision tree based algorithms [34] [55]. The process is depicted in Figure 3.15.

As can be seen in Figure 3.15, a large pool is sampled and subsets of these pool are used for training the model. The pool is labeled by running the ABM. Moreover, unlabeled points obtained from the metamodel after training the model are used again to run the original model and added to the training set if the budget is

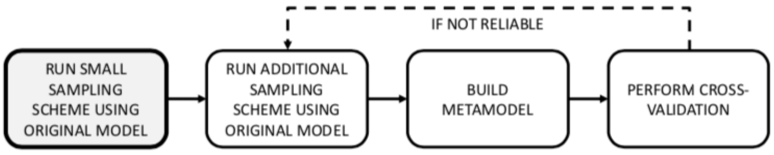

IF NOT RELIABLE
RUN SMALL SAMPLING SCHEME USING ORIGINAL MODEL
RUN ADDITIONAL SAMPLING SCHEME USING ORIGINAL MODEL
BUILD METAMODEL
PERFORM CROSS-VALIDATION

Figure 3.14: Flowchart of an iterative metamodeling strategy. [51]

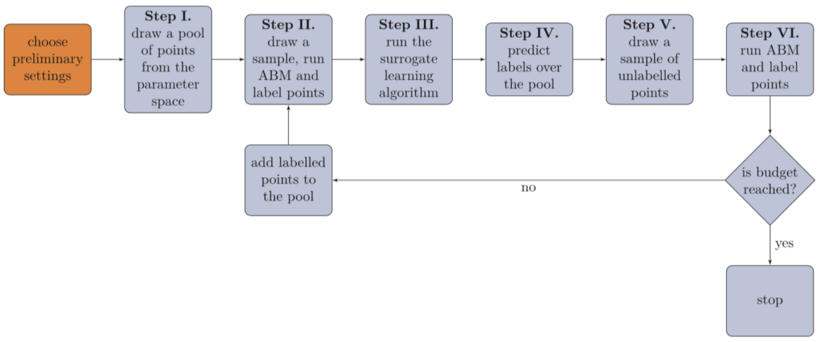

choose preliminary settings
Step I. draw a pool of points from the parameter space
Step II. draw a sample, run ABM and label points
Step III. run the surrogate learning algorithm
Step IV. predict labels over the pool
Step V. draw a sample of unlabelled points
Step VI. run ABM and label points
is budget reached?
no
yes
add labelled points to the pool
stop

Figure 3.15: Procedure for constructing a metamodel from decision tree based techniques. [34]

not reached. This approach makes use of potentially less points. Hence, the strategy can also be adapted to the metamodeling technique used.

Furthermore, an extra step can be taken after obtaining the outputs from the simulation of the original model. This can be done by detecting multimodality in the output distribution and categorizing the output if considered valuable [18]. With some tests, like the Silvermann test, it can be determined whether the output distribution is unimodal or multimodel. If the latter is true, then the output could be categorized prior to the training process of the metamodel. Hence, categorizing the output can help getting a better understanding of the dynamics the complex system. This has been applied in a past research around the model presented in Chapter 2[25]. The process is given in Figure 3.16.

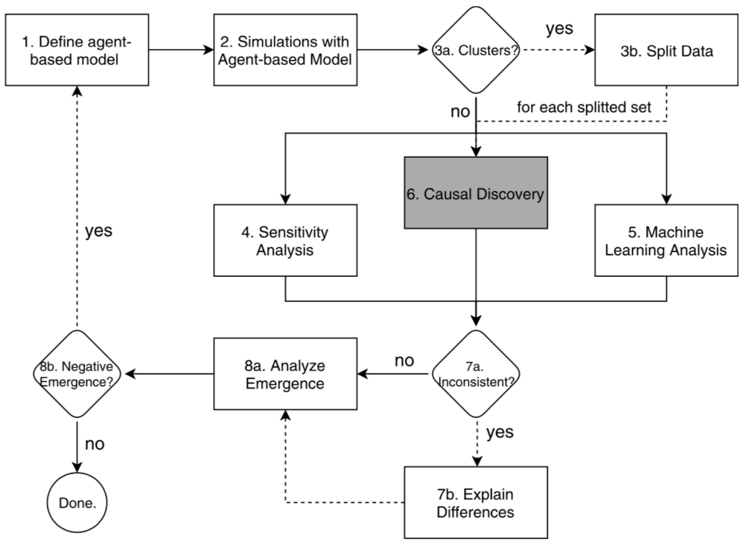


Figure 3.16: The AbACaD methodology. [25]

The methodology depicted in Figure 3.16 was used for analyzing the ABM emergence and causality. In this process machine learning was used as a metamodel. Hence, in that research the data set is clustered for its benefits of better understanding its dynamics. Furthermore, sensitivity analysis and machine learning are used to get a richer insight into the behavior if the model[25].

3.6. Conclusion

Abstracting an existing model requires a combination of different methods and approaches, starting with the sampling technique. Many design can be used for obtaining a set of sampled data points. However, the approach is to start with a simple static design and consider a more sophisticated design after evaluation. Other considerations are more complex static designs are adaptive designs. With regards to agent-based models, it is shown that surrogate models can be very interesting for approximating the agent-based models. Moreover, a tendency exist to apply machine learning algorithms as surrogates.

Furthermore, plenty of techniques exist for generating surrogate models. Several are given in this chapter. However, it is difficult to compare them and select the most appropriate technique given the AATOM. Hence a short evaluation is described based on the time efficiency, accuracy and transparency. The technique chosen from the evaluation is the Random Forest.

Finally, independent of the technique chosen, there are multiple strategies for training a surrogate model. Every strategy relies on a process involving the training of the surrogate model and the performance evaluation based on a certain measure. Hence, in the process the validation of the surrogate model plays a crucial role. The measure chosen in this report, as a starting point, is the MAPE. All these different elements composing the approach of abstracting an existing model will be part of the coming thesis research and are taken into the planning of the following chapter.

4 Research Plan

In this chapter, the main questions leading to the thesis research are presented and the planning for attempting to answer them. The questions contain a main research question and subquestions, dividing the main question into smaller questions to answer them. The subquestions are also supporting the work packages included in the thesis planning. The thesis planning is a division of work packages and tasks over time in order to keep the research time-bounded.

4.1. Research Questions

> RQ : How to obtain an approximation of an agent-based model for simulating airport terminal operations while preserving the important dynamic (emergent) properties of the model and getting an insight into the underlying function?

SubQ1 : Which factors and indicators are providing an insight into the underlying relations of the original model ?

- Which factors should be selected for the generation metamodel ?
- Which indicators should be used for for identifying the overall performance of the model ?
- What experimental design/sampling technique is the most appropriate for obtaining the training data set ?

SubQ2 : How can the metamodel, based on the original model, capture the relationships of inputs and outputs ?

- Which metamodeling technique should be used for approximating the ABM ?
- How to fit the metamodel on the training data set ?
- What method should be used for validating the generated metamodel ?

SubQ3 : How should the metamodel be generated in order to efficiently analyze the behavior of the underlying model?

- What strategy to adopt for the generation of the metamodel?
- How to maintain a modular approach in order to allow changes for factors and/or outputs ?

4.2. Research Proposal

Below are given the elaboration of the different work packages that are all incorporated into the thesis planning. The packages are chronologically ordered, giving a visual representation of the step-wise approach of the thesis research.

1. **Simple AATOM model & surrogate model setup**: The first work package is considered as the basis of the framework for approximating AATOM. Hence, initially a reduced scope AATOM model is taken as the original model for surrogate modelling. Around the reduced model the process for generating a surrogate model is developed, taking into account the modularity for dealing with larger scope models. The work package also includes analysis of the surrogate model.

    (a) Determine model parameters for simulation

    (b) Setup AATOM simulation

    (c) Develop surrogate model

    (d) Analyze & interpret first results

2. **Model scope & surrogate extension**: After obtaining results from the reduced scope AATOM model, parameters and indicators are added/removed to enlarge the scope of the model, also altering the data format for training the surrogate model. It requires to bring modifications on the AATOM model for simulation and adapt the framework structure for modelling the surrogate model. The obtained results can be compared with the results obtained by another researcher also focussing on surrogate modelling. Moreover, a second metamodelling algorithm can be incorporated for additional comparisons of the surrogate model performances.

    (a) Extend scope of AATOM model

    (b) Include second metamodelling algorithm

    (c) Compare performances of surrogate models

3. **Implement feedback & additional model**: Ultimately, the framework developed over the two first packages and the results obtained during that time are presented at the midterm. During the midterm, the structure of the thesis report is also presented. All the feedback obtained during the presentation is directly implemented after the midterm, the required changes or small additions on the existing framework are applied. Furthermore, all the collected results and other work are written down in the first version of the thesis report.

    (a) Implement feedback from midterm

    (b) Run last experiments

    (c) Report last experiments

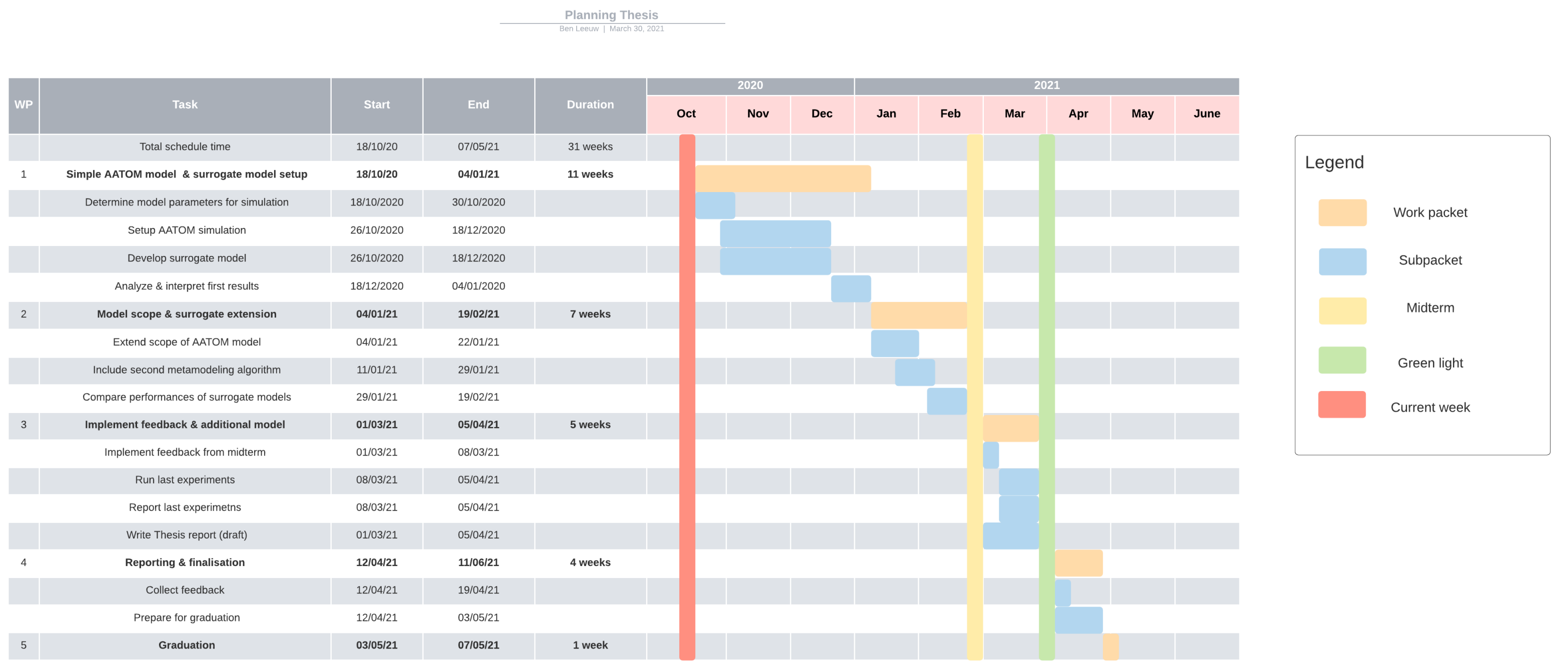
    (d) Write thesis report (draft)

4. **Reporting& finalisation**: The first version of the thesis report should be as close as possible to the final version, meaning that everything is included into a logical report structure. In addition, all auxiliary information needed to reproduce the research is also included in the first version of the report. The last steps before the graduation are the last changes on the report based on feedback and the preparation for the defence of the thesis.

    (a) Collect feedback

    (b) Prepare for graduation

5. **Graduation**



WP	Task	Start	End	Duration
	Total schedule time	18/10/20	07/05/21	31 weeks
1	**Simple AATOM model & surrogate model setup**	**18/10/20**	**04/01/21**	**11 weeks**
	Determine model parameters for simulation	18/10/2020	30/10/2020	
	Setup AATOM simulation	26/10/2020	18/12/2020	
	Develop surrogate model	26/10/2020	18/12/2020	
	Analyze & interpret first results	18/12/2020	04/01/2020	
2	**Model scope & surrogate extension**	**04/01/21**	**19/02/21**	**7 weeks**
	Extend scope of AATOM model	04/01/21	22/01/21	
	Include second metamodeling algorithm	11/01/21	29/01/21	
	Compare performances of surrogate models	29/01/21	19/02/21	
3	**Implement feedback & additional model**	**01/03/21**	**05/04/21**	**5 weeks**
	Implement feedback from midterm	01/03/21	08/03/21	
	Run last experiments	08/03/21	05/04/21	
	Report last experimetns	08/03/21	05/04/21	
	Write Thesis report (draft)	01/03/21	05/04/21	
4	**Reporting & finalisation**	**12/04/21**	**11/06/21**	**4 weeks**
	Collect feedback	12/04/21	19/04/21	
	Prepare for graduation	12/04/21	03/05/21	
5	**Graduation**	**03/05/21**	**07/05/21**	**1 week**

Figure 4.1: Gantt for chart of thesis planning.

III

Supporting work

1 AATOM Models

In this chapter the remaining models developed and evaluated throughout the study are given. Each model description contains a terminal layout illustration, an overview of the model parameters and indicators and the prediction accuracy obtained after training the model. For all simulations, including the simulations used in the thesis paper, the number of simulations have been determined through the coefficient of variation. The coefficient of variation determination is shown in Figure 1.1.

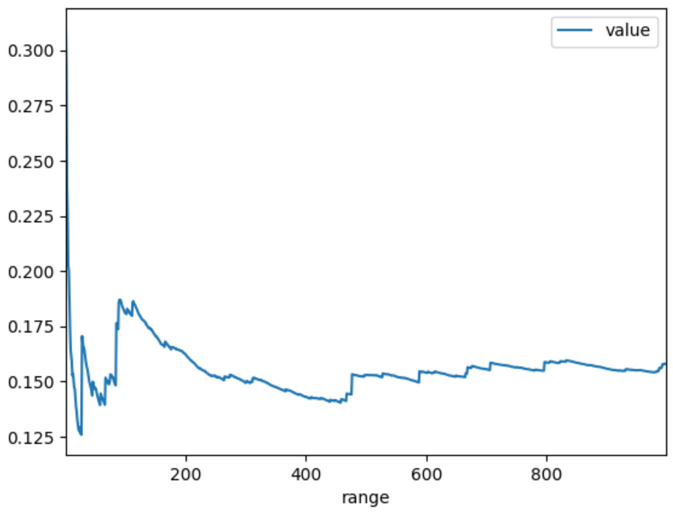


Figure 1.1: Coefficient of variation for 1000 simulation runs, for the simple single output model.

As can be seen in Figure 1.1, a large number of simulations runs is needed to obtain stable results. An approximate number of 500 runs is required. However, even for the simplest single output model given in Section 1.1, the required number of simulations runs is larger than the computational capacity available on the university server. Hence, the simulations for the different have been ran using all computational capacity, trying to reach the 500 runs. Moreover, the large set of parameters and the wide range of parameter values are limiting the possible number of times a specific data point can be ran through the simulation.

In the remainder of the Appendix the different AATOM models prior to the final model are presented. It has been an iterative process of identifying the relevant parameters and understanding the importance of the different indicators to better understand the system behaviour. Section 1.1 contains the first AATOM model used in the study for surrogate modelling. At the start, only one indicator was chosen to focus initially on the development of the methodology and to get experienced with applying Random Forest in the context of surrogate modelling. Section 1.2 elaborates on the second model, where multiple inputs and multiple outputs were included.

1.1. Single Output Security Checkpoint Model

The first developed surrogate model was a single output model, only predicting the behaviour of the system on the average queuing time of the security checkpoint. The AATOM model used for simulation is visually represented in Figure 1.2. As can be seen, the model structure is simple, only a small terminal including a check-in, checkpoint and gate. The model differs with the AATOM model presented in the thesis paper on the arrival of passengers. For the simple single output model only one flight is scheduled for every simulation run (scheduled at the end of the simulation time), whereas for the more elaborated model in the paper the flight schedule is a model parameter. The parameters and indicator for the single output model are given in Table 1.1. All the parameters, except for the v_{des}, are related to the security checkpoint.

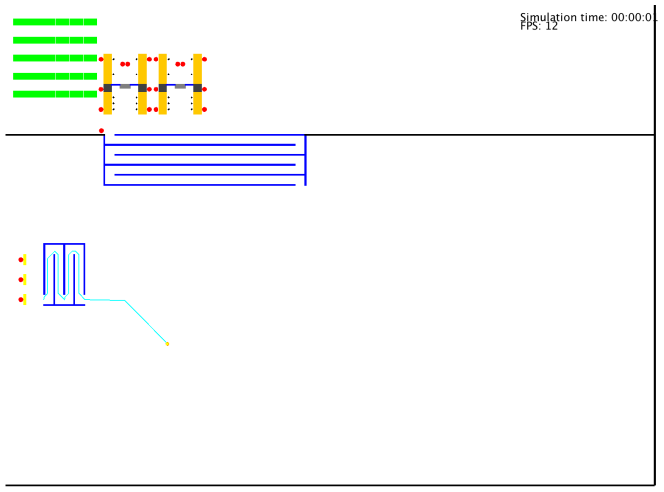


Figure 1.2: Airport Terminal layout for the single output checkpoint model.

Table 1.1: The defined model parameters and indicators of the single output security checkpoint model, according to the AATOM architecture, chosen for simulation purposes.

Parameters	Description	Unit
n_{lanes}	number of lanes at the security check-point	-
$n_{dropoff}$	number of luggage drop-off points	-
$n_{collect}$	number of luggage collection points	-
$t_{collect}$	Mean collection time	[s]
$t_{lugdrop}$	Mean luggage drop-off time	[s]
v_{des}	Desired walking speed for passengers	[m/s]
proportionETD	Proportion of passengers that receive an ETD check	-
proportionWTMD	Proportion of passengers that receive a patdow	-
illegalObjectThreshold	Proportion of passengers that does not have an illegal item	-
Indicators		
$scQueue_{avg}$	average security checkpoint queuing time	[s]

The evaluation of the surrogate model accuracy is given in Table 1.2. From the table, it can be observed that surrogate model has a low execution time and a high accuracy. Moreover, the relative importance of the model parameters is given in Figure 1.3(a). The most important variable is the number of lanes at the security checkpoint, with the number of drop-off and collection points in second place. These three variables are distinctively more important than the remaining model variables. This is also the reason for using these three variables in the final surrogate model. Figure 1.3(b) and Table 1.3 also identify the same three parameters as most crucial for obtaining a good approximation of the model.

Table 1.2: Performances of the default Random Forest model for the single output checkpoint model on predicting the average security checkpoint queuing time.

Performances	**Default RF**
Training time [sec]	3.24
Execution time [sec]	0.20
Average Error [sec]	41.40
Accuracy (MAPE)	90.35%

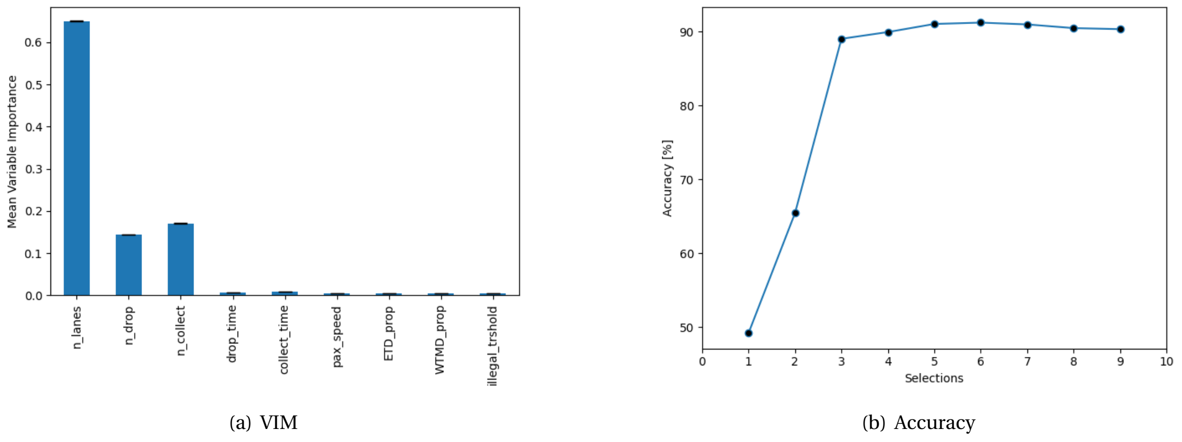


Figure 1.3: The variable importance and model accuracy with addition of variables from the Random Forest surrogate model for predicting the checkpoint average queuing time.

Table 1.3: Mean accuracy of the surrogate models based on the Random Forest and the Artificial Neural Network model with increasing numbers of variables, visually represented in Figure 1.3(b) for the Random Forest model.

ID	Parameters	Included Parameters	RF	ANN
1	$[n_{lanes}]$	[1]	49.19%	43.97%
2	$[n_{dropoff}]$	[1,2]	65.43%	51.34%
3	$[n_{collect}]$	[1,2,3]	89.05%	81.62%
4	$[t_{lugdrop}]$	[1,2,3,4]	89.96%	84.56%
5	$[t_{collect}]$	[1,2,3,4,5]	91.06%	87.39%
6	$[v_{des}]$	[1,2,3,4,5,6]	91.24%	91.1%
7	$[proportionETD]$	[1,2,3,4,5,6,7]	91.00%	91.37%
8	$[proportionWTMD]$	[1,2,3,4,5,6,7,8]	90.49%	92.12%
9	$[illegalObjectThres]$	[1,2,3,4,5,6,7,8,9]	90.35%	93.41%

1.2. Multi Output Checkpoint Check-in Model

The second model developed, similar to the final (and third) model, is given in Figure 1.4. The airport terminal layout is identical to the final model presented in the thesis paper. The difference between the two models is that the model presented in this section contains less time slots and a reduced parameter range with only 0, 142 or 248 passengers for each time slot. The indicators, however, are the same for both models. The parameters and indicators are given in Table 1.4.

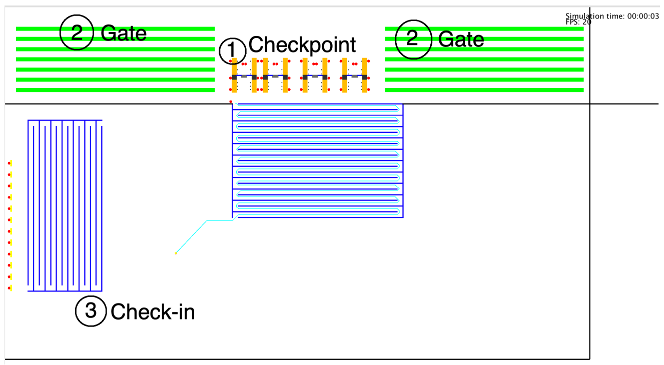


Figure 1.4: Airport Terminal layout for the single output checkpoint model.

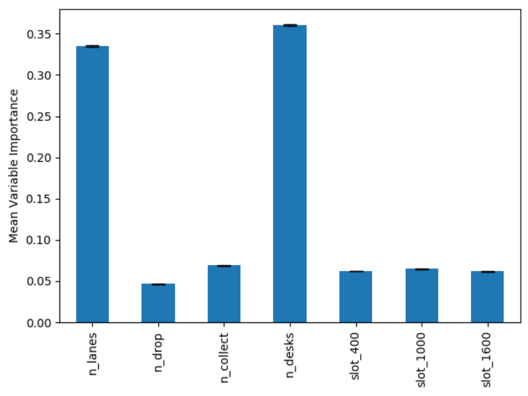
Table 1.4: The defined model parameters and indicators of the multi output checkpoint check-in model, according to the AATOM architecture, chosen for simulation purposes.

Parameters	Description	Unit
n_{lanes}	number of lanes at the security check-point	-
$n_{dropoff}$	number of luggage drop-off points	-
$n_{collect}$	number of luggage collection points	-
n_{desks}	number of check-in desks	-
$slot_{400}$	schedule slot at time 400(sec)	[s]
$slot_{1000}$	schedule slot at time 1000(sec)	[s]
$slot_{1600}$	schedule slot at time 1600(sec)	[s]
Indicators		
$scQueue_{avg}$	average security checkpoint queuing time	[s]
$checkinQueue_{avg}$	average check-in queuing time	[s]
$throughput_{checkin}$	number of passenger that passed the check-in	-
$throughput_{sc}$	number of passenger that passed the security checkpoint	-
$TimeToGate_{avg}$	average time to the gate	[s]
$n_{missedFlights}$	number of missed flights	-

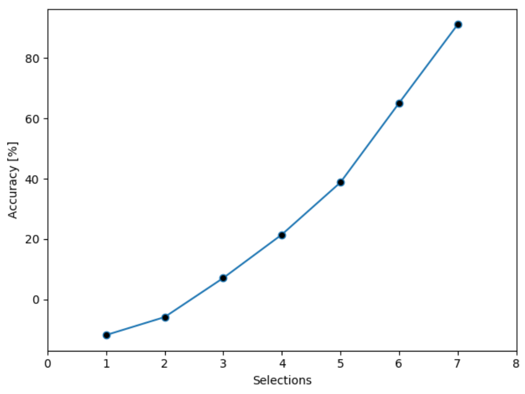
The performance and accuracy of the second model is given in Table 1.5, and compared with the final model (Default RF Extended) presented in the thesis paper. The final is only slightly slower on training and execution time. However, for the accuracy, there is a large improvement on the overall accuracy and on the $n_{missedFlights}$. The main reason is that for the final model, the range of values for the time slot parameters has been enlarged. Hence, the surrogate model has more information regarding the flight schedule to better predict the system behaviour. Moreover, the importance of variables is given in Figure 1.5(a). It can be observed that the same two variables as the final model are considered most important to predict the behaviour of the system. The difference is that the time slots have more importance than for the final model. This phenomenon can also be seen in Figure 1.5(b). The increase in accuracy with the addition of model parameters is almost linear, indicating the importance of the information contained in every parameter to reach an acceptable surrogate model accuracy. The negative accuracy given in for the first two variables is a value linked to the measure of error (MAPE). Both accuracy values are representing an accuracy of 0%. The precise accuracy at every point is given in Table 1.6.

Table 1.5: Comparison of performances between the the two options of scheduling featuring.

Performances	**Default RF**	**Default RF Extended**
Training time [sec]	1.47	3.66
Execution time [sec]	0.06	0.11
Accuracy		
$checkinQueue_{avg}$	95.05%	91.62%
$scQueue_{avg}$	95.50%	96.03%
$TimeToGate_{avg}$	96.66%	97.13%
$n_{missedFlights}$	**67.55%**	**83.38%**
$throughput_{checkin}$	93.16%	94.69%
$throughput_{sc}$	93.94%	94.46%
Mean Accuracy	90.31%	92.89%



(a) VIM



(b) Accuracy

Figure 1.5: The variable importance and model accuracy with addition of variables from the Random Forest surrogate model for predicting the checkpoint average queuing time.

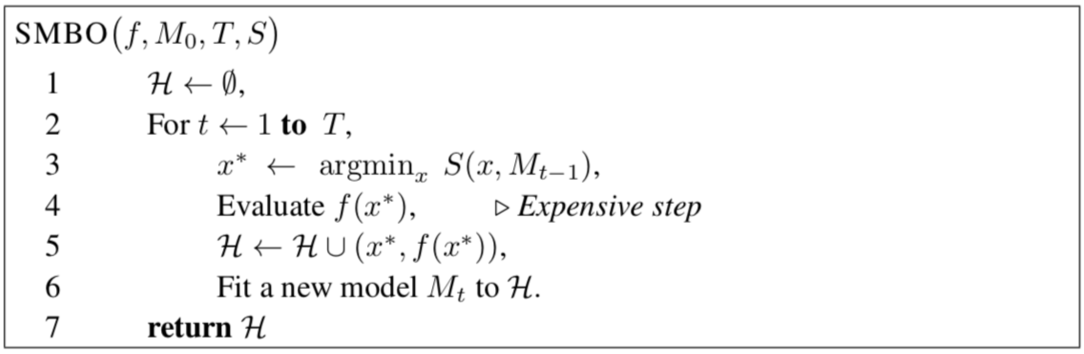
Table 1.6: Mean accuracy of the surrogate model based on the Random Forest model with increasing numbers of variables, visually represented in Figure 1.3(b).

ID	Parameters	Included Parameters	RF
1	$[n_{lanes}]$	[1]	0.00%
2	$[n_{dropoff}]$	[1,2]	0.00%
3	$[n_{collect}]$	[1,2,3]	7.11%
4	$[n_{desks}]$	[1,2,3,4]	21.55%
5	$[slot_{400}]$	[1,2,3,4,5]	38.77%
6	$[slot_{1000}]$	[1,2,3,4,5,6]	65.05%
7	$[slot_{1600}]$	[1,2,3,4,5,6,7]	90.31%

2 Hyperparameter Tuning

2.1. Bayesian Optimization

The surrogate model used for this research is the random forest. Random forest can be very powerful in approximating models in a very accurate way. However, there are parameters (hyperparameters), proper to the random forest, that influence the performance and accuracy of the model. Multiple options exist for tuning these parameters in order to obtain an accurate surrogate model. The most intuitive one is the use of domain experts for choosing the values of the different hyperparameters. In this research, the domain knowledge is not sufficient for tuning the model. Hence, an alternative is to automatically optimize the hyperparameters. The bayesian optimization, which has been proven reliable and efficient in past research, is chosen as the optimization method for this research [4] [54]. The algorithm dictating the method is presented in Figure 2.1.



```
SMBO($f, M_0, T, S$)
1    $\mathcal{H} \leftarrow \emptyset$,
2    For $t \leftarrow 1$ **to** $T$,
3        $x^* \leftarrow \operatorname{argmin}_x S(x, M_{t-1})$,
4        Evaluate $f(x^*)$,        ▷ *Expensive step*
5        $\mathcal{H} \leftarrow \mathcal{H} \cup (x^*, f(x^*))$,
6        Fit a new model $M_t$ to $\mathcal{H}$.
7    **return** $\mathcal{H}$
```

Figure 2.1: The pseudo-code for Bayesian optimization [4]

As can be seen in the pseudo-code depicted in Figure 2.1, the bayesian optimization relies on the observation history H. For every trial t from T, the next sample point is determined by optimizing the predefined criterion (step 3). Then both the sample point and its respective function value $(x^*, f(x^*))$ are added to the observation history and a model is fitted on these observations. Hence, both the criterion for optimization and the model to fit on the observation history need to be determined prior to the optimization iterations. They are both elaborated in the rest of this section.

Improvement Criteria

There are numerous criterion for which to optimize in the bayesian optimization. For this research three criteria are considered: Probability of improvement, Expected improvement and Gaussian process (GP) upper confidence bound [54]. They are all listed below:

Probability of improvement - The PI attempts to find a point close to the current optimal point that exceeds it. The function is based on the probability of the given point being greater than the current optimal, the equation is given below:

$$PI(x) = P\left(f(x) \geq f(x^*)\right) \tag{2.1}$$

The PI, as given in equation 2.1, has a major disadvantage, it can lead to a local optimum instead of resulting with the global optimum. This issue can be resolved by introducing an additional parameter ε under which the data point is never selected as optimal. The upgraded equation is given below:

$$PI(x) = P\left(f(x) \geq f(x^*) + \varepsilon\right) \tag{2.2}$$

Expected improvement - EI, in contrast with PI, does not easily fall into a local optimum. It is based on the expected value of improvement. Here, the improvement is the difference between the data point function value and the current optimal value. The degree of improvement is given below [54]:

$$I(x) = max\left\{0, f_{t+1}(x) - f(x^*)\right\} \tag{2.3}$$

From equation 2.3, if the function value of the data point is less than the current optimal value then the improvement equals zero. For the optimization, the expected value of improvement is considered. Hence, in a more general case, the expected improvement is given by (where y is the function value at the data point):

$$EI_{y^*}(x) = \int_{-\infty}^{\infty} max\left(y^* - y, 0\right) p_M(y|x) dy \tag{2.4}$$

GP upper confidence bound - Here, the trade-off is made between the current optimal value or the function value of that data point based on the distribution of x. The trade-off is given below:

$$UCB(x) = \mu(x) + K\sigma(x) \tag{2.5}$$

where K is the trade-off parameter. As depicted in equation 2.5, the trade-off is made between the current optimum with high values for $\mu(x)$ and a data point in a low confidence region (with a low $sigma(x)$). Finally, the EI criterion is chosen as the optimization criterion for the bayesian optimization method. The reasons for this are the simplicity of the criterion (intuitive) and proven to be applicable on different settings [4].

Approximation of f

The improvement criterion, discussed in the previous section, serves for the optimization process. However, there are multiple algorithm/techniques that can search the parameter space for an optimal point based on the improvement criterion. In this research two algorithm are considered: the Gaussian Process and the Tree-structured Parzen Estimator. Both are discussed in this section.

Gaussian Process - The GP is a stochastic process in which every sub-collection has a Gaussian distribution [54]. Similar to the Gaussian distribution, the GP has a mean and covariance function. Multiple functions are possible for defining the mean and covariance Thus, if the prior distribution is assumed to be a GP, then the distribution of f is also a GP [4]. On other words, the Gaussian process returns the distribution of f over all its values, with a certain mean and variance. An intuitive representation of the Gaussian process is given in Figure 2.2.

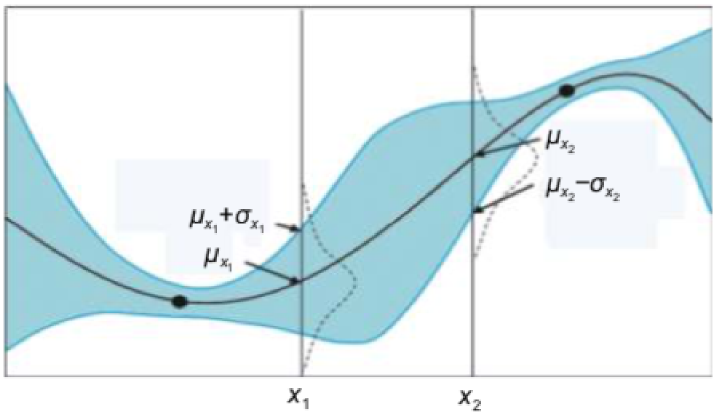


Figure 2.2: One-dimensional Gaussian Process Representation with two data points. [54]

As can be seen in Figure 2.2, close to the two data points that dictate the GP the standard deviation remains small. However, further away form the data points, the values become more uncertain. hence, referring back to equation 2.4, p_M is the modeled by the posterior GP of the observation history H. EI, if optimized in the GP, represents the trade-off between a mean close or higher than the current optimal and the unexplored regions.

TPE - TPE, in contrast with GP, models $p(x|y)$ and $p(y)$ instead of $p(y|x)$ [4]. TPE models $p(x|y)$ by transforming the prior distribution with non-parametric densities. Two densities are used for defining $p(x|y)$:

$$p(x|y) = \begin{cases} l(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases} \tag{2.6}$$

From equation 2.6, TPE determines y^* to be same quantile γ for which $p(y < y*) = \gamma$. No model is needed for $p(y)$ [4]. Hence, equation 2.4 defining the expected improvement can be rewritten as:

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} max\left(y^* - y, 0\right) \frac{p(y|x)p(y)}{p(x)} dy \tag{2.7}$$

which ultimately leads, through substitution [4],

$$EI_{y^*}(x) = \left(\gamma + \frac{g(x)}{l(x)}(1-\gamma)\right)^{-1} \tag{2.8}$$

From equation 2.8, it can be deduced that high probability for $l(x)$ and a low probability for $g(x)$ would be ideal for optimizing the improvement. Taking into account equation 2.6, candidates for x^* can be drawn from l and then evaluated through $\frac{g(x)}{l(x)}$.

In this research, the TPE is opted for optimizing the EI chosen in previous section, The two main reasons are: TPE outperforms GP in research (admitting that both perform better than manual or random search) and its implementation in the used optimization library (and GP not).

Implementation

For the implementation, the Python Hyperopt library is used [3], based on research around automatic hyperparameter optimization [5]. The automization is divided into four components:

- **Distribution specification language**: this is the language used for specifying the sample configuration of the hyperparameters. In the case of TPE, as mentioned earlier, the distribtuion are transformed into non-parametric densities.
- **Loss function**: This is the criterion that we decide to minimize, e.g. the mean squared error. It is the objective value for which the method will search the optimal hyperparameter set.

- **Hyperparameter Optimization Algorithm (HOA)**: There are several algorithms possible, However, in practice, the random search and TPE algorithm are implemented in the hyperopt library. The algorithm uses the specified distribution and the observation history for determining the next sampling point.
- **Database**: Database refers to the storage of all prior observations in order to support the optimization algorithm.

The implementation of the four components is given in Appendix. There are still some aspects of the implementation can vary depending on the performance results of the optimization on the given model. These are the main settings that can changed to alter the performance of the optimization:

- Boundaries of the parameter space
- Distribution of the hyperparameters (according to the specification language)
- Metric for the loss function
- Number of trials

2.2. Results

The optimization results for the different models presented in Appendix 1. Table 2.1, Table 2.2 and Table 2.3 contain the hyperparameters obtained for the single output, multi-output and final model respectively. In the same order, the model performances are given in Table 2.4, Table 2.5 and Table 2.6. For the single output, in Table 2.4, only a small improvement of 0.36% is achieved with 500 trials and 1000 trials. In addition, in order to achieve this improvement both the training time and execution time are largely increased. Hence, the optimization is not improving the surrogate model of the single output checkpoint model. In Table 2.5 two optimization strategies have been applied, one optimizing the mean accuracy of the surrogate model and one optimizing the weakest accuracy of the surrogate model. As can be seen, there is no improvement on the $n_{MissedFlights}$ or on the mean accuracy. The optimization is not improving the mutli-output. The same result holds for the final model, presented in the Thesis paper. Table 2.6 shows that no improvement is obtained by applying the hyperparameter optimization for the final surrogate model. Figure 2.3, 2.4, 2.5, 2.6 and 2.7 depict the change of hyperparameters and accuracy over the optimization trials.

Table 2.1: The optimal hyperparameters for the optimized RF 500 trials and the optimized RF 1000 trials.

Hyperparameter	**Default RF**	**Optimized RF 500**	**Optimized RF 1000**
mtry	9	9	8
sample size	num_observations	3907	4031
replacement	with	without	without
nodesize	1	5	4
ntree	100	558	910

Table 2.2: The optimal hyperparameters for the optimized RF 500 trials and the optimized RF 1000 trials.

Hyperparameter	**Default RF**	**Optimized RF 1000 mean**	**Optimized RF 1000 min**
mtry	9	8	6
sample size	num_observations	9352	6861
replacement	with	without	with
nodesize	1	3	2
ntree	100	690	287

Table 2.3: The optimal hyperparameters for the optimized RF 500 trials and the optimized RF 1000 trials.

Hyperparameter	**Default RF**	**Optimized RF 500**
mtry	9	4
sample size	num_observations	2932
replacement	with	with
nodesize	1	2
ntree	100	471

Table 2.4: Comparison of performances between the default RF, the optimized RF for 500 trials and the optimized RF for 1000 trials.

Performances	**Default RF**	**Optimized RF 500**	**Optimized RF 1000**
Training time [sec]	3.24	18.14	29.51
Execution time [sec]	0.20	0.86	1.37
Accuray			
$scQueue_{avg}$	90.35%	90.71%	90.71%

Table 2.5: Comparison of performances between the default RF, the optimized RF for 1000 trials based on mean accuracy and the optimized RF for 1000 trials based on the weakest accuracy.

Performances	**Default RF**	**Optimized RF 1000 mean**	**Optimized RF 1000 min**
Training time [sec]	1.47	10.60	2.16
Execution time [sec]	0.06	0.37	0.14
Accuracy			
$checkinQueue_{avg}$	95.05%	94.69%	95.12%
$scQueue_{avg}$	95.50%	95.22%	95.37%
$TimeToGate$	96.66%	96.52%	96.78%
$n_{missedFlights}$	**67.55%**	**67.44%**	**66.81%**
$throughput_{checkin}$	93.16%	92.98%	93.55%
$throughput_{sc}$	93.94%	93.65%	93.70%
Mean Accuracy	90.31%	90.08%	90.22%

Table 2.6: Comparison of performances between the Random Forest surrogate model with default hyperparameters and the optimized Random Forest surrogate model with 500 trials.

Performances	**Default RF**	**Optimized RF 500**
Training time [sec]	3.82	3.11
Execution time [sec]	0.12	0.36
Accuracy		
$checkinQueue_{avg}$	91.66%	90.48%
$scQueue_{avg}$	96.02%	95.32%
$TimeToGate$	97.13%	96.81%
$n_{missedFlights}$	83.44%	78.53%
$throughput_{checkin}$	94.71%	92.94%
$throughput_{sc}$	94.44%	92.60%
Mean Accuracy	92.90%	91.11%

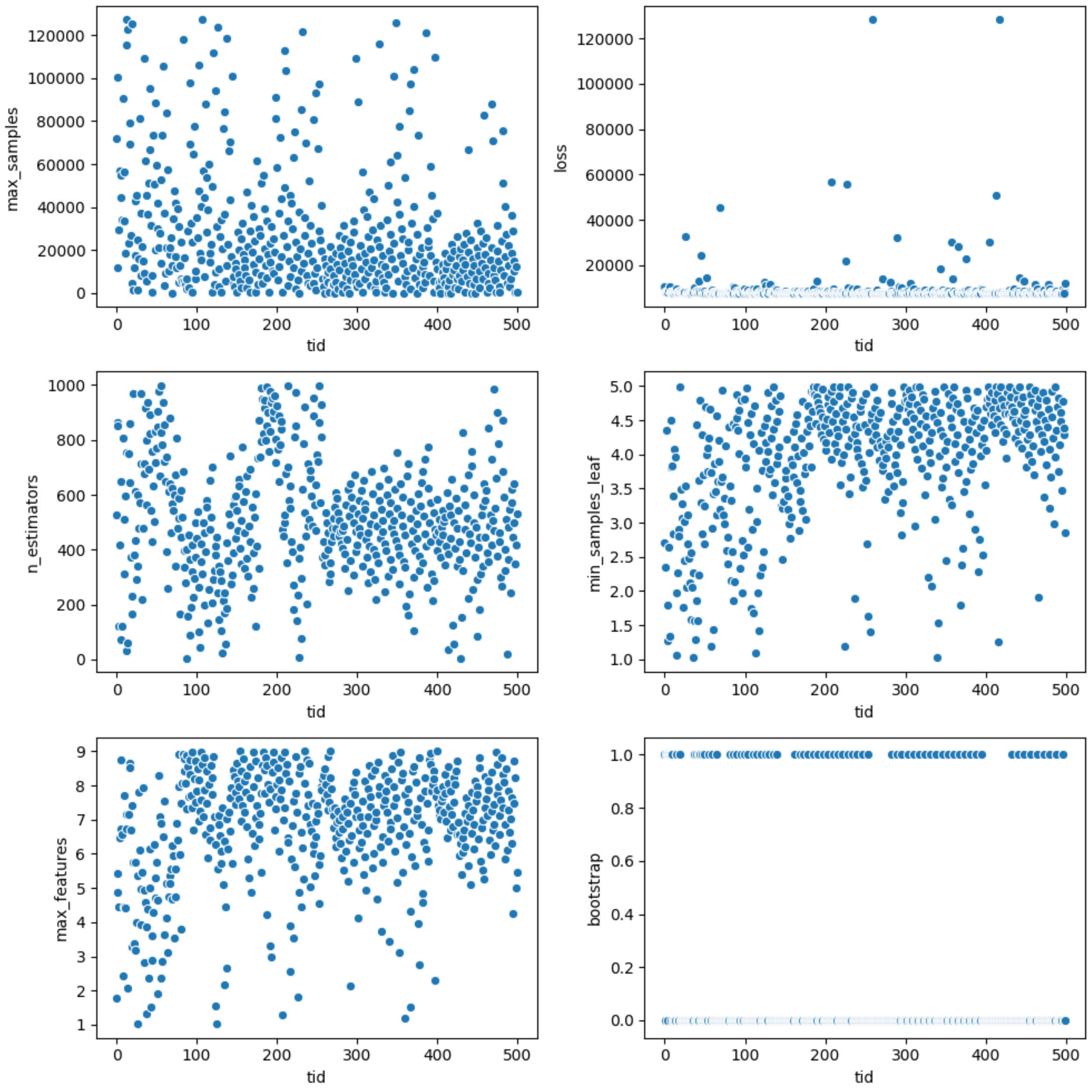


Figure 2.3: Hyperparameters values for the single output checkpoint model over the number of trials for 500 trials.

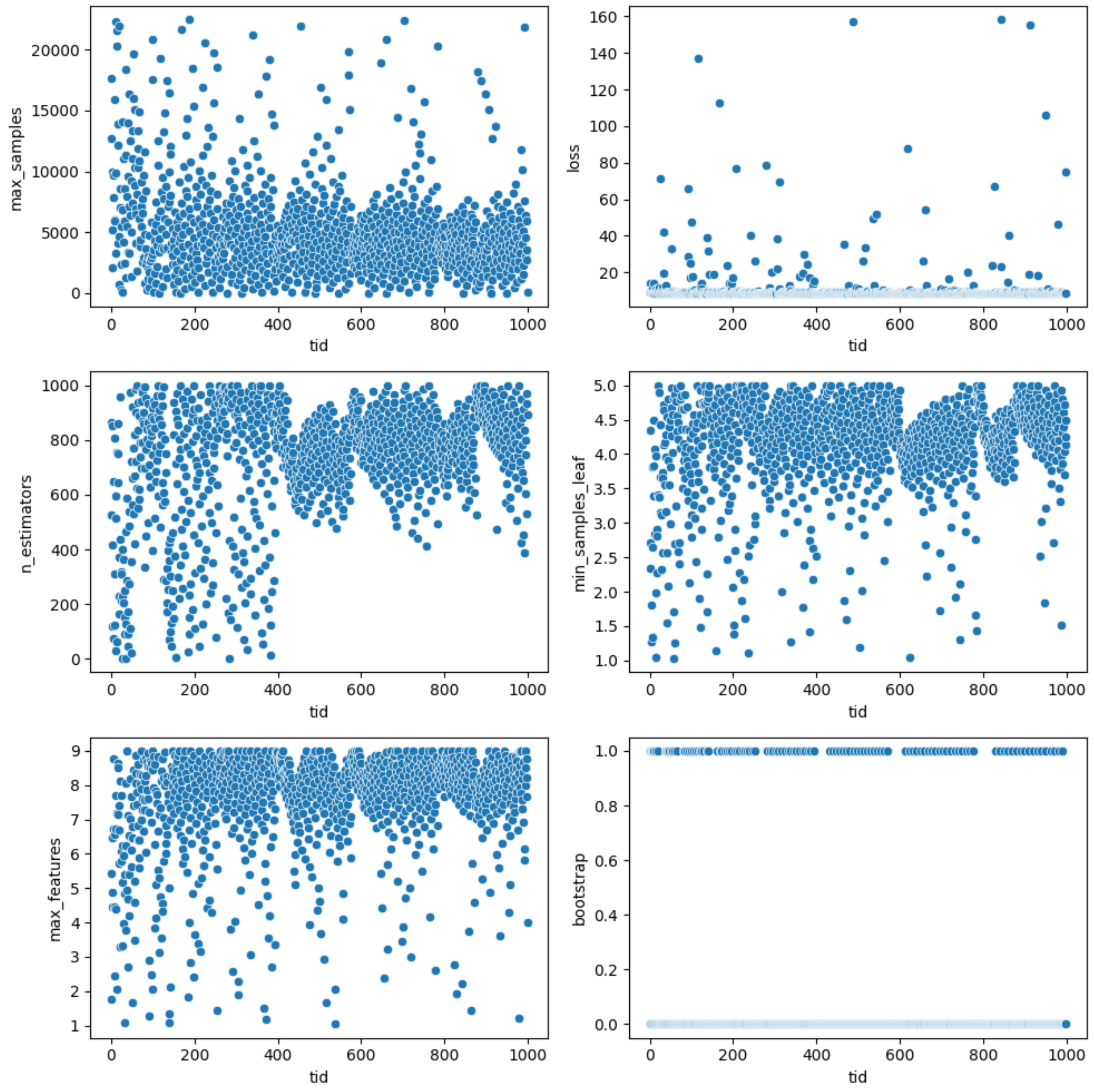


Figure 2.4: Hyperparameters values for the single output checkpoint model over the number of trials for 1000 trials.

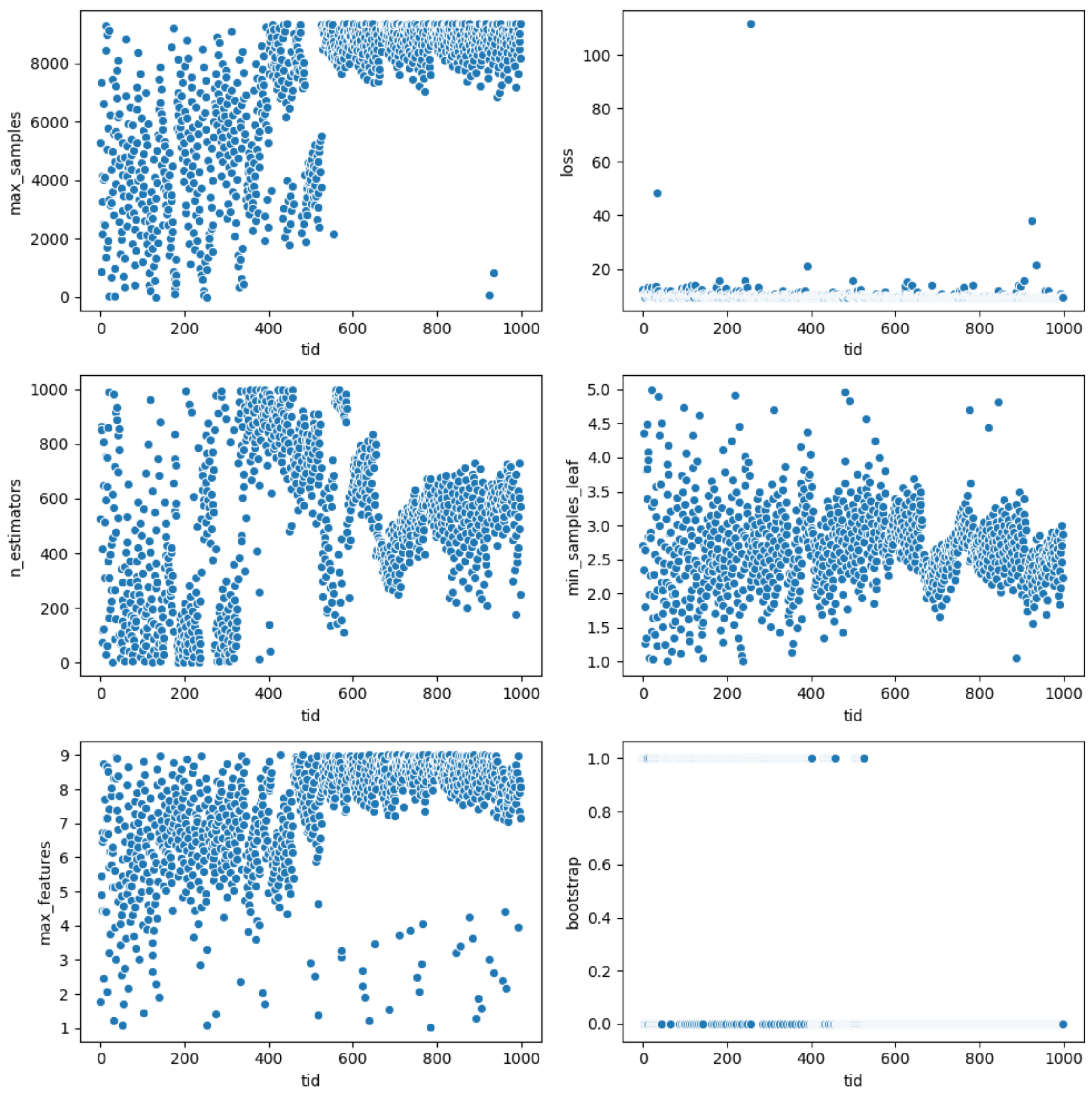


Figure 2.5: Hyperparameters values for the model presented in Appendix 1 Section 1.2 over the number of trials for 1000 trials optimizing the mean accuracy.

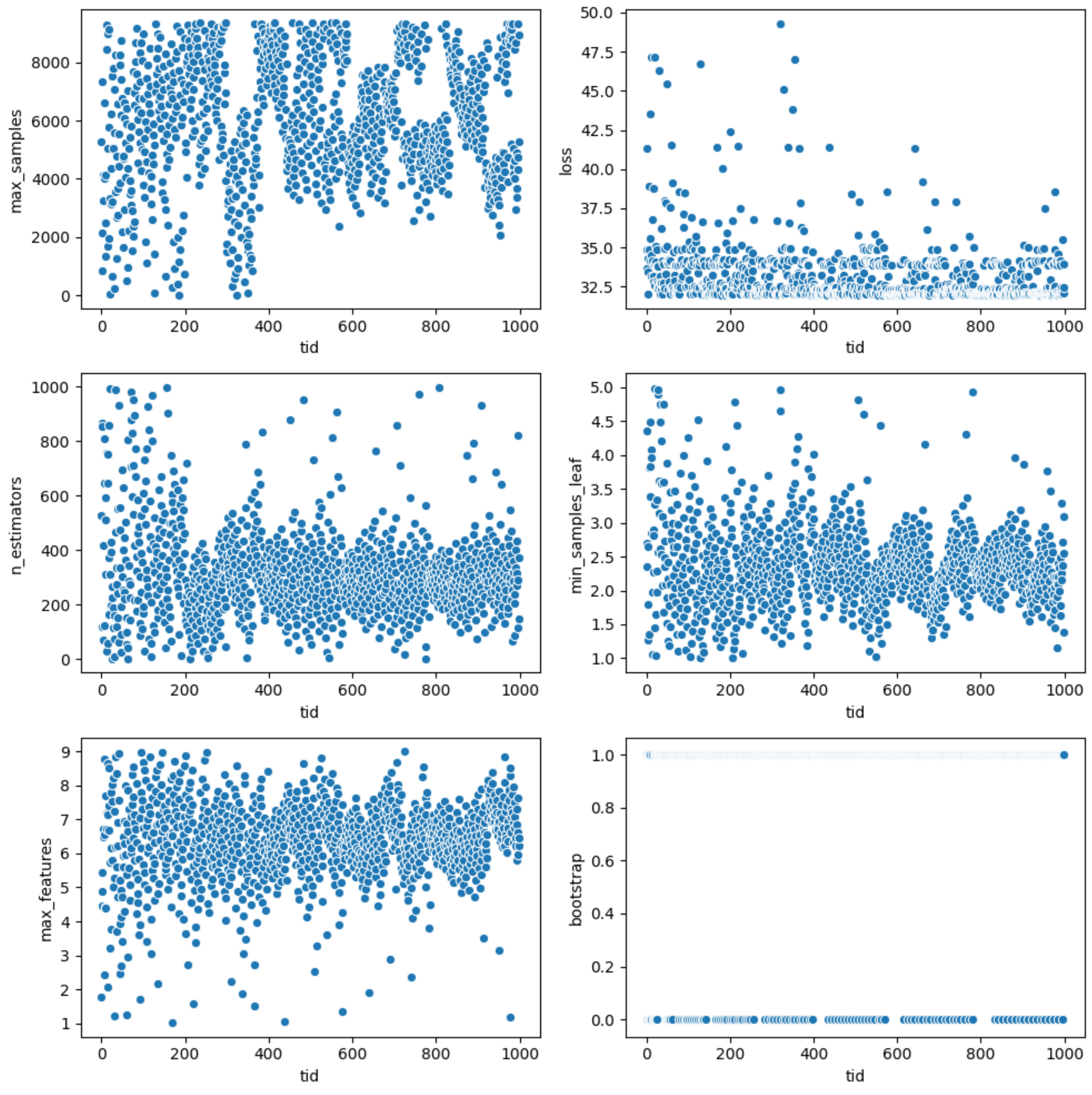


Figure 2.6: Hyperparameters values for the model resented in Appendix 1 Section 1.2 over the number of trials for 1000 trials optimizing the weakest accuracy.

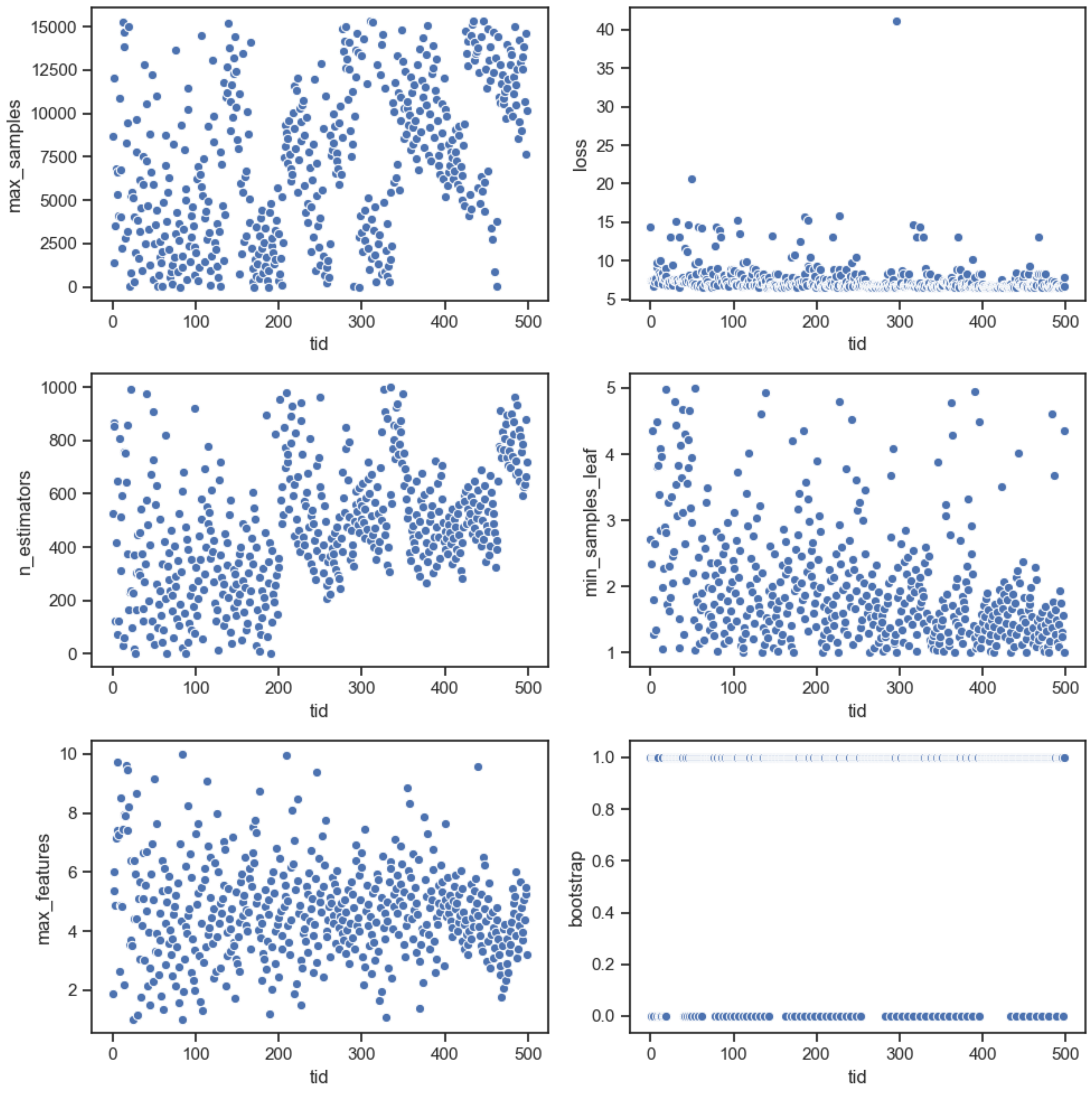


Figure 2.7: Hyperparameters values for the final model presented in the thesis paper over the number of trials for 500 trials.

3
Re-sampling Imbalanced Domains

3.1. Re-sampling under imbalanced domains

The surrogate model, currently, is experiencing difficulties predicting a specific indicator. In general, predictive models can encounter issues with the distributions of certain target variables. One of the issues is the imbalanced domain of the target variable value, making it harder to predict for the predictive model. Hence, different strategies exist for coping with this phenomenon [10]. The strategies can be subdivided into four categories: **data pre-processing**, special-purpose learning methods, prediction post-processing or hybrid methods. The first type of strategy is deemed as the simplest approach for application. Therefore, this category is considered for attempting to improve the predictive performance of the surrogate for the specific target variable.
Pre-processing of the data can be done in several ways, however, the simplest form of pre-processing is the re-sampling approach. This approach is also the one considered for this research. Furthermore, the re-sampling strategy for regression is structured around a set of elements: **utility-based regression**, **precision & recall** and the **re-sampling approach** itself. The elements are further elaborated in this section [49].

3.1.1. Utility-based regression

The utility-based regression is a proposal for evaluating a predictive model on its utility [47]. The utility is the balance between the costs and benefits on the prediction of the model. In this subsection, the utility-based regression method is further discussed.

Relevance function

In some applications, the relevance (or importance) of the target variable is not uniform over its domain. Hence, the relevance can be mapped through the relevance function. The relevance function $\phi(y)$ is a continuous function mapping the target variable in a [0,1] domain with 1 being the most relevant values. The user can specify the function as he deems most relevant for the problem. Here, the relevance is chosen to be inversely proportional to the density function of the target variable. With an inversely proportional relevance function, rare/extreme values are considered as most relevant for the problem. An example is given in Figure 3.1.

From Figure 3.1, it can be seen that the values close to the median are equal to zero (or almost zero). Furthermore, the relevance is increasing towards the third quartile of the target variable, reaching a value of 1 for values beyond the third quartile (outliers). The function is based on a monotonic piecewise cubic spline.

Utility

The utility score, from the utility-based regression framework, is based on the net balance between costs and benefits. The prediction is beneficial if it leads to the correct type of extreme value [47]. In addition, the benefit is also related to the accuracy of the predicted value. Hence, the cost can be determined with the following equation [47],

$$B_\phi\left(\hat{y}, y\right) = \phi(y) \cdot \left(1 - \Gamma_B\left(\hat{y}, y\right)\right) \tag{3.1}$$

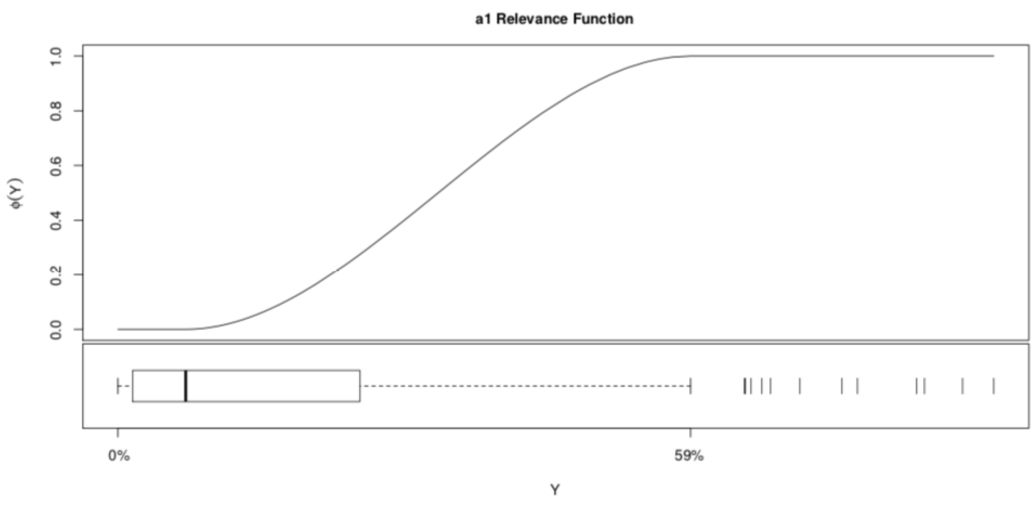


Figure 3.1: An example of the relevance function [8]

where y is the true value, $\hat{y}$ is the predicted value, $\phi(y)$ is the relevance function and $\Gamma_B\left(\hat{y}, y\right)$ is the bounded loss function for benefits. The latter is zero when the the target variable is perfectly predicted. Γ_B ensures that the benefits are zero when either the value is not perfect or when the type of predicted is not coinciding with the type of the true value (e.g. low or high).
The cost, in contrast with the benefits, depends on both the relevance of both the predicted and true value. The joint relevance is defined by the following equation,

$$\phi^P\left(\hat{y}, y\right) = (1-p)\cdot\phi\left(\hat{y}\right) + p\cdot\phi(y) \tag{3.2}$$

where p ([0,1]) is a factor differentiating the types of errors. Equation 3.2 is weighted average of the relevance of both the true and predicted value. Thus, the mistakes on the prediction can be: false alarms, missed events and confusing events. The cost, making use of the joint relevance in Equation 3.2, is given by the following equation,

$$C_\phi^p\left(\hat{y}, y\right) = \phi^p\left(\hat{y}, y\right)\cdot\Gamma_C\left(\hat{y}, y\right) \tag{3.3}$$

where Γ_C is the bounded loss function for costs. Similar to Equation 3.1, Γ_C ensures the costs are zero for a perfect prediction and increases up to 1 for a prediction moving away from the true value.
Last, as mentioned earlier in this section, the utility is the net balance between the benefit and cost for a prediction. The utility is given by the following equation,

$$U_\phi^p\left(\hat{y}, y\right) = B_\phi\left(\hat{y}, y\right) - C_\phi^p\left(\hat{y}, y\right) \tag{3.4}$$

3.1.2. Precision and recall

Precision and recall are two standard metric used for evaluation of classification models in a skewed domain. For a classification problem, the target is identify the events belonging to the minority class. Precision is then the number of events given by the model that are real events and recall is the number of events occurring that are captured by the model.
Within the utility-based regression framework, both metrics are defined in relation with the utility and relevance of the target variable. Both metrics are defined by the following equations [46],

$$recall = \frac{\sum_{i:\hat{z}_1=1,z_i=1}(1+u_i)}{\sum_{i:z_i=1}\left(1+\phi(y_i)\right)} \tag{3.5}$$

$$precision = \frac{\sum_{i:\hat{z}_1=1,z_i=1}(1+u_i)}{\sum_{i:\hat{z}_1=1,z_i=1}\left(1+\phi(y_i)\right) + \sum_{i:\hat{z}_1=1,z_i=0}\left(2-p\left(1-\phi(y_i)\right)\right)} \tag{3.6}$$

where p is a factor differentating the type of errors and $\hat{z}$ and z are binary variable indicating the presence of a rare case. Ultimately, Equation 3.5 and Equation 3.6 can be comcined into a signle measure, namely the F_1 measure. The measure is defined as follows,

$$F = \frac{(\beta^2 + 1) \cdot precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{3.7}$$

where β is a weight factor between precision and recall. Having $\beta = 1$ means that an equal importance is given to the precision and recall.

3.1.3. Re-sampling approaches

Different approaches are possible for redistributing the target variable in order to reduce the imbalance character of the distribution. All based on re-sampling the existing data set. The three main approaches for regression problems are: **under-sampling** [49], **SmoteR** [48] and **SMOGN** [9] where SmoteR is a type of over-sampling strategy and SMOGN is a combination of under- and over-sampling. Under-sampling is considered for the sake of simplicity. The basis, however, of dividing the data set is common to the three re-sampling approaches. The set is divided on the following condition into rare and normal cases[49],

$$D_r = \left\{ \langle x, y \rangle \in D : \phi(y) \geq t \right\} \tag{3.8}$$

where D_r are the relevant (rare) cases and t is a user-defined threshold. Furthermore, the amount of under-sampling is defined by the ratio $D\ D_r$. The ratio determines how many points are randomly sampled from the set of "normal" (irrelevant) cases.

Results

Results obtained from applying the re-sampling approach are given below. One case is presented, however, none of the results increased the prediction accuracy of the surrogate model.

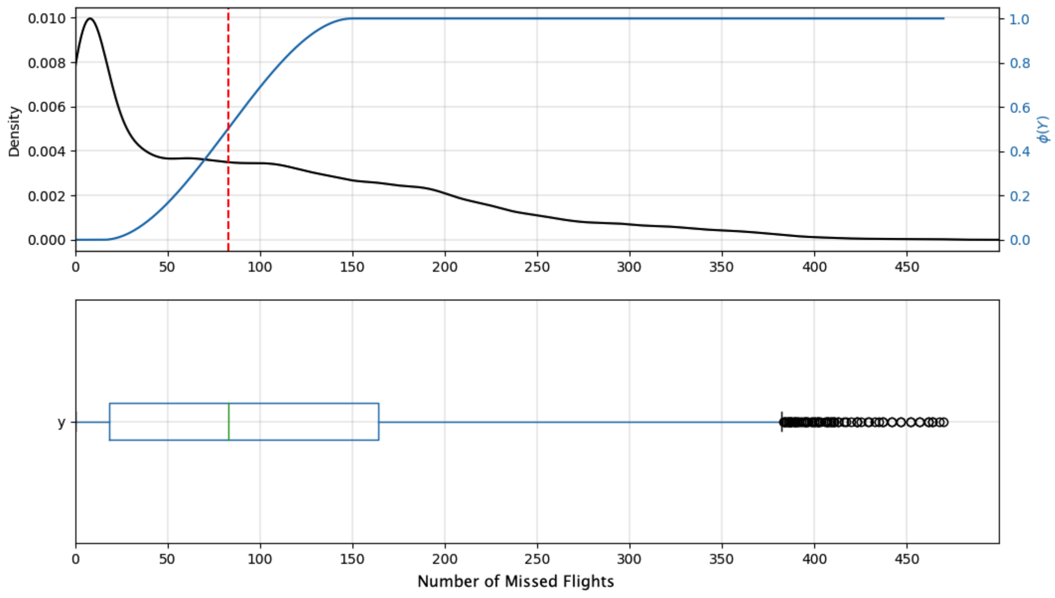


Figure 3.2: Relevance function for $MissedFlights$.

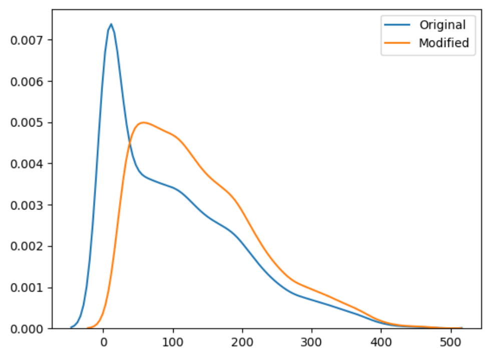


Figure 3.3: Distribution of the target variable $MissedFlights$ before and after re-sampling.

Table 3.1: Re-sampling parameters with the input of the monotonic piecewise cubic spline, the threshold t and the the ratio normal and rare cases.

Parameters	Value
Spline input	[0, 0], [15, 0], [150, 1]
t	0.01
D/D_r	15%

Table 3.2: Performance of the balanced default RF.

Performances	**Default RF Balanced**
Training time [sec]	1.50
Execution time [sec]	0.11
CheckinQueue (MAPE)	95.37%
ScQueue (MAPE)	95.90%
TimeToGate (MAPE)	96.93%
MissedFlights (MAPE)	**69.89%**
CheckinPassed (MAPE)	93.67%
ScPassed (MAPE)	94.80%
Mean (MAPE)	91.09%

Bibliography

[1] Fasihul M. Alam, Ken R. McNaught, and Trevor J. Ringrose. A comparison of experimental designs in the development of a neural network simulation metamodel. Simulation Modelling Practice and Theory, 12(7):559–578, 2004. ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2003.10.006. URL http://www.sciencedirect.com/science/article/pii/S1569190X04000747.

[2] Javier Arroyo, Samer Hassan, Celia Gutiérrez, and Juan Pavón. Re-thinking simulation: a methodological approach for the application of data mining in agent-based modelling. Computational and Mathematical Organization Theory, 16(4):416–435, 2010. ISSN 1572-9346. doi: 10.1007/s10588-010-9078-y. URL https://doi.org/10.1007/s10588-010-9078-y.

[3] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In International conference on machine learning, pages 115–123. PMLR.

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. Advances in neural information processing systems, 24:2546–2554, 2011.

[5] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In International conference on machine learning, pages 115–123. PMLR, 2013.

[6] Gérard Biau and Erwan Scornet. A random forest guided tour. Test, 25(2):197–227, 2016. ISSN 1133-0686.

[7] Anne-Laure Boulesteix, Silke Janitza, Jochen Kruppa, and Inke R König. Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(6):493–507, 2012. ISSN 1942-4787.

[8] Paula Branco. Re-sampling approaches for regression tasks under imbalanced domains. Unpublished Master's Thesis), Dep. Computer Science, Faculty of Sciences-University of Porto, 2014.

[9] Paula Branco, Luís Torgo, and Rita P Ribeiro. Smogn: a pre-processing approach for imbalanced regression. In First international workshop on learning with imbalanced domains: Theory and applications, pages 36–50. PMLR. ISBN 2640-3498.

[10] Paula Branco, Luis Torgo, and Rita Ribeiro. A survey of predictive modelling under imbalanced distributions. arXiv preprint arXiv:1505.01658, 2015.

[11] K. Crombecq, L. De Tommasi, D. Gorissen, and T. Dhaene. A novel sequential design strategy for global surrogate modeling. In Proceedings of the 2009 Winter Simulation Conference (WSC), pages 731–742, 2009. doi: 10.1109/WSC.2009.5429687.

[12] Carla Currin, Toby Mitchell, Max Morris, and Don Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. Journal of the American Statistical Association, 86(416):953–963, 1991. ISSN 01621459. doi: 10.2307/2290511. URL www.jstor.org/stable/2290511.

[13] E. Rumelhart David and L. McClelland James. Learning Internal Representations by Error Propagation, pages 318–362. MITP, 1987. ISBN 9780262291408. URL http://ieeexplore.ieee.org/document/6302929.

[14] Ramón Díaz-Uriarte and Sara Alvarez de Andrés. Gene selection and classification of microarray data using random forest. BMC Bioinformatics, 7(1):3, 2006. ISSN 1471-2105. doi: 10.1186/1471-2105-7-3. URL https://doi.org/10.1186/1471-2105-7-3.

[15] John Eason and Selen Cremaschi. Adaptive sequential sampling for surrogate model generation with artificial neural networks. Computers & Chemical Engineering, 68:220–232, 2014. ISSN 0098-1354.

[16] Mert Edali and Gönenç Yücel. Comparative analysis of metamodeling techniques based on an agent-based supply chain model. In ECMS, pages 114–120, .

[17] Mert Edali and Gönenç Yücel. Comparative analysis of metamodeling techniques based on an agent-based supply chain model. In ECMS, pages 114–120, .

[18] Mert Edali and Gönenç Yücel. Exploring the behavior space of agent-based simulation models using random forest metamodels and sequential sampling. Simulation Modelling Practice and Theory, 92: 62–81, 2019. ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2018.12.006. URL http://www.sciencedirect.com/science/article/pii/S1569190X18301941.

[19] Kai-Tai Fang, K. J. Lin Dennis, Peter Winker, and Yong Zhang. Uniform design: Theory and application. Technometrics, 42(3):237–248, 2000. ISSN 00401706. doi: 10.2307/1271079. URL www.jstor.org/stable/1271079.

[20] D. J. Fonseca, D. O. Navaresse, and G. P. Moynihan. Simulation metamodeling through artificial neural networks. Engineering Applications of Artificial Intelligence, 16(3):177–183, 2003. ISSN 0952-1976. doi: https://doi.org/10.1016/S0952-1976(03)00043-5. URL http://www.sciencedirect.com/science/article/pii/S0952197603000435.

[21] Alexander I. J. Forrester and Andy J. Keane. Recent advances in surrogate-based optimization. Progress in Aerospace Sciences, 45(1):50–79, 2009. ISSN 0376-0421. doi: https://doi.org/10.1016/j.paerosci.2008.11.001. URL http://www.sciencedirect.com/science/article/pii/S0376042108000766.

[22] Jerome H. Friedman. Multivariate adaptive regression splines. The Annals of Statistics, 19(1):1–67, 1991. ISSN 00905364. URL www.jstor.org/stable/2241837.

[23] Simon S. Haykin. Neural networks : a comprehensive foundation. Prentice Hall, Upper Saddle River, N.J., 1999. ISBN 0132733501 9780132733502 0139083855 9780139083853.

[24] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An Introduction to Statistical Learning. 2013. ISBN 1461471389.

[25] Stef Janssen, Alexei Sharpanskykh, Richard Curran, and Koen Langendoen. Using causal discovery to analyze emergence in agent-based models. Simulation Modelling Practice and Theory, 96: 101940, 2019. ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2019.101940. URL http://www.sciencedirect.com/science/article/pii/S1569190X19300735.

[26] R. Jin, W. Chen, and T. W. Simpson. Comparative studies of metamodelling techniques under multiple modelling criteria. Structural and Multidisciplinary Optimization, 23(1):1–13, 2001. ISSN 1615-1488. doi: 10.1007/s00158-001-0160-4. URL https://doi.org/10.1007/s00158-001-0160-4.

[27] M. E. Johnson, L. M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. Journal of Statistical Planning and Inference, 26(2):131–148, 1990. ISSN 0378-3758. doi: https://doi.org/10.1016/0378-3758(90)90122-B. URL http://www.sciencedirect.com/science/article/pii/037837589090122B.

[28] R. A. Kilmer, Alice Smith, and Larry Shuman. An emergency department simulation and a neural network metamodel. Journal of the Society for Health Systems, 5:63–79, 1997.

[29] Jack P. C. Kleijnen. Kriging metamodeling in simulation: A review. European Journal of Operational Research, 192(3):707–716, 2009. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2007.10.013. URL http://www.sciencedirect.com/science/article/pii/S0377221707010090.

[30] Jack P. C. Kleijnen and David Deflandre. Validation of regression metamodels in simulation: Bootstrap approach. European Journal of Operational Research, 170(1):120–131, 2006. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2004.06.018. URL http://www.sciencedirect.com/science/article/pii/S0377221704004576.

[31] Jack P. C. Kleijnen and Robert G. Sargent. A methodology for fitting and validating metamodels in simulation. European Journal of Operational Research, 120(1):14–29, 2000. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(98)00392-0. URL http://www.sciencedirect.com/science/article/pii/S0377221798003920.

[32] Jack PC Kleijnen. Design and analysis of simulation experiments. In International Workshop on Simulation, pages 3–22. Springer.

[33] Jack PC Kleijnen, Susan M Sanchez, Thomas W Lucas, and Thomas M Cioppa. State-of-the-art review: a user's guide to the brave new world of designing simulation experiments. INFORMS Journal on Computing, 17(3):263–289, 2005. ISSN 1091-9856.

[34] Francesco Lamperti, Andrea Roventini, and Amir Sani. Agent-based model calibration using machine learning surrogates. Journal of Economic Dynamics and Control, 90:366–389, 2018. ISSN 0165-1889. doi: https://doi.org/10.1016/j.jedc.2018.03.011. URL http://www.sciencedirect.com/science/article/pii/S0165188918301088.

[35] Ju-Sung Lee, Tatiana Filatova, Arika Ligmann-Zielinska, Behrooz Hassani-Mahmooei, Forrest Stonedahl, Iris Lorscheid, Alexey Voinov, J. Gareth Polhill, Zhanli Sun, and Dawn C. Parker. The complexities of agent-based modeling output analysis. Journal of Artificial Societies and Social Simulation, 18(4):4, 2015. ISSN 1460-7425. doi: 10.18564/jasss.2897. URL http://jasss.soc.surrey.ac.uk/18/4/4.html.

[36] Y. F. Li, S. H. Ng, M. Xie, and T. N. Goh. A systematic comparison of metamodeling techniques for simulation optimization in decision support systems. Applied Soft Computing, 10(4):1257–1273, 2010. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2009.11.034. URL http://www.sciencedirect.com/science/article/pii/S1568494609002841.

[37] L. Little C. Edmonds B. Lessard-Phillips and Fieldhouse Edward. Analysing a complex agent-based model using data-mining techniques, 2014.

[38] 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, 2018. ISSN 1615-1488.

[39] Martin Meckesheimer, Andrew J Booker, Russell R Barton, and Timothy W Simpson. Computationally inexpensive metamodel assessment strategies. AIAA journal, 40(10):2053–2060, 2002. ISSN 0001-1452.

[40] Raymond H. Myers, Douglas C. Montgomery, and Christine M. Anderson-Cook. Response Surface Methodology : Process and Product Optimization Using Designed Experiments. John Wiley & Sons, Incorporated, New York, UNITED STATES, 2016. ISBN 9781118916025. URL http://ebookcentral.proquest.com/lib/delft/detail.action?docID=4312599.

[41] Mark JL Orr. Introduction to radial basis function networks. Technical Report, center for cognitive science, University of Edinburgh, 1996.

[42] Art B. Owen. Orthogonal arrays for computer experiments, integration and visualization. Statistica Sinica, 2(2):439–452, 1992. ISSN 10170405, 19968507. URL www.jstor.org/stable/24304869.

[43] Susan M. Sanchez and Thomas W. Lucas. Exploring the world of agent-based simulations: simple models, complex analyses, 2002.

[44] Janssen Stef, Blok Anne-Nynke, and Knol Arthur. Aatom - an agent-based airport terminal operations model. 2018.

[45] Stef Janssen. Aatom. URL https://github.com/StefJanssen/AATOM.

[46] Luis Torgo and Rita Ribeiro. Precision and recall for regression. In International Conference on Discovery Science, pages 332–346. Springer, .

[47] Luis Torgo and Rita Ribeiro. Utility-based regression. In European Conference on Principles of Data Mining and Knowledge Discovery, pages 597–604. Springer, .

[48] Luís Torgo, Rita P Ribeiro, Bernhard Pfahringer, and Paula Branco. Smote for regression. In Portuguese conference on artificial intelligence, pages 378–389. Springer.

[49] Luís Torgo, Paula Branco, Rita P Ribeiro, and Bernhard Pfahringer. Resampling strategies for regression. Expert Systems, 32(3):465–476, 2015. ISSN 0266-4720.

[50] Sander van der Hoog. Surrogate modelling in (and of) agent-based models: A prospectus. Computational Economics, 2018. doi: 10.1007/s10614-018-9802-0.

[51] Liesje Van Gelder, Payel Das, Hans Janssen, and Staf Roels. Comparative study of metamodelling techniques in building energy simulation: Guidelines for practitioners. Simulation Modelling Practice and Theory, 49:245–257, 2014. ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2014.10.004. URL http://www.sciencedirect.com/science/article/pii/S1569190X14001555.

[52] Nathalie Villa-Vialaneix, Marco Follador, Marco Ratto, and Adrian Leip. A comparison of eight metamodeling techniques for the simulation of n2o fluxes and n leaching from corn crops. Environmental Modelling & Software, 34:51–66, 2012. ISSN 1364-8152. doi: https://doi.org/10.1016/j.envsoft.2011.05.003. URL http://www.sciencedirect.com/science/article/pii/S1364815211001083.

[53] G Gary Wang and Songqing Shan. Review of metamodeling techniques in support of engineering design optimization. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, volume 4255, pages 415–426. ISBN 079184255X.

[54] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on bayesian optimizationb. Journal of Electronic Science and Technology, 17(1):26–40, 2019. ISSN 1674-862X. doi: https://doi.org/10.11989/JEST.1674-862X.80904120. URL http://www.sciencedirect.com/science/article/pii/S1674862X19300047.

[55] Yi Zhang, Zhe Li, and Yongchao Zhang. Validation and calibration of an agent-based model: A surrogate approach. Discrete Dynamics in Nature and Society, 2020:6946370, 2020. ISSN 1026-0226. doi: 10.1155/2020/6946370. URL https://doi.org/10.1155/2020/6946370.