# Machine learning based aircraft arrival / departure registrations

*Version of June 4, 2017*

Mike de Waard

# Machine learning based aircraft arrival / departure registrations

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Mike de Waard
born in Zwijndrecht, the Netherlands


Software Engineering Research Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

# Machine learning based aircraft arrival / departure registrations

Author: Mike de Waard
Student id: 4081064
Email: mikedewaard@gmail.com

**Abstract**

The aviation industry is vastly growing, as travelling by air is more common today than it ever was. However due too inefficiency and lack of communication of accurate flight information between airports, congestion and delays are occurring on a daily basis. While Collaborative Decision Making (CDM) is developed by Euro control to address this issue, the problem of transmitting accurate flight information near real time is not yet solved. Adecs Airinfra did a first attempt at automatic landing and departure registration by a fixed rule based algorithm to address this issue. However, this algorithm has limitations that cannot be solved with tweaking and tuning. In this work, we aim to create a replacement based on machine learning models. In this thesis we present the complete process, starting from raw real world data, turning this into labelled data up to the point where we define a validation method and present the final results. We managed to create a machine learning landing / departure detection system with up to 99% precision and recall for arrivals, and for departures we managed to get a precision of 94% against 98% recall.

Thesis Committee:

| | |
|---|---|
| Chair: | Prof. Dr. E. Meijer, Faculty EEMCS, TU Delft |
| University supervisor: | Dr. G. Gousios, Faculty EEMCS, TU Delft |
| Committee Member: | Prof Dr. A van Deursen, Faculty EEMCS, TU Delft |

# Preface

I've written this thesis as part of a research to support automation in the aviation industry. This was a combined effort between TU Delft University of Technology, Adecs Airinfra and me. I would like to thank everyone involved in this project for their support. First of all Erik Meijer, my supervisor for supporting me throughout the thesis and keeping me on my goal. Secondly, but not less important, Georgios Goussios for his guidance, time and effort throughout the thesis. Finally from the Delft university I'd like to thank Arie van Deursen for his time and effort while I was finalizing my work. From Adecs Airinfra, I want to thank Andy van Helden, and Peter Frankena for making this research possible by providing data sources and insights into the Airfee system. In addition I'd like to thank Zhi-Kang, Richard and Lars for their feedback throughout this research. Finally I'd like to thank my mother Marian, Jules and my friends for their support and patience.

Mike de Waard
Delft, the Netherlands
June 4, 2017

# Contents

**Preface** iii

**Contents** v

**List of Figures** vii

**1 Introduction** 1
- 1.1 Research Questions . . . 2
- 1.2 Context . . . 3
- 1.3 Outline . . . 4

**2 Background and Related work** 7
- 2.1 Background . . . 7
- 2.2 Related work . . . 9

**3 Method** 13
- 3.1 Problem identification . . . 13
- 3.2 Dataset preparation . . . 14
- 3.3 Feature creation and selection . . . 16
- 3.4 Validation and evaluation methods . . . 22
- 3.5 Algorithm selection . . . 23

**4 Results** 25
- 4.1 Original Algorithm . . . 25
- 4.2 Machine learning Models . . . 27
- 4.3 Results comparison . . . 29

**5 Conclusions and Future Work** 33
- 5.1 Conclusion . . . 33
- 5.2 Contributions . . . 34
- 5.3 Discussion . . . 34

5.4 Future work . . . . . 35

**Bibliography** **37**

**A Glossary** **41**

**B Measurement results** **43**

**C Visualisation of performance difference** **45**

# List of Figures


1.1 Radar vs ADS-B . . . . . 4
1.2 CDM Milestones . . . . . 5

3.1 Ground truth labeling UI . . . . . 15
3.2 Plotted altitude and signal-strength per event type . . . . . 17
3.3 Altitude vs signal strength for arrivals and departures . . . . . 18
3.4 Window size scores for arrivals and departures . . . . . 21

4.1 Precision and recall for arrivals of original algorithm plotted against $\tau$ . . . . . 25
4.2 Precision and recall for departures of original algorithm plotted against $\tau$ . . . . 26
4.3 Precision and recall for arrivals of model plotted against $\tau$ . . . . . 28
4.4 Precision and recall for departures of model plotted against $\tau$ . . . . . 29
4.5 Precision and recall for arrivals of model and algorithm plotted against $\tau$ . . . . 30
4.6 Precision and recall for departures of model and algorithm plotted against $\tau$ . . 31

C.1 Visualisation tool . . . . . 46

# Chapter 1

# Introduction

With the globalization of the world comes that the amount of aircraft traffic is vastly increasing. While in 2016, 3.8 billion people travelled by air, International Air Transport Association (IATA) expects the passenger demand will double in the upcoming 20 years up to 7.2 billion[1]. To support these vast amounts of passengers, the current network, as well as airport processes themselves have to be optimized. To allow for this optimization, accurate information has to be exchanged in a rapid manner. To solve the issue of standardization and communication, IATA and the International Civil Aviation Organization (ICAO) created standards to allow for exchanging information between airports. In addition, Eurocontrol introduced Collaborative Decision Making (CDM) 1.2.3 with co-financing of the European Union. CDM is a protocol for airport collaborative decision making which describes what information should be communicated at which times, and how accurate this information has to be. While this addresses the problem of communication, the challenge of acquiring this accurate information in a fast manner still exists up to today.

Currently, flight information is being tracked manually by ground handling service providers and air traffic control staff. This information is digitalized and ready to be sent to the next airport in the chain only 15 to 30 minutes later, as the staff also has to take care of handling the aircraft, and keeping the airspace safe. This delay in information sharing has a great impact on the decision making process of the following airports. While CDM aims at sharing accurate information fast, this cannot be done in the current setting without hiring more people to manually record and directly transmit this information to other airports. However, due to the growing market, and pricing pressure from airlines, this is infeasible. Another solution would be to automatically detect information, which then can be verified and sent to other airports by the press of a button. This significantly reduces the workload, while providing accurate flight information in a fast manner. The company Adecs Airinfra made a first attempt to automatically detect arrival and departures for aircrafts based on Automatic Dependent Surveillance-Broadcast (ADS-B) 1.2.2 data. This system consists of a fixed set of rules, taking 13 parameters and is currently operational at two Dutch Airports. The performance of this algorithm however fluctuates heavily due to its nature. To improve this algorithm, it would have to be redesigned from scratch based on practical experience gained over the last 4 years. However, coming up with a fixed set of rules that define an arrival or

departure given the ADS-B data feed is hard. In addition, these rules requires parameters that have to be tuned for every location. In contrast to a fixed algorithm, machine learning is known to extract its own rules from the dataset based on statistics. Model improvement is done by adding more/cleaner data to the dataset. This makes machine learning an interesting alternative to the rule based flight landing predictor.

In this work we aim at completely replacing the current algorithm with machine learning models that perform significantly better in not only precision and recall but also response time, while being based on statistics in comparison to some hand crafted fixed rules with parameters that need to be tuned manually.

## 1.1 Research Questions

In order to reach our goal of detecting arrivals and departures with machine learning, we define the following research question:

*Can machine learning be used for detecting aircraft arrivals and departures based on ADS-B data and how does it compare to an existing fixed rule based system?*

This main research question is broken into the following three sub-questions:

1. *How can machine learning be applied for detecting of arrivals and departures of aircrafts using ADS-B data?*
   Going from a real-world ADS-B data stream to a machine learning model brings challenges that have to be dealt with. For example, a machine learning model takes a fixed set of features, while our data feed puts out one point of data at a time. In addition, applying machine learning requires choices to be made for algorithms, features, and validation methods. By answering this research question we will identify which algorithm(s), features and validation method(s) to use for creating a time series based machine learning model for the prediction of arrivals/departures.

2. *How can the performance of the fixed rule based algorithm be measured and compared to a machine learning model?*
   While various validation methods exist for machine learning models, each method has their own benefits and risks. However, most of these validation methods do not take into account the aspect of time. In addition, not all of these methods can be used to compare a model to a fixed algorithm. By answering this question we define how to evaluate the performance of our time series machine learning model, and how to compare it to the original algorithm.

3. *How does the machine learning model perform in comparison to the fixed rule system?*
   Given the machine learning models, and a validation method, we can answer this question by evaluating the models and algorithm, which allows us to answer the main research question.

## 1.2 Context

Since this work combines aviation with machine learning, we use this section to clarify some core principles of aviation that we lean on in this work. Possibly unknown terms and definitions used throughout this work are explained in appendix A.

### 1.2.1 Turn-around

In the field of aviation, air traffic control, ground handling, airlines and various other stakeholders are involved in making travelling by air possible. There are many different processes involved, but in this work we are looking in particular at the process of landing and departing, which are the first and last step in the process of making a turn-around. A turn-around is the process that an aircraft goes through from the moment of landing, up until the moment of departing. The turn-around time is the time it takes for the process to be completed, and is widely used by airlines to measure the performance of an airport. Depending on the airline, turn-around times can range from 45 down to even 20 minutes. This means that in the lower case, an aircraft has a total of 20 minutes to have all service done, passengers and baggage unloaded, new passengers and baggage loaded and is ready for take off. To meet these times, receiving accurate flight information is key, as preparations for the services take time.

### 1.2.2 Automatic Dependent Surveillance-Broadcast system

The Automatic Dependent Surveillance-Broadcast (ADS-B) system is an alternative on the traditional radar system for determining the position of an aircraft in the air. Figure 1.1 shows how a radar communicates with an aircraft by sending out a signal, at which the aircraft then responds. In europe, the ADS-B transponders, also referred to as mode-S transponders are mandatory for new aircrafts since 2015. In addition, all registered aircrafts have to be equipped with these transponders by 2020. In contrast to radar, ADS-B transponders transmit their information every second, and can be freely picked up by any ADS-B receiver antenna. These receivers operate on 1090MHZ and people even succeeded in building these antennas with a raspberry pi for under 100 dollar [2]. This is a phenomenal difference in price when compared to radar systems which start at 350k.

The information ADS-B transponders broadcast consist of an aircraft identifier and altitude. In addition, the ADS-B transponders can be linked to GPS receivers with a cable to send out their location in GPS coordinates. However, this link between GPS and the transponder is not mandatory for most aircrafts, causing this information to only be sporadically available. In addition, the precision of ADS-B transponder altitudes depend on the manufacturer. While some transponders are precise up to 33 feet (10.06 meter), others are precise up to 100 feet (30.48 meter). These step sizes can cause the signal to toggle constantly between two altitudes, while the aircraft is flying level.

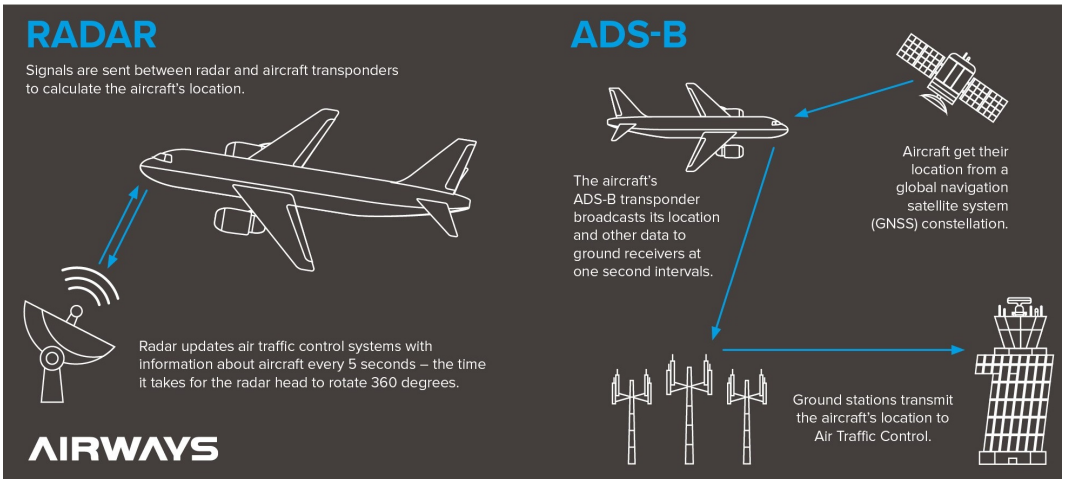


Figure 1.1: Radar vs ADS-B ([3])

### 1.2.3 Collaborative Decision Making (CDM)

Currently, airports have limited cooperation. In addition even at airports themselves, co-operation between the individual parties such as the Air Traffic Control (ATC) tower and ground handling is limited. Airport CDM is a Eurocontrol initiative to improve the overall efficiency of airport operations by optimizing the use of resources and improving the predictability of events [4]. This initiative is co-founded by the European union. Part of this initiative, is to use standardized definitions for information sharing such as milestones. Figure 1.2 displays these milestones in order from 1 up to 16. In this work we use landing (milestone 6) as the definition for an arrival. The time corresponding to this milestone is called the ALDT (Actual Landing Time). The departure definition throughout this work is milestone Take Off (milestone 16). The corresponding time for this milestone is the ATOT (Actual Take off time). Note that for a flight departing from Airport A to airport B, milestone 16 of Airport A, is identical to milestone 3 for Airport B.

## 1.3 Outline

The rest of this document is structured as follows: in chapter 3 we answer research question 1 by working through the process of going from raw data to a machine learning model for arrivals and departures. In addition we answer research question 2 in this chapter by defining a validation method that works for both the original algorithm and the machine learning models, such that they can be measured in the same way. In chapter 4 we answer research question 3 by presenting the results and making a comparison between the original algorithm and machine learning models. In chapter 5 we recap the research questions, discuss our work and present possible future work. Finally, uncommon terms are explained in appendix A.

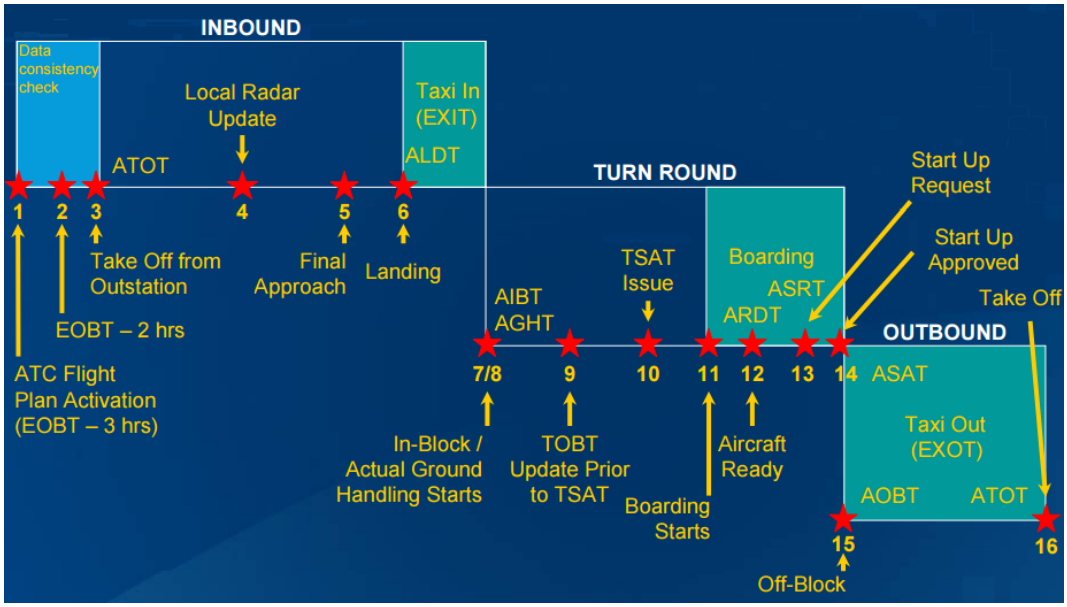


Figure 1.2: CDM Milestones ([5])

# Chapter 2

# Background and Related work

In this chapter we describe the background of the original algorithm, how this algorithm works and what its core design flaws are. In the second part of this chapter we present other work aimed at improving the aviation industry. Finally we refer to related researches based on the techniques and nature of our data, and motivate how we use those techniques in this work.

## 2.1 Background

The aviation sector is rapidly growing due to demand caused by for example globalization. This drives airlines into enforcing strict turn-around time rules on airports with high fines in case they are not met. The turn around time of an aircraft is the time it takes from the moment of landing until departing again. In order to meet these turn around times, airports require detailed information about the flights arrival and departure times. For example, if an aircraft arrives at airport Alpha, with destination airport Beta, which is an hour away, preparations should already start at airport Beta the moment this aircraft landed on Alpha. These preparations can range from preparing the fuelling trucks, de-icing tanks and catering services. On the other hand, the same services have to be applied on other aircrafts as well, and the time-frames for performing each of the services is minutes rather than hours. Due to this, it is important to have accurate information available as soon as possible. However, this information is currently manually registered and reported to the next airport with delays of up to 30 minutes. These delays cause the next airport to be planning based on scheduled information rather than real time information. This causes inefficient use of resources and last minute changes for the operational staff.

In addition to this, airports need to keep track of the amount of movements (arrivals/departures) they handle throughout the year for environmental and financial reports to the government. This is currently tracked by hand and verified manually against schedules and flight packs for commercial flights. For flights such as training, recreational, medical and those for private transport this verification can't be done as these are often not scheduled, and do not have these flight packs available. This makes tracking the amount of movements

| Parameter | Description |
|---|---|
| EMFRC | maximum possible rate of climb for all aircrafts |
| EAOS | manual time correction in seconds for arrivals |
| EDOS | manual departure time correction in seconds for departures |
| LBFD | altitude which the aircraft has to be been below to be allowed to be marked as departure |
| ALFD | altitude which the aircraft has to go through to be marked as departure |
| STS | minimum amount of data needed by the algorithm to allow for a departure / arrival detection |
| ENGB | altitude that the aircraft has to be below to allow for marking as arrival |
| ALDL | altitude which the aircraft has to go through to be marked as arrival |
| EMTBDD | minimal time that has to pass from the former departure to allow for new departure detection |
| EMTBAA | minimal time that has to pass from the former arrival to allow for new arrival detection |
| EMTBAD | minimal time that has to pass from the former arrival to allow for new departure detection |
| EMTBDA | minimal time that has to pass from the former departure to allow for a new arrival detection |
| EMSSTBL | minimally required signal-strength to allow for arrival or departure detection |

Table 2.1: AirFee parameters

at an airport for these flights resource intensive and error prone due too all the manual work. Automatic registration can significantly reduce the workload of these mandatory reports.

Adecs Airinfra designed a fixed-rule algorithm that operates on ADS-B data from individual aircrafts. This algorithm contains 13 different parameters that have to be tuned. We listed each of the parameters including a description on how they are used in table 2.1.While most of these parameters are to check if the data is even a candidate for arrival or departure, the most important parameters are ALDL (altitude for landing) and ALDD (altitude for departure). These two parameters are the foundation of the algorithm, whereas the other parameters were introduced over time as patches to problems with the algorithm. The core idea of the algorithm is to detect an event (arrival or departure) based on the aircraft going through a certain altitude level. However, this brings a challenge, as the altitude of an aircraft is computed based on barometric pressure, which fluctuates with weather. While these fluctuations are in the range of approximately 300 feet, this causes the altitude boundary for altitudes and departures to be placed on the upper bound of these altitudes. In other words, on a good weather day, arrivals and departures may be detected 10's of seconds earlier or later than on a bad weather day. This is strongly undesirable, as detection should be near real time for verification purposes. This real time registration is important, as one person of operations is constantly monitoring the runway and this system for errors, and to add

in additional data. This tracking becomes hard when flights are registered with delays of 30 to 60 seconds, as there might already be another aircraft on the runway by that time. In addition, the person verifying is responsible for several other aspects, making this a side-job that should not be resource intensive.

Parameter EMSSTBL (minimum required signal-strength) was introduced to ignore aircrafts arriving at other airports, or flying particularly low for example for aerobatics or training. Prior to the introduction of this parameter, these cases were also detected as arrivals and departures, due to the wide range of the ADS-B signal, which can receive data from aircrafts up to 200 km away, depending on the altitude and strength of the ADS-B transponder on the vehicle itself.

Given this patching over time, the algorithm has grown to some kind of black box where the impact of tweaking and adding of rules cannot be overseen easily. To make things more difficult, a direct performance measurement method for the algorithm does not currently exist. Creating a new rule based algorithm is hard as coming up with rules that encapsulate near real time detection is non trivial, if not impossible. In contrast, machine learning is widely used for pattern recognition and extracting underlying statistical rules from data for detection purposes. In addition, machine learning models can be improved by training it with corrections for false positive and false negative cases [6][7]. This makes machine learning an suitable alternative for a fixed rule based algorithm.

## 2.2 Related work

Our research is not the first attempt at making more accurate flight information available through machine learning. In 2014, Ravizza et al. performed a collaborative research for aircraft taxi time predictions [8]. In this work they compared different forms of regression to determine the taxi time in minutes for aircrafts based on historical airport data. In figure 1.2, the arrival taxi time is defined as the time between landing (6) and In-Block (7). For a departure the taxi time is defined as the time between Off-Block (15) and Take Off (16). Note that our work focusses on predicting milestone 6 and 16, rather than these two intervals, but can complement the research of Ravizza et al. for arrivals by providing milestone 6 for their regression algorithm.

Aircrafts, like any other vehicle need maintenance, especially when a rough landing has occurred. X. Wang et al. did a research on detecting these kinds of landings [9]. Their main goal was to allow for reporting of preventive maintenance after a hard landing, such that minor issues are addressed before they become major issues. To detect a rough landing, they collected accelerometer data from landings, and used domain knowledge to label this data. They then trained a Support Vector Machine (SVM) using a training split of the original data, and did parameter tuning by running for a range of parameters against the test set. By this method, they managed to get a 97% accuracy for detecting rough landings. Where they can detect the roughness of a landing on board of an aircraft, our work differs in that we

attempt to detect time rather than roughness. In contrast, we do not require any additional device to be on board of an aircraft, whereas the accelerometer data is not transmitted via for example ADS-B. This allows our work to be deployed without need of access to the actual vehicles involved.

Using time series sensor data in machine learning is interesting in many different research fields. One of the most popular examples of this is activity recognition based on accelerometer data from a mobile phone [10]. The main difference with our work however, is that their recognition is done based on a repetitive pattern. For example, one takes a step, which takes approximately 1.2 seconds, and this pattern is repeated many times to detect that someone is walking. In our case, an arrival or departure is a pattern that only occurs once in a time frame of minutes, sometimes even hours or days. In other words, where with recognizing walking, the model has a series of chances to detect the pattern, the model has only one shot in our case. Identically to the detection of accelerometer data, a team of researchers from Norway and Spain published a research on detecting cardiac arrest based on cardiogram data. [11]. What we find interesting is that both researches conclude that K-NN outperforms other algorithms.

Our data source is a raw ADS-B data-stream from an on-site receiver that is provided by Adecs Airinfra. We have full access to this stream such that we can freely collect data. We know in advance that our neutral data class will be several orders of magnitude larger. This means that our data is extremely imbalanced. Garcia et al. have conducted a large scale research, on performing machine learning on imbalanced data [12]. In this work they summarize state of the art techniques for dealing with this imbalanced data, and review evaluation metrics in machine learning and the effect of imbalanced data on these metrics. In this evaluation research they combined knowledge of over 140 different papers on machine learning, imbalance in data and evaluation metrics. Their research greatly helps us in performing balancing techniques as well as picking the evaluation metrics, based on which we will measure the performance of our machine learning models against the original algorithm.

Given our ADS-B data stream, we also know that we only have raw data points available as possible features to start with. These points contain an underlying structure that can be modelled into separate features. R. Chatterjee did a talk on extracting features from time series data [13]. In this talk he presents the process of extracting features from time series data to predict possible problems with IT systems. While we are not attempting to detect anomalies in IT systems, his work helps us defining features based on time series data, as using only raw data might not be enough information to perform proper classification. In contrast, Geurts research [14] shows that K-NN with K =1, and raw data outperforms constructed features and other algorithms such as boosting. This gives an interesting contrast, on which we based our feature selection method.

Using all possible features can result in the curse of dimensionality. This is the commonly used name for having too many features for too small of a dataset. This was argued by Ver-

leysen et al. in their work on the curse of dimensionality in time series classification [15]. Fortunately, this is now a well known problem, and various feature selection processes have been explored. Chandrashekar et al.[16] did an extensive survey on these methods, including exploration of 83 publications. In their work they show that having more features is certainly not always better. In addition they elaborate on three very common feature selection techniques: filtering, wrapping, and embedded feature selection. We use their extensive research as the base for picking our feature selection method for model creation. In addition we use the book feature selection for knowledge discovery[17] by Lui et al. for verification purposes.

# Chapter 3

# Method

To go from a raw, real-world data source to a machine learning model requires a variety of aspects to be covered. From data preparation, algorithm selection, to feature composition and last but not least validation metrics and evaluation method(s). Our main research question was split down into three sub questions. In this chapter we answer the first question by describing the methods used to come from raw, real world ADS-B data to a machine learning model. In addition we answer the second question by describing a validation method that allows for comparison of the fixed rule based algorithm against the machine learning models. This allows us to answer our final sub question in chapter 4 such that we can answer the main research question.

## 3.1 Problem identification

The rule based system reports for each received data point whether it is an arrival, departure or neutral point. The time-stamp of a data point is then used as the time that belongs to the detected arrival/departure. Since we are working on a replacement of the rule based system, the resulting output has to contain the same information. In other words, a data point should be marked as an arrival / departure / neutral. This is seen in the machine learning field as a classification problem. A classification problem is that of labelling new data based on prior labelled data. This makes classification belong to the category of supervised-learning types, as labels are used in the training dataset as well. Since a time aspect is involved, the exact problem type can be identified as time-series classification, which comes with an extra set of challenges which we will elaborate on in section 3.3. Since the possible classes are either arrival, departure or neutral, the problem also belongs to non-binary classification. We can however reduce the problem to binary classification by creating a separate model for each respective class. This allows for feature selection per event type, but introduces the possibility that 1 data point is marked as both arrival and departure at the same time. In practice this is very unlikely as the characteristics of an arrival such as descending, and coming closer to the airport are opposite of a departure which is ascending and moving away from the airport. Due to this we claim that separate models make more sense as they allow for feature selection per event type, and for more specific fine tuning for each model.

## 3.2 Dataset preparation

For this work we have collected a dataset of 35.4 million ADS-B messages from the stream provided by Adecs Airinfra. We collected this data in a time frame of 4 months. The ADS-B receiver is a TRX-1090 traffic receiver [18] and a message contains an aircraft identifier, altitude (feet), signal-strength (dB), time-stamp. In very rare cases a latitude and longitude are included, but due too the rarity we cannot use this as a feature. The total amount of aircrafts for which messages are contained in the dataset is 10019. We could choose to create a machine learning model for each individual aircraft. However, this is difficult due to the limited data available for some aircrafts, and this is undesirable as whenever an aircraft arrives or departs for the first time, it won't be automatically detected. Due to this, we will use the complete dataset while generalizing over all aircrafts. If a particular type of aircraft, say for example gliders, come forward as wrongly detected in the final results, we will revisit this decision and look for an alternative.

Given that our data is unlabelled we can't directly use it for supervised learning. Fortunately, field experts from the airport and Adecs Airinfra have offered to help labelling this dataset. Performing labelling on a dataset of 35.4 million records is highly infeasible, thus filtering prior to the labelling process has to be done. Since the dataset contains all received ADS-B data, and is not limited to aircrafts that departed or arrived at the airport, we can perform an informed filtering step to filter out aircrafts that do not have an altitude lower than flight level 010 which corresponds to 1000 feet. This leaves aircrafts in the dataset that have data both above and below flight level 010. The resulting dataset of this first filtering step consists of a total of 4.8 million messages. If we only present the data below 1000 feet of these aircrafts and automatically mark everything above 1000 feet as neutral, a total of 932609 points has to be viewed and labelled manually. To identify whether it is feasible to label this data, we plot a graph for one of the aircrafts with a window size of 200 points. A quick test for a single aircraft showed that with adding shortcut-keys for stepping forward, and marking arrivals/departures it would take a few days of work to go through all data, which was accepted by the field experts. We provided the field experts with the tool as seen in figure 3.1, where one can select an aircraft, click a point in the graph, and then apply the appropriate action depending on the point. The possible actions are, mark as arrival, mark as departure and move forward in time from this point. To significantly ease the process we added key bindings 'A', 'D' and 'M' for these commands.

The result of this process is a dataset containing 4555 arrivals, 3023 departures and 4.063.180 neutrals. The difference of +/- 800.000 in comparison to the 4.8 million messages we had after our first filtering step is explained by the removal of messages for aircrafts that did not land or depart from the airport but were not filtered out with the first step of filtering ou based on flight level 010. If we express these numbers in percentages, then the data consists of 0.112% arrivals, 0.074% departures, and 99.814% neutrals. This shows a severe imbalance of classes in the data, which is common in real world data, as a study by Torelli et al.[19] shows. They summarize reports of issues with imbalanced data in classification with real world cases. Garcia et al. summarized methods for dealing with imbalanced data

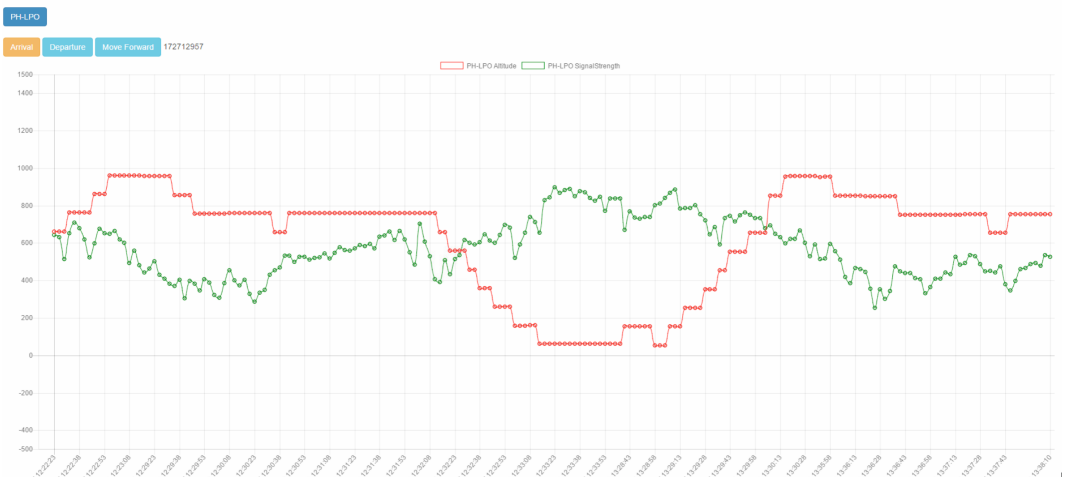


Figure 3.1: Ground truth labeling tool

in [12]. There are two important things to take into account when dealing with imbalanced data. First-off the evaluation metric(s), which we will address in section 3.4. And secondly the algorithm used for creating a model. We will discuss the algorithm choice in section 3.5. Even though we are aware of the main aspects to take into account with imbalanced data, we try to reduce the imbalance significantly for performance reasons. However, before we can do this, we first determine a size for test and training data, because the testing data should not be manipulated with any balancing techniques. Since we are working with time series data, a random train test split won't work as we need consecutive data points for feature modelling. Instead we will split the data in 2 time frames. By taking the first 2 months of data for training, and the latter 2 months for testing, we create a test dataset of 1221 arrivals, 829 departures and 1.171.250 neutrals. On the complete dataset this results in 26.80% of all arrivals, 27.36% of all departures and 28.88% of all neutrals for testing, while leaving 3334 arrivals, 2194 departures and 2891930 neutrals available for training purposes.

Now that we have separated the training and test set, we can apply balancing on the training set. We start by removing all data above Flight level 010, just like we did in the presentation of data to the field experts. This reduces the neutral class down to 876.633 points, while leaving the arrival and departures at the same level. The neutral class is still two orders of magnitude larger in comparison to the departures and arrivals. To deal with this, Garcia's work [12] discusses various methods. sub-sampling and oversampling are the most commonly used techniques in various forms. However, with oversampling, model over-fitting can occur, and sub-sampling brings the risk of removing important parts of the majority class. To deal with the risk of sub sampling, informed sub sampling was developed. In this way of sub sampling, the distribution of data is taken into account, limiting the risk of removing important data-points of the majority class. Since we do not want to have an overfitted model based on 2 months of data, we go with informed sub sampling on the neutral dataset in a ratio of 1 of 25. We do this informed sub sampling by taking

| Formula | Description | example |
|---|---|---|
| $T(x_n)$ | Absolute time-stamp | 2016-09-01 08:18:23.393 |
| $A(x_n)$ | Absolute altitude in feet, compensated for airport elevation | 300 ft |
| $F(x_n)$ | Altitude in feet, relative to pressure of 1013.25 hPa | 543 ft |
| $S(x_n)$ | Signal-strength in dB | 526 dB |

Table 3.1: Properties available on each data point

more neutral samples that lie close to the arrival or departure event while taking less neutral points that lay far away from these points, while maintaining a ratio of 1 to 25. With this random sub sampling we reduced the neutral class to 35605 points, which makes the complete training dataset 41133 points with a balance distribution of 8.2% arrivals, 5.3% departures and 86.5% neutrals. Balancing it further to equal class sizes is not necessary and can re-introduce the risk of removing important data of the majority class, which harms the performance of the resulting model. Since the ground truth labelling was performed by the field experts we need to verify the correctness of their work. To do so we performed 2-fold cross validation on the training and testing data. We then manually inspected false positives and false negatives for both runs. By performing this step we managed to correct for 112 errors in the complete arrival dataset, against a correction of 75 departures. These corrections were done in collaboration with the field experts, and consisted of failing to label an event or falsely labelling an arrival as departure or visa versa.

## 3.3 Feature creation and selection

In machine learning, the data usually already exists of a large set of features such as all pixel values of an image. This makes up for 3 values per pixel (RGB), resulting in a total of 30000 possible features in a picture of only 100x100 pixels. Our case is different in that one ADS-B message only brings us an altitude, time-stamp, signal-strength and aircraft identifier. We know on beforehand that a single point of this information will not suffice to predict properly, as for example the altitude of an arrival is the same as that of the upcoming points on the ground. Field experts used the trend in signal-strength and altitude together with flight plan information they had available to label the data, which tells us we need to somehow incorporate this data into features. In order to define these features we first need an understanding of the data we have available. For each individual aircraft, we have an ordered set of data-points $D$ where the most recent point is the first point in the set. The formal definition of this set is defined in formula 3.1. In this formula, $T(x)$ represents the time-stamp of point $x$ in the set.

$$D = \{x_n, x_{n-1}, ..., x_0\} \text{ where } \{\forall x_i \in D : T(x_i) > T(x_{i-1})\} \tag{3.1}$$

Given this set we can define features for point $x_i$ by combining the data of that point with points in the past. We already defined the time-stamp to be $T(x)$ but there are more properties available for each point, which we listed in table 3.1. Given these properties we can

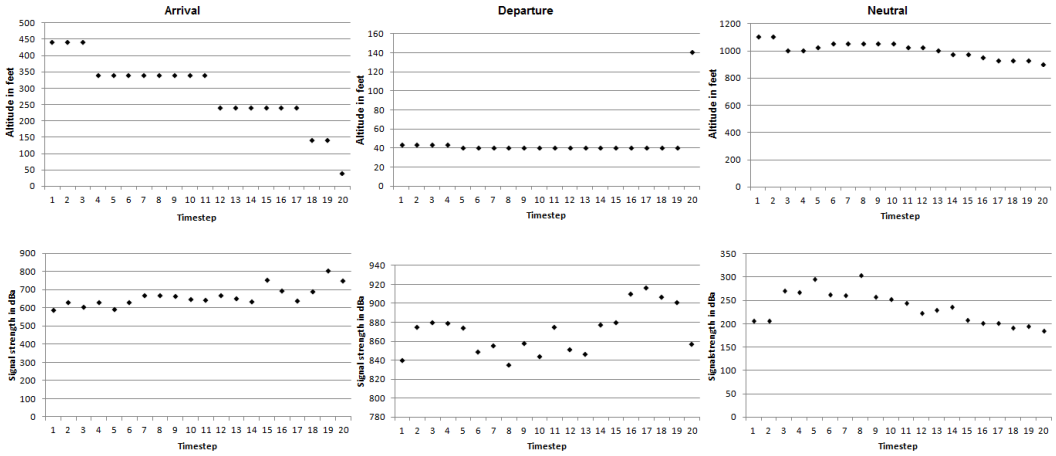


Figure 3.2: Plotted altitude and signal-strength per event type for 20 points

| | Altitude in feet | | | Signal-strength in dBa | | |
|---|---|---|---|---|---|---|
| | **Arrival** | **Departure** | **Neutral** | **Arrival** | **Departure** | **Neutral** |
| **Max** | 440,0 | 140,0 | 1105,0 | 803,0 | 916,0 | 304,0 |
| **Mean** | 290,0 | 45,6 | 1010,0 | 633,2 | 870,5 | 236,0 |
| **Min** | 40,0 | 40,0 | 905,0 | 587,0 | 835,0 | 186,0 |
| **Mode** | 340,0 | 40,0 | 1055,0 | 670,0 | 875,0 | 208,0 |
| **Peak to peak** | 400,0 | 100,0 | 200,0 | 216,0 | 81,0 | 118,0 |
| **Slope** | -16,5 | 1,3 | -8,5 | 7,1 | 1,9 | -3,6 |
| **Variance** | 85,0 | 9,4 | 47,5 | 39,8 | 19,9 | 30,1 |

Table 3.2: Metric calculations of examples from figure 3.2

now model the time series into features. For this there are various options as described by Chatterjee [13]. Some examples are taking the mean, slope, mode, minimum, maximum or even the signals variance. Since we want to have some idea before starting feature composition, we take one of each event type (arrival, departure and neutral) and plot the altitude and signal-strength in figure 3.2. These plots contain the actual event, and 19 points prior to this event. We then compute the options as mentioned by Chatterjee and list them in table 3.2. This table and the plots are not sufficient to do feature selection, as the depicted neutral case is actually one of the many options out there. For example, an aircraft can be moving further away or closer to the airport, and can ascend, stay steady or descend. In addition, we only took one arrival and departure of the set, which is not representative for the set. However, the plots and data do give some initial idea on what kind of data we are dealing with, which is important for the feature selection process. In addition to the in table 3.2 possible time series based features, we can also include the raw altitude and signal-strength values in the feature selection process. The latter part was argued to be giving worse results by Lines et al. [20]. However, we know from the original algorithm that multiple arrivals/departures in a row were detected as moving 1 step forward does not significantly change

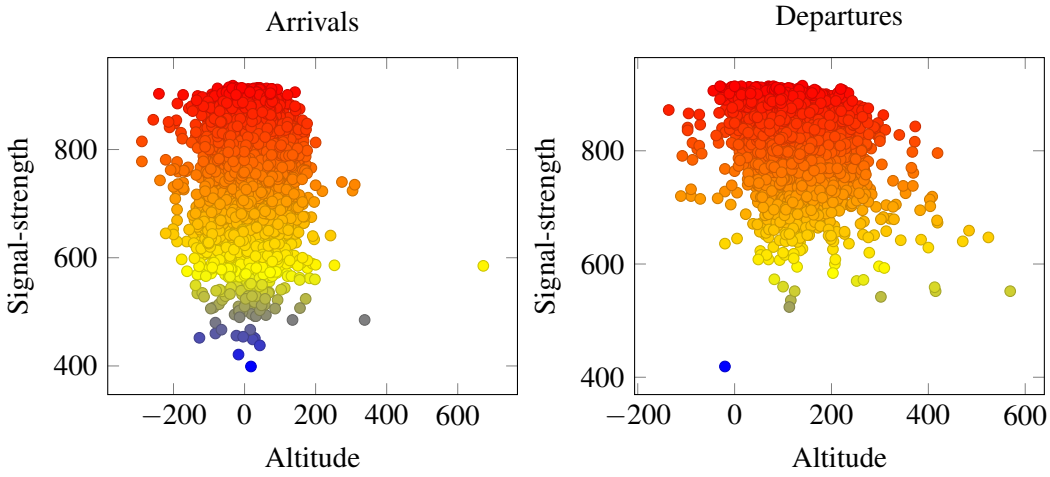


Figure 3.3: Altitude vs signal strength for arrivals and departures

the signal. To deal with this, the original algorithm ignored detections for a short timespan after its first detection. We also introduce this short mute time in our model. We set this timespan to correspond to the window size of the points we use, such that every point still is contained in a set that is classified. Since we are using a real world dataset, which was labelled by field experts, the chances of noise are significant. Before we continue with the feature selection process, we plot the signal-strength against the altitude to find outliers. These plots can be seen in figure 3.3 for both arrivals and departures. The arrivals are nicely concentrated around altitude 0. We see however that there is an outlier at altitude 672, and a few points with a low signal-strength in comparison to the majority. We discussed these cases with the field experts. The outlier at altitude 672 for arrivals was a labelling error. Of the 23 low signal-strength cases, 17 were wrongly labelled, which were also removed. This was decided upon by the field experts after looking into other data of the same aircraft which revealed they were performing acrobatics in those cases. 5 cases were actual events.

The departure plot shows a bit more spreading. The low altitude and signal-strength point, was a labelling error. The low signal-strength, high altitude points, which are green in the plot are caused by 1 aircraft which has a so called sticky transponder. It sticks to the low altitude for a longer time, and comes loose after reaching a certain altitude. The small scatterings with low altitude and high signal-strength contained 4 falsely labelled departures, and the ones with high altitude contained 3 falsely labelled departures. The rest was found valid.

### 3.3.1 Methods for selecting features

While it is useful to collect any kind of feature one finds available for a model, using all of them directly often results in the curse of dimensionality [15]. This means you have too many dimensions (features) in comparison to the amount of data available. The effect of this is that a model with all features performs worse than a model that includes a subset of features. The most common methods for finding which features to use, according to various papers[16, 17, 21] are filtering and wrapping. In filtering, each individual feature is given a score based on test methods such as the Welch's test, chi-squared test, information gain, and correlation coefficient [17]. This method is usually applied when there are too many features available to apply wrapping. Wrapping, in contrast to filtering, takes the effect of selecting subsets of features on the models performance into account. The wrapping algorithm can be described by the following 3 steps:

1. Select a subset of features
2. Train and evaluate a model
3. Use the evaluation metric for each result as feedback to select the next set of features

For the selection of subsets, various heuristics exist. The most common ones are forward and backward search, which are hill climbing techniques for optimization [22]. In case of using forward search, each individual feature would be used for training and validation, to then select the best first feature based on these scores. The next feature would be determined in the same way, up until the point where the difference in score when adding a feature is neglect-able. We provide a pseudo-code implementation here below to clarify.

```
var selectedFeatures = {}
var maxScore = 0
val minDelta = 0.001
var featureSet = allFeatures

while (featureSet.size > 0)
{
  var featureMetrics = {}
  for(var feature in featureset) {
    var model = trainModel(selectedFeatures +: feature)
    var score = model.evaluate()
    featureMetrics.Add((feature,score)})
  }
  selectedRecord = featureMetrics.max(record => record.score)
  if (maxScore + minDelta < selectedRecord.score) {
    maxScore = selectedRecord.score
    selectedFeatures.add(selectedRecord.feature)
    featureSet.remove(selectedRecord.feature)
  }
  else {
    break
  }
}
```

Listing 3.1: Forward search feature selection using wrapper method

Backwards search works similarly, but instead of adding features to the selection, it works by leaving one feature out each run, up until the point where leaving a feature out would

significantly impact the performance score of the resulting model. For clarification purposes we provide the pseudo-code for backward search here below.

```
var maxScore = 0
var featureSet = allFeatures

while (featureSet.size > 0)
{
  var featureMetrics = {}
  for(var feature in featureset) {
    var model = trainModel(selectedFeatures -: feature)
    var score = model.evaluate()
    featureMetrics.Add((feature,score))}
  }
  var potentialRemovableFeature = featureMetrics.min(record => record.score)
  if (maxScore <= potentialRemovableFeature.score) {
    maxScore = potentialRemovableFeature.score
    featureSet.remove(potentialRemovableFeature.feature)
  }
  else {
    break
  }
}
```

Listing 3.2: Backward search feature selection using wrapper method

In the backward search, if the score stays equal, the feature is also removed, as it apparently was not of influence to the performance, or in other words, is noise for the model. Note that both forward and backward search are greedy optimization algorithms. Selecting the very best feature set can be done by computing all possible solutions, but this is infeasible with the computational power that is currently available. In forward search, the algorithm does not have all features necessary for good predictions available from the start, which intuitively makes it harder to perform a good feature selection. Due to this, backward search wrapping is more commonly used. We also decided to go with backward search for feature selection. The score calculation is done by using the F measure with F = 1 as described by Garcia [12]. The calculation for the F measure is denoted in equation 3.2. The F measure allows for giving a relative importance of precision and recall by parameter β, but since precision and recall are equally important in our case, we leave this value on 1.

$$
\begin{aligned}
F &= \frac{(1+\beta)^2 \times \text{Recall} \times \text{Precision}}{\beta^2 \times \text{Recall} + \text{Precision}} \\
\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \quad &\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}
\end{aligned}
\tag{3.2}
$$

### 3.3.2 Selecting the features

In figure 3.2 we noted that the plots where for a window size of 20 points. This size was picked based on the original rule based system. However, this number was pseudo-randomly picked in the original algorithm. Due to this, we perform the wrapper method for window sizes 3 up to 20, and construct the feature sets accordingly to find an optimal window size and feature set given the computational resources available. The results of this feature selection process are shown in figure 3.4. What we see with arrivals is a curve that reaches

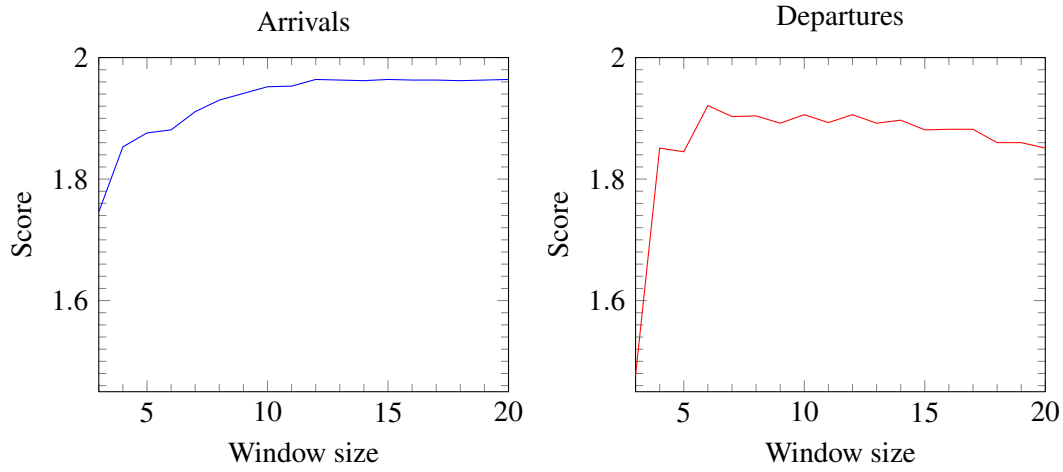


Figure 3.4: Window size scores for arrivals and departures

| **Model** | **Window** | **Selected Features** |
|---|---|---|
| Arrival | 12 | Max($A$), $A(x_1)$, $A(x_2)$, $A(x_3)$, $A(x_4)$, $A(x_7)$, $A(x_8)$, $A(x_9)$, $A(x_{10})$, $A(x_{12})$<br><br>Max(S), Min(S), P2P(S), $S(x_1)$, $S(x_7)$, $S(x_8)$, $S(x_10)$, $S(x_{11})$, $S(x_{12})$ |
| Departure | 6 | Max(A), Mean(A), Min(A), P2P(A), $A(x_1)$,$A(x_2)$, $A(x_3)$, $A(x_5)$, $A(x_6)$<br><br>Max(S), Mean(S), Min(S), Mode(S) , P2P(S), $S(x_1)$, $S(x_3)$, $S(x_4)$, $S(x_5)$, $S(x_6)$ |

Table 3.3: Results of feature selection process

it's highest point at window size 12, and then stays steadily around this point. This is explainable by the fact that larger window sizes have all features available that are also there for smaller window sizes. The slight fluctuation is due to the fact that computed features such as average and mean are being calculated over more points, causing their information to be less precise. Given that less is more in machine learning, we decide to go with the lowest window size with the highest score for arrivals, which is 12. The features selected for this window size are listed in table 3.3.

In figure 3.4 we see that for departures, the peak is at window size 6, and then has a fluctuating but declining trend towards 20. This is due to the fact that some people tend to turn on their transponders relatively late, causing the amount of available data points for

training, as well as the amount of test points to be detected to decline as the window size increases. However, since the peak is already at window size 6, this is not a big issue, as in contrast, the original algorithm requires 20 points of data, which causes a larger blind spot. Given this, we take window size 6 for departures and listed the selected features in table 3.3. As we expected, the resulting window size differs significantly for arrivals and departures. In fact, the window size of departures is half that of arrivals. In addition, we see that while the departure model is more based on computed features, the arrival model uses mostly raw values as features. This could be caused by the high window size of the arrivals, which makes the computed values less informative. However, since the overall score is high, further exploration of features is not necessary.

## 3.4 Validation and evaluation methods

The machine learning field uses various methods and metrics for validating a model. The most commonly used methods are a test/training split and cross validation. Usually, a train/test split is done to have an indication of how the model would perform in the future on data it has not seen before, while cross validation is used to evaluate how well the model generalizes over the data. Both methods work by training a model, and then testing the model by running predictions for the test data and computing a confusion matrix, that allows for computation of all kinds of other metrics such as precision, recall, accuracy and discovery rates. Since we are already generalizing over aircrafts, we go with a training and testing split for our validation method, as also described in section 3.2. Our data is split into two parts with each a time frame of 2 months. This ensures that we have various aircrafts and weather conditions in both of our datasets. To verify whether our models learned properly, we look at each individual FP and FN and discuss how they can be addressed in chapter 4.

Both the train/test and cross validation method are aimed at measuring exact performance and do not take into account the time aspect, while this is of importance with time series data. To illustrate, suppose a series of predictions of $\{0,0,0,1,0,0,0\}$ where the interval between each point is 1 second. If we now take the truth dataset which is represented by $\{0,0,1,0,0,0,0\}$, and strictly match these two, like what would be done with non-time series classification, the model predicted 5 TN, 1 FP, and 1 FN. This gives a very negative result in case a time tardiness is allowed for detection. In our case, we would like this tardiness to be reduced to a minimum, but want to know how well the model performs if we allow this slight offset in time. To solve for this we validate the model by first predicting for the complete test set, and then doing a mapping of the prediction results on the test results. This mapping is done by using the Gale-Shapley stable marriage algorithm. This algorithm is designed to find a stable mapping between two sets of data points based on certain constraints. In our case we have ground truth data points $G = \{g_1, ..., g_n\}$ and the predictions made based on our model $P = \{p_1, ..., p_m\}$. Each point in both $G$ and $P$ has an aircraft id, event type and time-stamp which we can use to define our constraints for the mapping process. We start off by applying matching constraints, such that only points of the same

event type and belonging to the same aircraft can be matched. Additionally, we introduce a constraint such that the time difference between two points may be at most $\tau$ seconds. This causes an extra variable to be evaluated. The area under the curve (AUC) of the Receiver Operator Characteristic (ROC)[23] was introduced to deal with a similar problem, namely that of the decision boundary threshold. This threshold decides whether a point belongs to the positive or negative class, and is usually set to 0.5. The AUCROC curve made it possible to evaluate the algorithm while moving this decision boundary from 0 to 1, and is defined by $ROC = \int_0^1 \frac{TP}{TP+FP} d(\frac{FP}{TP+FP})$. It evaluates the true positive rate against the false positive rate.

Davis et al. [24] have shown that these measures cause an overly optimistic view of a model when the data is imbalanced. In addition, Garcia argued the same in [12]. As our data is significantly imbalanced, the ROC curve would not work for us. Davis et al. presented the precision-recall curve (PRC) [24] to deal with this imbalance. It works similar to ROC, but instead of using the false positive rate, the precision rate is used. This results in the following definition: $PRC = \int_0^1 \frac{TP}{TP+FP} d(\frac{TP}{TP+FP})$. However, both of these metrics are aimed at finding the optimal decision boundary, while we are not trying to find the optimum $\tau$ value, but rather try to evaluate our model over different $\tau$ values. Due to this we cannot directly use either of these metrics. However, we can plot precision and recall for different $\tau$ values just like is done for the decision boundaries in prior metrics. This allows us to compare the precision and recall individually for the machine learning model against the original algorithm. We let $\tau$ range from 0 to 60 seconds with step sizes of 1 second. This makes for a computation of 61 confusion matrices per model, allowing us to plot individual curves for precision and recall against the $\tau$ values. These plots then represent how well the model performs when a tardiness in time of $\tau$ is allowed. For example, if the precision is 60% at $\tau = 10$, then 60% of the detected events were actual events that happened within 10 seconds of detection, while 40% of the detected events did not happen within 10 seconds of detection. In addition, if the recall is 70% at $\tau = 10$, then 70% of the actual events are detected when a tardiness of 10 seconds is allowed.

Given a metrics curve plotted over $\tau$ we can then state that the model/algorithm with the largest area under the curve has the best performance. This is due to the fact that we plot $\tau$ from 0 to 60, and want to have recall and precision as high as possible as early as possible. This gives us a fair way to compare both the algorithm and machine learning model's performance.

## 3.5 Algorithm selection

An important part of model creation is algorithm selection. Unfortunately, there is no predefined rule that states which algorithm best fits our problem. With that, the fact that there are numerous algorithms already out there does not make this process easier. Some of the mainstream algorithms are K-NN, Naive Bayes, Support Vector Machines, Random Forest, various Artificial Neural Networks and several deep learning techniques. Especially deep

learning has become interesting ever since a breakthrough in training deep belief networks by Hinton et al. in 2015[25].

We looked at deep learning and quickly found that the arrival and departure class data is too small to allow for performing deep learning. In addition, we found that Naive Bayes does not take into account a pattern within a group of features. It works by evaluating each feature individually, and finally multiplying them together to determine which class the point belongs to. While this makes Naive Bayes perform well in text classification, it won't work well in our case. This leaves Random forest, SVM's, a shallow neural network and K-NN open.

Before we went further into exploring each algorithm we looked at similar research in the field of time series classification. Keogh et al. explored K-NN (with K=1) versus a multi layer perceptron neural network and various other algorithms in 2006 [26]. They reported that K-NN with K=1 is hard to beat. In 2009, independent research on classifying activity is done by Brezmes et al. This research also uses K-NN for to classify their raw data as activities. In 2011, Keogh et al. show that his claim on K-NN still holds [27]. In 2013 Hu et al. publish a paper titled 'Time series classification under more realistic assumptions' [28]. In this work, K-NN was also used for the time classification part. While these publications are all done before the finding of Hinton et al. in 2015, we already explained that deep learning can unfortunately not be applied in our case, thus these publications still apply on our case.

Due to these publications we looked further into the benefits of using K-NN and found that it does not require long training times, is easy to understand and can be corrected locally on false predictions by simply adding corrections into the existing tree. Shallow neural networks on the other hand need to have a network architecture defined and are harder to understand and fine tune with limited data. In addition, SVMs need kernel selection and parameter tuning while this is not the case for K-NN. This makes K-NN a desirable algorithm, as it is maintainable and predictable. For K we pick K=1 as we only have a limited set of arrivals and departures available in comparison to the majority neutral class. Picking K=2 could introduce ties, which is why picking K=2 is generally not done when using the K-NN algorithm. Picking K=3 or higher introduces the risk of having too few arrivals and departures. In addition, prior work showed that K=1 worked best in most cases.

# Chapter 4

# Results

Let us recap research question 3: '*How does the machine learning model perform in comparison to the fixed rule system*'. In this chapter we answer this question by first presenting the evaluation results of the original fixed rule system and machine learning version, followed by a comparison and discussion on these results. The evaluation method we performed to compute these results can be found in chapter 3.

## 4.1 Original Algorithm

The original algorithm flags a point as arrival, departure or neutral. Due to this, we have to measure the performance of the original algorithm separately for arrivals and departures. To run the algorithm, we also need to fill in all parameters. For this we use the configuration as is used at the Dutch airport where the algorithm is active.

### 4.1.1 Arrivals

By testing the algorithm's performance against the arrivals test set with $\tau$ ranging from 0 up to 60, we created a precision and recall curve as seen in figure 4.1. Let us recall that $\tau$ stands for the time in seconds the algorithm is allowed to be off the actual moment of an event.

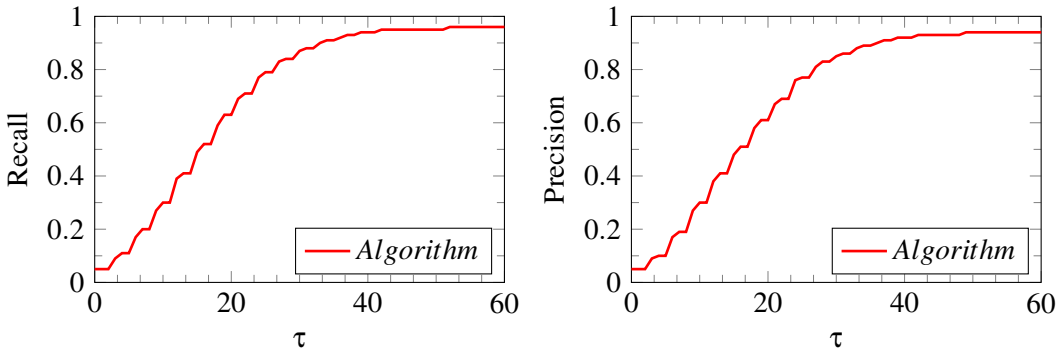


Figure 4.1: Precision and recall for arrivals of original algorithm plotted against $\tau$

The raw values used to create this plot are also included in appendix B. Both plots have $\tau$ on the $x$ axis, and respectively recall and precision on the $y$ axis. The first thing to notice is that there are steps between the t values in both plots. These are explained by the interval of approximately 3 seconds between each data point coming in from the transponder. These steps are partially caused by the transponders, and partly by the mechanism on how the transponder data is read from the system. However, since the same data is used among all experiments this does not introduce an advantage for either of the options.

The second thing to notice is that the curves for both recall and precision follow the same trend. This can be explained by the fact that precision and recall only differ slightly in their computation. However, this small difference greatly influences the performance measured. With precision the percentage of detected events that were actual events is measured, while with recall the percentage of actual events that are also detected as events is measured. As we can see from the plots, only 5% of the arrivals are detected exactly at the moment that they occur. The original algorithm approaches precision and recall of 90% when allowing for a time difference of up to 36 seconds, and reaches a 94% precision and 96% recall at 52 seconds, after which it no longer increases up to our measured $\tau$ of 60 seconds.

With these results we can safely say that the original algorithm manages to get a high precision and recall, but the offset in time to reach these values is high as well. This is strongly undesirable, as in practice one needs to verify whether the data is correct. If the system detects an arrival, while the aircraft is still 36 seconds away from landing, verifying whether the system detected the aircraft is hard, especially when aircrafts are coming and going constantly.

### 4.1.2 Departures

Just like we did with the arrivals, we tested the algorithms performance against the departure test set with $\tau$ ranging from 0 up to 60. The resulting precision and recall curves can be seen in figure 4.2, and the measurements table is available in appendix B.

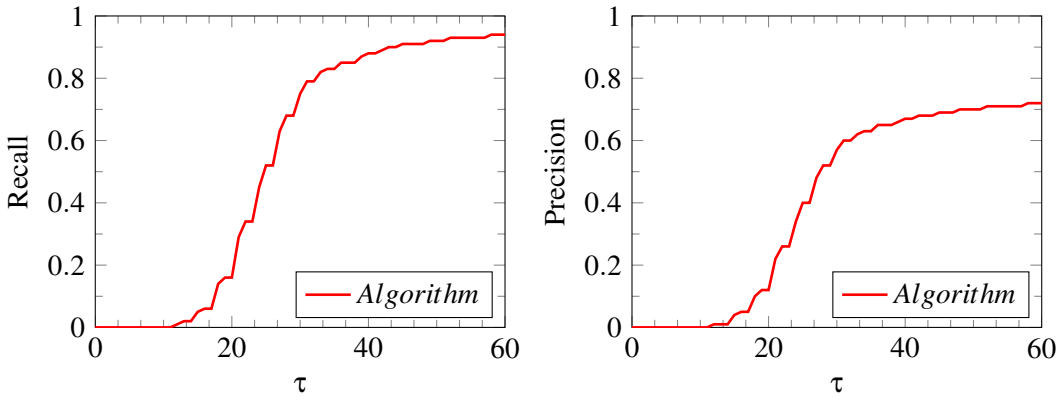


Figure 4.2: Precision and recall for departures of original algorithm plotted against $\tau$

Just like with the arrival plots, the curves contain steps, which is again explained by the interval at which the data is received by the system. In contrast to arrivals, the curves for recall and precision differ significantly for departures. While recall goes towards 94%, precision only comes up to 72% for $\tau = 60$. If we look once more at the definition of precision and recall, we find that the algorithm detects quite some false departures. In chapter 2 we explained how departures are detected based on going through an altitude level.

These results give an indication that for departures, the detection altitude might be too low for specific weather conditions, causing false positives to occur. However, if we look at the recall and precision rates with respect to $\tau$, we see that with the current altitude level it already takes 12 seconds before a single departure is detected, and up to 43 seconds to get a recall of 90%. Moving the altitude up more would increase this delay even further. This shows the fundamental issue with this algorithm for accurate and fast departure detection.

## 4.2 Machine learning Models

Based on the techniques that we described in chapter 3, we created two separate models, one for arrivals and one for departures. These models are not based on a fixed set of rules and parameters like the original algorithm, but instead are trained based on a dataset. Due to this, no assumptions are made on the importance of, for example, altitude versus signal strength. Instead, the importance of these values is determined automatically via wrapping. In addition, the time delay caused by having to go through a fixed altitude level, as is the case with the algorithm, should be removed, as training data is labelled on the exact point in time, rather than $x$ seconds earlier or later. In the rest of this section we present the results of the arrivals and departures model.

### 4.2.1 Arrivals

Based on our feature selection result in section 3.3.2, we created an arrival model using window size 12 and 19 distinct features. This model was then evaluated using the test set with $\tau$ ranging from 0 to 60. We then computed the precision and recall values for each of the 61 measurements, and plotted a precision and the recall curves which can be seen in figure 4.3.

From this plot we can see that 26% of the arrivals are detected exactly on point. From there we see that for every new batch of data, the precision and recall go up by approximately 16%. At $\tau = 16$ recall reaches 89% versus a precision of 90%. From there on, the recall and precision slowly climb towards 99% at $\tau = 40$. From there, no new arrivals are detected. This is also unlikely as these results are close to perfect.

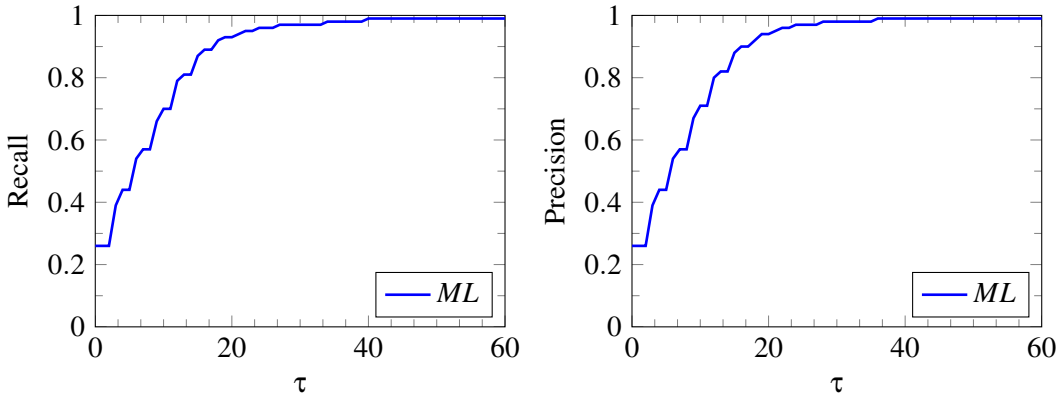


Figure 4.3: Precision and recall for arrivals of model plotted against $\tau$

Arrivals detected in the range of $16 < \tau < 40$ are arrivals where the aircraft's transponder had a large altitude offset. In other words, the transponder reported altitudes as low as -400 feet, while the arrival was detected around 50 feet. While this is a bit unfortunate, it causes no harm, and is more of an issue of the transponders than it is of our model. The last missing percent of precision and recall is caused by a total of 7 false positives, and 19 false negatives. The false positives are caused by 2 transponders which strangely enough had a bouncy signal, but only for a short time. For example, in one case the transponder reported an altitude of -2, followed by an altitude of 2321 and went down to -2 again. This is natural to real world data, where things are not perfect, so we accept these cases as they are. One could address this by a form of preprocessing on the sensor data, to remove such infeasible cases. In addition, this makes for a great new study on detecting faulty transponders.

### 4.2.2 Departures

Given the window size of 6, and 19 distinct features for the departure detection found in section 3.3.2, we created a model and ran our validation method for $\tau$ values 0 up to 60. The resulting precision and recall curves can be seen in figure 4.4. The first thing to notice is that with $\tau = 0$ a precision and recall of approximately 18% is reached. Given the next batch of data coming available at $\tau = 3$ this shoots up to around 49%. With allowing a time error of only 12 seconds, 90% precision and 94% recall is reached. The cause for this time difference comes from generalisation over aircrafts and runway usage. The runway can be used from two different sides, and each side gives a slightly different signal. In addition, there are various transponder types and brands, and we are now generalizing over all of them by creating one model for all aircrafts. If we were to build a model per aircraft we could reduce this delay even further. However, as we explained in chapter 3, this would introduce the limitation of not detecting aircrafts when they have not been seen before. From 12 seconds onward, the precision and recall climb slowly towards 94% and 98% respectively. The error in precision comes down to a total of 51 false positives. For recall a set of 20 false negatives is the cause of the missing 2 percent.

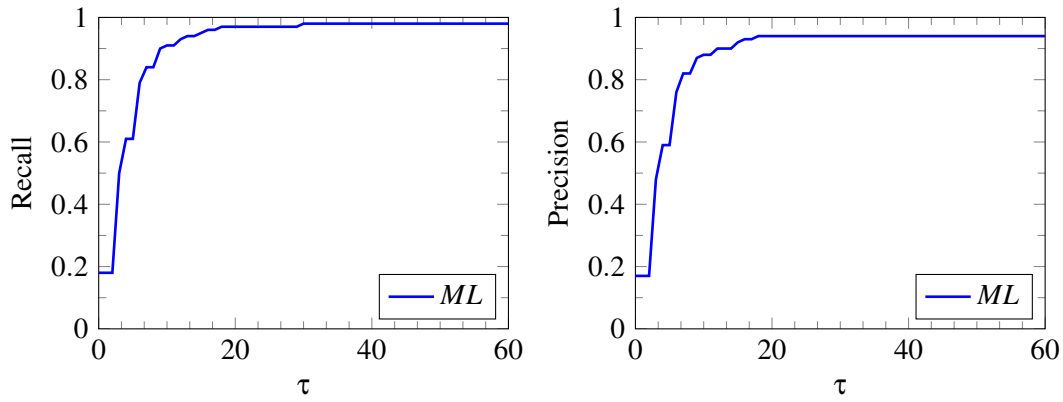


Figure 4.4: Precision and recall for departures of model plotted against $\tau$

Of these 51 false positives, 42 are caused by 3 aircrafts, of which the transponder gives a bouncy signal. The other 9 positives look like departures in their signal but don't take off in the end. These are most likely cancelled departures, as for most of these cases, a little while later, the aircraft did take off. Of the 21 false negatives, 16 were caused by a lack of data. There simply were no 6 data-points available for feature creation for these 16 cases. Most of them occurred in the same time window, indicating a possible issue with our data collector at that time. The other 5 were very particular signals that almost looked like overshoots on the runway. An overshoot happens when the aircraft attempts to land, but pulls up again because it can't make the landing safely. We did not train our model in particular on these cases, and as they are very rare, the chance of them being in our training data is slim. We asked a field expert regarding these specific cases and they noted that these 5 cases would have been removed from the system, so would not have been false negatives in practice.

The overall view on false positives and negatives do not indicate an issue with our model, but rather with outside factors such as a broken transponder and a drop of communication in the receiver, and unfortunate events on the runway. The overall precision and recall of the departure model climb vastly with low $\tau$ values, which indicates that this model detects departures fast and accurately.

## 4.3 Results comparison

Given the performance measures of both the algorithm and machine learning models for each individual event type, we can compare them such that we can answer our final sub question. Since arrivals and departures are separate models, we go into detail on each model individually.

### 4.3.1 Arrivals

For comparison reasons, we plotted the curves of both the original algorithm and the arrival model in figure 4.5. We can see here that at the initial $\tau$ value of 0, our model outperforms the algorithm by 21%. From there, our model climbs steeply towards 100% precision and recall, but a similar curve is happening for the original algorithm. The fact that these trends are similar is explainable by the fact that the original algorithm detects arrivals at a certain altitude threshold, which lies around 350 feet to prevent issues with detection. In contrast, our model can detect arrivals much more on point, while not having other issues with detection. This comes from the fact that our model exploits multiple values of the signal, and does not solely rely on altitude but also takes signal strength as a strong indicator. Furthermore we can see that our model reaches a higher precision and recall. The missed arrivals by the original algorithm are caused by not having a high enough signal strength when they were within the altitude range. When the signal strength was high enough to be accepted as arrival, the aircrafts were already below the altitude level, which caused them to be missed out on.

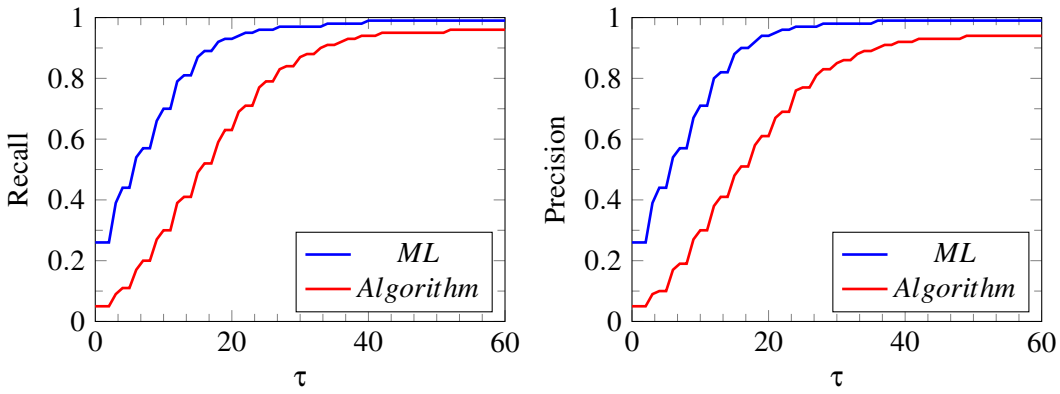


Figure 4.5: Precision and recall for arrivals of model and algorithm plotted against $\tau$

### 4.3.2 Departures

In figure 4.6 we plotted the precision and recall curves for both the original algorithm and machine learning model for departures. From these plots it is clear that the overall performance of our newly created machine learning model exceeds that of the original algorithm. In particular, the precision significantly exceeds that of the original algorithm, even with an allowed delay in detection of up to 60 seconds. The recall is fairly close at $\tau = 60$. However, we can see that while the original algorithm starts detecting departures only after 12 seconds, our machine learning model already detected up to 90% of the departures at that time. This shows how our machine learning model detects departures significantly faster in comparison to the original algorithm. Since the goal of arrival and departure detection is to be near real time, this difference is a significant improvement over the original algorithm.

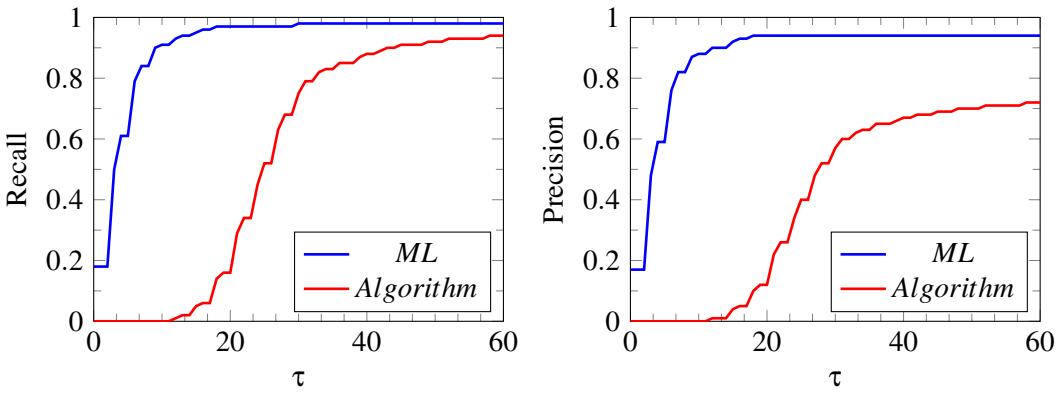


Figure 4.6: Precision and recall for departures of model and algorithm plotted against $\tau$

# Chapter 5

# Conclusions and Future Work

## 5.1 Conclusion

We conclude our work by going over our sub questions, and finally answering our main research question:

1. *How can machine learning be applied for detection of arrivals and departures of aircrafts using ADS-B data?*
   Detecting arrivals and departures is a form of time series classification that can be performed using techniques as described in chapter 3. These techniques are not unique to arrival and departure detection, but are generally used for time series classification. However, we noticed that there is a significant difference between detecting from a repeating pattern versus detecting from a single pattern. In the latter case, allowing a small offset in time can greatly increase performance indicators, as matching on point can give very negative results, while it would be detected a matter of seconds later.

2. *How can the performance of the fixed rule based algorithm be measured and compared to a machine learning model?*
   By defining a test set and a measurement method using the Gale Shapley algorithm, we were able to do a thorough measurement of the performance of both the original algorithm and machine learning model with respect to tardiness. Given that we can use this measurement method on both the algorithm and models, and by using the same test set we can make a fair comparison between these two ways of detecting arrivals and departures. The metrics we used for this were precision and recall, as they give a good representation of the model's performance, even when dealing with imbalanced data.

3. *How does the machine learning model perform in comparison to the fixed rule system*
   Based on our results we can safely say that our machine learning model for both arrivals and departures significantly outperform the original algorithm when it comes to detection time and precision. It also outperforms on recall, however the difference here is lower when a large tardiness is allowed.

Answering all sub questions brings us to answer the our main research question: '*Can machine learning be used for detecting aircraft arrivals and departures based on ADS-B data and how does it compare to an existing fixed rule based system*'. We found that applying machine learning on ADS-B data can indeed allow for arrival and departure detection. In addition we found that our machine learning models outperform the original algorithm for both arrivals and departures, although the improvement is more significant for departures.

## 5.2 Contributions

In this work we described a complete case on how to come from raw data to a machine learning model. In addition we have shown that accurate and fast automatic arrival and departure detection at an airport is possible by applying machine learning on ADS-B receiver data. ADS-B receivers are inexpensive and do not require new hardware on board of the aircrafts, making it a feasible and cost efficient solution in comparison to acquiring people to do this manual work as a dedicated task. This allows more accurate information to be available at airports while reducing the workload. More accurate and fast information allows for airports to do a more informed planning of their available resources, which on its own results in an increase of efficiency and reduction of congestion. Coming to this final result of reduction of congestion still requires some work, which we will elaborate on in section 5.4. In addition, we presented a new way of measuring performance of a time series classification model where some tardiness is allowed. By running all predictions and allowing an offset in $\tau$ we were able to get a performance indication of our model with respect to tardiness. Finally we showed independently of Keogh et al. that K-NN with K=1 performs well in time series classification in both our arrival and departure model case.

## 5.3 Discussion

In this work we used a manually labeled dataset with a timespan of 4 months. To reduce issues of false labeling and bias, we did a 2 fold cross validation step, and looked at the false positives and false negatives for both cases. Using this method we corrected falsely labeled data and added missing data. However, by doing this, there is no guarantee that the data is completely clean of errors. Unfortunately this is a limitation of our data collection process. In addition we used data from one airport, while generalizing over multiple airports would be the ideal case. Unfortunately we did not data sources available from multiple airports to perform a multi airport test, as the system only became operational at the second airport in the last 2 months of this research. Future work could however cover this aspect.

We used wrapping for feature selection, which is an approximation algorithm. While this means that we possibly did not find the optimal feature set and window size of all possible combinations, finding this would take infinite time using the current computational power available. Since wrapping is a commonly used technique in machine learning, we believe this is an acceptable method to determine our features for a good performing model. In addition, by allowing tardiness, we introduced the risk of over-fitting on falsely linked

data. However, as the precision and recall curves displayed the performance significantly increases in the first few time steps, and is barely affected afterwards, and given that we performed cross validation we believe this risk is minor. Especially as we also looked into the false positives and false negatives to verify whether the models errors were explainable, which was the case for both arrivals and departures.

## 5.4 Future work

While we generalized over aircrafts and weather conditions by using a dataset that represents a significant portion of this data, the current model is not yet tested at multiple airports. Future work could go into how far the current models can be generalized such that they can be picked up and put into operation at different airports. One way could be to find the minimum amount of data needed to train a new model. Another idea could be to collect and label data similarly to what we did, but for another airport and compare the altitude and signal strength datasets. Alternatively, semi supervised or even unsupervised learning techniques could be explored in order to create a model. The latter can however be a difficult process due to the low representation of the arrival and departure class in contrast to the high amount of neutral data points. We did an initial test using K-Means to see if we could do without manual labelling but this came out negative. However, before we tried this, we did not know which features to use, so all of them were included, which most likely affected the performance of the K-Means algorithm.

In chapter 4 we briefly mentioned overshoots and cancelled departure events to be among our false positives and false negatives. Our research did not go into detail on these particular cases, as we were not sure if machine learning could generalize over the different aircrafts and their respective transponders. However, in this work we showed that this can be done, thus future research could go into detecting these cases, as these are also reported in the CDM system as designed by Euro control.

While we focused on measuring arrival and departure times using a single ADS-B antenna, we now believe that using multiple antennas, and combining those data sources, more CDM milestones, such as on-block and off-block could be detected. Bringing in multiple antenna's does bring new challenges that have to be overcome, such as positioning, but these are challenges that have been addressed before by for example indoor localization using WIFI signals.

We know from the prior system that one could escape automatic detection by flying low while moving away from the airport, to then climb when the signal strength was significantly reduced. This is obvious from the design of the algorithms and corresponding parameters. However, for the machine learning models, we cannot directly tell how one can fool the system. Future research could go into this attempt of fooling the system, and could look into adversarial online learning [7] as possible solution to this problem.

# Bibliography


[1] IATA. Iata forecasts passenger demand to double over 20 years. http://www.iata.org/pressroom/pr/Pages/2016-10-18-02.aspx, 2016.

[2] FlightAware. Do you want to build your own flightaware piaware ads-b ground station? https://flightaware.com/adsb/piaware/build, 2017.

[3] Air Traffic Management. Nz satellite-based aircraft tracking on the way. http://www.airtrafficmanagement.net/2016/09/nz-satellitebased-aircraft-tracking-on-the-way, 2016.

[4] Eurocontrol. Airport collaborative decision making. http://www.eurocontrol.int/articles/airport-collaborative-decision-making-cdm, 2016.

[5] Eurocontrol. Airport collaborative decision making presentation 2011. https://www.eurocontrol.int/sites/default/files/event/files/airport_cdm_presentation_madrid-30-03-2011.pdf, 2011.

[6] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[7] Myriam Abramson. Toward adversarial online learning and the science of deceptive machines. In *2015 AAAI Fall Symposium Series*, 2015.

[8] Stefan Ravizza, Jun Chen, Jason AD Atkin, Paul Stewart, and Edmund K Burke. Aircraft taxi time prediction: comparisons and insights. *Applied Soft Computing*, 14:397–406, 2014.

[9] Xuhui Wang and Ping Shu. Incremental support vector machine learning method for aircraft event recognition. In *Enterprise Systems Conference (ES), 2014*, pages 201–204. IEEE, 2014.

[10] Tomas Brezmes, Juan-Luis Gorricho, and Josep Cotrina. Activity recognition from accelerometer data on a mobile phone. In *International Work-Conference on Artificial Neural Networks*, pages 796–799. Springer, 2009.

[11] Ali Bahrami Rad, Trygve Eftestol, Jan Terje Kvaloy, Unai Ayala, Jo Kramer-Johansen, and Kjersti Engan. Nearest-manifold classification approach for cardiac arrest rhythm interpretation during resuscitation. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3621–3625. IEEE, 2014.

[12] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[13] Rohit Chatterjee Microland. Using modes for time series classification. `http://www.slideshare.net/bangaloremicroland/using-modes-for-time-series-classification-rohit-chatterjee`, 2015.

[14] Pierre Geurts. Pattern extraction for time series classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 115–127. Springer, 2001.

[15] Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In *International Work-Conference on Artificial Neural Networks*, pages 758–770. Springer, 2005.

[16] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[17] Huan Liu and Hiroshi Motoda. *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media, 2012.

[18] Garrecht Avionik GmbH. Trx-1090 ads-b traffic receiver. `http://www.air-avionics.com/support/TRX_1090_USR_e_rev1.0f.pdf`.

[19] Giovanna Menardi and Nicola Torelli. Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery*, pages 1–31, 2014.

[20] Anthony Bagnall and Jason Lines. An experimental evaluation of nearest neighbour time series classification. *arXiv preprint arXiv:1406.4757*, 2014.

[21] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[22] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[23] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145 – 1159, 1997.

[24] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[26] Li Wei and Eamonn Keogh. Semi-supervised time series classification. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 748–753. ACM, 2006.

[27] Gustavo EAPA Batista, Xiaoyue Wang, and Eamonn J Keogh. A complexity-invariant distance measure for time series. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 699–710. SIAM, 2011.

[28] Bing Hu, Yanping Chen, and Eamonn Keogh. Time series classification under more realistic assumptions. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 578–586. SIAM, 2013.

# Appendix A

# Glossary

In this appendix we give an overview of frequently used terms and abbreviations.

**Turn-around:** The complete process of arriving at an airport, unloading, processing services, boarding and departing. More information can be found in section 1.2.1.

**Automatic Dependent Surveillance-Broadcast (ADS-B):** A surveillance system in which aircrafts periodically broadcast information such as altitude, transponder code. More information can be found in section 1.2.2.

**Flight level xxx:** An altitude at a standard pressure of 1013.2 hPa translated into a number (xxx) for ease of communication. For example Flight level 010 is at 1000 feet.

**Supervised learning:** Supervised learning is the subclass of machine learning where a training dataset including expected results is used to create a model. This is in contrast to unsupervised learning where the training data does not contain expected results.

**Classification:** Classification is the subclass of machine learning problems where a record is labelled as belonging to one of either classes. In contrast to for example regression where a real value is computed for a record.

**Filtering:** Filtering is a feature selection method that selects statistically independent features based on a set of tests, but does not take the effect this has on a machine learning model into account.

**Wrapping:** Wrapping is a method for feature selection that takes the bias of a trained model into account when selecting features. More information can be found in section 3.3.

**Precision:** Precision is a metric that represents the percentage of positively marked cases by a model, that were actually positive cases. The formula for precision can be found in section 3.3 in equation 3.2.

**Recall:** Recall is a metric that represents the percentage of actual positive cases that were marked as positives by a model. The formula for recall can be found in section 3.3 in equation 3.2.

**TP (True Positives):** Records marked as positive by a machine learning model, that were actually positive cases.

**FP (False Positives):** Records marked as positive by a machine learning model, that were actually negative cases.

**TN (True Negatives):** Records marked as negative by a machine learning model, that were actually negative cases.

**FN (False Negatives):** Records marked as negative by a machine learning model, that were actually positive cases. In binary classification this can also be seen as missed cases by the model.

# Appendix B

# Measurement results

| τ | Precision | Recall | τ | Precision | Recall | τ | Precision | Recall | τ | Precision | Recall |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.05 | 0.05 | 16 | 0.51 | 0.52 | 32 | 0.86 | 0.88 | 48 | 0.93 | 0.95 |
| 1 | 0.05 | 0.05 | 17 | 0.51 | 0.52 | 33 | 0.88 | 0.90 | 49 | 0.94 | 0.95 |
| 2 | 0.05 | 0.05 | 18 | 0.58 | 0.59 | 34 | 0.89 | 0.91 | 50 | 0.94 | 0.95 |
| 3 | 0.09 | 0.09 | 19 | 0.61 | 0.63 | 35 | 0.89 | 0.91 | 51 | 0.94 | 0.95 |
| 4 | 0.1 | 0.11 | 20 | 0.61 | 0.63 | 36 | 0.9 | 0.92 | 52 | 0.94 | 0.96 |
| 5 | 0.1 | 0.11 | 21 | 0.67 | 0.69 | 37 | 0.91 | 0.93 | 53 | 0.94 | 0.96 |
| 6 | 0.17 | 0.17 | 22 | 0.69 | 0.71 | 38 | 0.91 | 0.93 | 54 | 0.94 | 0.96 |
| 7 | 0.19 | 0.20 | 23 | 0.69 | 0.71 | 39 | 0.92 | 0.94 | 55 | 0.94 | 0.96 |
| 8 | 0.19 | 0.20 | 24 | 0.76 | 0.77 | 40 | 0.92 | 0.94 | 56 | 0.94 | 0.96 |
| 9 | 0.27 | 0.27 | 25 | 0.77 | 0.79 | 41 | 0.92 | 0.94 | 57 | 0.94 | 0.96 |
| 10 | 0.3 | 0.30 | 26 | 0.77 | 0.79 | 42 | 0.93 | 0.95 | 58 | 0.94 | 0.96 |
| 11 | 0.3 | 0.30 | 27 | 0.81 | 0.83 | 43 | 0.93 | 0.95 | 59 | 0.94 | 0.96 |
| 12 | 0.38 | 0.39 | 28 | 0.83 | 0.84 | 44 | 0.93 | 0.95 | 60 | 0.94 | 0.96 |
| 13 | 0.41 | 0.41 | 29 | 0.83 | 0.84 | 45 | 0.93 | 0.95 | | | |
| 14 | 0.41 | 0.41 | 30 | 0.85 | 0.87 | 46 | 0.93 | 0.95 | | | |
| 15 | 0.48 | 0.49 | 31 | 0.86 | 0.88 | 47 | 0.93 | 0.95 | | | |

Table B.1: Precision and recall of arrivals per τ value for original algorithm

| τ | Precision | Recall | τ | Precision | Recall | τ | Precision | Recall | τ | Precision | Recall |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 16 | 0.05 | 0.06 | 32 | 0.6 | 0.79 | 48 | 0.7 | 0.91 |
| 1 | 0 | 0 | 17 | 0.05 | 0.06 | 33 | 0.62 | 0.82 | 49 | 0.7 | 0.92 |
| 2 | 0 | 0 | 18 | 0.1 | 0.14 | 34 | 0.63 | 0.83 | 50 | 0.7 | 0.92 |
| 3 | 0 | 0 | 19 | 0.12 | 0.16 | 35 | 0.63 | 0.83 | 51 | 0.7 | 0.92 |
| 4 | 0 | 0 | 20 | 0.12 | 0.16 | 36 | 0.65 | 0.85 | 52 | 0.71 | 0.93 |
| 5 | 0 | 0 | 21 | 0.22 | 0.29 | 37 | 0.65 | 0.85 | 53 | 0.71 | 0.93 |
| 6 | 0 | 0 | 22 | 0.26 | 0.34 | 38 | 0.65 | 0.85 | 54 | 0.71 | 0.93 |
| 7 | 0 | 0 | 23 | 0.26 | 0.34 | 39 | 0.66 | 0.87 | 55 | 0.71 | 0.93 |
| 8 | 0 | 0 | 24 | 0.34 | 0.45 | 40 | 0.67 | 0.88 | 56 | 0.71 | 0.93 |
| 9 | 0 | 0 | 25 | 0.4 | 0.52 | 41 | 0.67 | 0.88 | 57 | 0.71 | 0.93 |
| 10 | 0 | 0 | 26 | 0.4 | 0.52 | 42 | 0.68 | 0.89 | 58 | 0.72 | 0.94 |
| 11 | 0 | 0 | 27 | 0.48 | 0.63 | 43 | 0.68 | 0.9 | 59 | 0.72 | 0.94 |
| 12 | 0.01 | 0.01 | 28 | 0.52 | 0.68 | 44 | 0.68 | 0.9 | 60 | 0.72 | 0.94 |
| 13 | 0.01 | 0.02 | 29 | 0.52 | 0.68 | 45 | 0.69 | 0.91 | | | |
| 14 | 0.01 | 0.02 | 30 | 0.57 | 0.75 | 46 | 0.69 | 0.91 | | | |
| 15 | 0.04 | 0.05 | 31 | 0.6 | 0.79 | 47 | 0.69 | 0.91 | | | |

Table B.2: Precision and recall of departures per τ value for original algorithm

| τ | Precision | Recall | τ | Precision | Recall | τ | Precision | Recall | τ | Precision | Recall |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.26 | 0.26 | 16 | 0.90 | 0.89 | 32 | 0.98 | 0.97 | 48 | 0.99 | 0.99 |
| 1 | 0.26 | 0.26 | 17 | 0.90 | 0.89 | 33 | 0.98 | 0.97 | 49 | 0.99 | 0.99 |
| 2 | 0.26 | 0.26 | 18 | 0.92 | 0.92 | 34 | 0.98 | 0.98 | 50 | 0.99 | 0.99 |
| 3 | 0.39 | 0.39 | 19 | 0.94 | 0.93 | 35 | 0.98 | 0.98 | 51 | 0.99 | 0.99 |
| 4 | 0.44 | 0.44 | 20 | 0.94 | 0.93 | 36 | 0.99 | 0.98 | 52 | 0.99 | 0.99 |
| 5 | 0.44 | 0.44 | 21 | 0.95 | 0.94 | 37 | 0.99 | 0.98 | 53 | 0.99 | 0.99 |
| 6 | 0.54 | 0.54 | 22 | 0.96 | 0.95 | 38 | 0.99 | 0.98 | 54 | 0.99 | 0.99 |
| 7 | 0.57 | 0.57 | 23 | 0.96 | 0.95 | 39 | 0.99 | 0.98 | 55 | 0.99 | 0.99 |
| 8 | 0.57 | 0.57 | 24 | 0.97 | 0.96 | 40 | 0.99 | 0.99 | 56 | 0.99 | 0.99 |
| 9 | 0.67 | 0.66 | 25 | 0.97 | 0.96 | 41 | 0.99 | 0.99 | 57 | 0.99 | 0.99 |
| 10 | 0.71 | 0.70 | 26 | 0.97 | 0.96 | 42 | 0.99 | 0.99 | 58 | 0.99 | 0.99 |
| 11 | 0.71 | 0.70 | 27 | 0.97 | 0.97 | 43 | 0.99 | 0.99 | 59 | 0.99 | 0.99 |
| 12 | 0.80 | 0.79 | 28 | 0.98 | 0.97 | 44 | 0.99 | 0.99 | 60 | 0.99 | 0.99 |
| 13 | 0.82 | 0.81 | 29 | 0.98 | 0.97 | 45 | 0.99 | 0.99 | | | |
| 14 | 0.82 | 0.81 | 30 | 0.98 | 0.97 | 46 | 0.99 | 0.99 | | | |
| 15 | 0.88 | 0.87 | 31 | 0.98 | 0.97 | 47 | 0.99 | 0.99 | | | |

Table B.3: Precision and recall of arrivals per τ value for the machine learning model

| τ | Precision | Recall | τ | Precision | Recall | τ | Precision | Recall | τ | Precision | Recall |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.17 | 0.18 | 16 | 0.93 | 0.96 | 32 | 0.94 | 0.98 | 48 | 0.94 | 0.98 |
| 1 | 0.17 | 0.18 | 17 | 0.93 | 0.96 | 33 | 0.94 | 0.98 | 49 | 0.94 | 0.98 |
| 2 | 0.17 | 0.18 | 18 | 0.94 | 0.97 | 34 | 0.94 | 0.98 | 50 | 0.94 | 0.98 |
| 3 | 0.48 | 0.50 | 19 | 0.94 | 0.97 | 35 | 0.94 | 0.98 | 51 | 0.94 | 0.98 |
| 4 | 0.59 | 0.61 | 20 | 0.94 | 0.97 | 36 | 0.94 | 0.98 | 52 | 0.94 | 0.98 |
| 5 | 0.59 | 0.61 | 21 | 0.94 | 0.97 | 37 | 0.94 | 0.98 | 53 | 0.94 | 0.98 |
| 6 | 0.76 | 0.79 | 22 | 0.94 | 0.97 | 38 | 0.94 | 0.98 | 54 | 0.94 | 0.98 |
| 7 | 0.82 | 0.84 | 23 | 0.94 | 0.97 | 39 | 0.94 | 0.98 | 55 | 0.94 | 0.98 |
| 8 | 0.82 | 0.84 | 24 | 0.94 | 0.97 | 40 | 0.94 | 0.98 | 56 | 0.94 | 0.98 |
| 9 | 0.87 | 0.90 | 25 | 0.94 | 0.97 | 41 | 0.94 | 0.98 | 57 | 0.94 | 0.98 |
| 10 | 0.88 | 0.91 | 26 | 0.94 | 0.97 | 42 | 0.94 | 0.98 | 58 | 0.94 | 0.98 |
| 11 | 0.88 | 0.91 | 27 | 0.94 | 0.97 | 43 | 0.94 | 0.98 | 59 | 0.94 | 0.98 |
| 12 | 0.90 | 0.93 | 28 | 0.94 | 0.97 | 44 | 0.94 | 0.98 | 60 | 0.94 | 0.98 |
| 13 | 0.90 | 0.94 | 29 | 0.94 | 0.97 | 45 | 0.94 | 0.98 | | | |
| 14 | 0.90 | 0.94 | 30 | 0.94 | 0.98 | 46 | 0.94 | 0.98 | | | |
| 15 | 0.92 | 0.95 | 31 | 0.94 | 0.98 | 47 | 0.94 | 0.98 | | | |

Table B.4: Precision and recall of departures per τ value for the machine learning model

# Appendix C

# Visualisation of performance difference

Since both the algorithm and machine learning model are abstract things, and detection of arrivals and departures is a practical problem, we've build a demonstration tool which visualises the ADS-B data feed, and event decisions made by both the algorithm and model. To see how far off the decisions are from the actual events, we used the test data set for this visualisation. This allows us to also mark the ground truth point. A complete screenshot of the tool can be seen in figure C.1.

The image shows a total of 5 important components: a time, aircraft legend two graphs, and feed controls. The legend, to the left in this image, shows which color belongs to which aircraft registration in both graphs. The first graph shows the altitude in feet for each aircraft on the y axis, and the time on the X axis. The second graph shows the signal-strength in dBA for each aircraft on the y axis, and the time on the X axis. Note that the X axis for both graphs goes from $T-15$ to $T$, where $T$ is the time represented on top of the screen, $T-1$ stands for the last data-point received before time $T$, and so forth. Note that the actual time for the data-points $T-1$ can vary between aircrafts, in case their transponders were not turned on at exactly the same time, or are not reporting at the same interval. Also note that the only difference between the two graphs is the Y axis values.

The graphs themselves show an aircraft icon facing downwards in case of a landing, and facing upwards in case of a departure. Text above the aircraft icon tells whether the algorithm (ALG), machine learning model (ML), or ground truth (GT) was presented. In case a combination of them is presented, the texts are combined, as can be seen on step $T-4$ where the Machine learning model was on point with the ground truth for Aircraft PH-JBG.

The data presented in the graphs can be controlled by the feed controls. These controls allow for playback, pausing and fast forwarding through the data. In addition, 6 pre-set times have been defined for both arrivals and departures. Finally a search box is added to allow for quick stepping to specific times in the data.

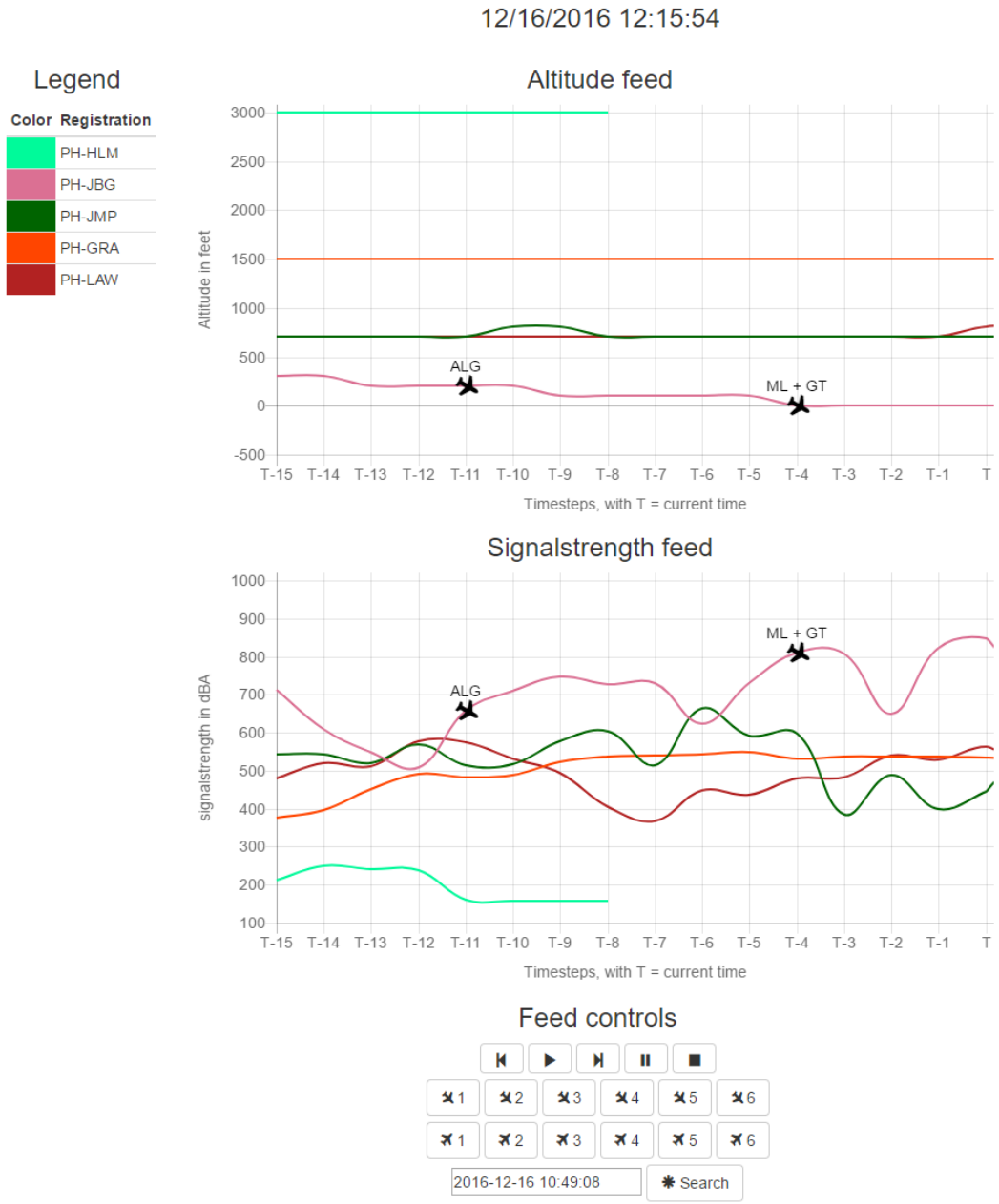


Figure C.1: Visualisation tool