Jarno Kiesiläinen

Predicting aircraft arrival times with machine learning

Master's thesis in Information Technology

May 11, 2020

University of Jyväskylä

Faculty of Information Technology

Author: Jarno Kiesiläinen

Contact information: jarno.e.kiesilainen@student.jyu.fi

Supervisors: Ilkka Pölönen, and Hannu-Heikki Puupponen

Title: Predicting aircraft arrival times with machine learning

Työn nimi: Ilma-alusten saapumisaikojen ennustaminen koneoppimismenetelmin

Project: Master's thesis

Study line: Computational sciences

Page count: 58+2

Abstract: This Master's Thesis studies the viability of using aircraft flight, flight plan and weather data with machine learning to predict aircraft travel time.

Keywords: machine learning, time prediction, air travel data, weather data, open data, boosting, neural network, random forest, support-vectors, regression

Suomenkielinen tiivistelmä: Tässä Pro Gradu -tutkielmassa tutkitaan lentokoneiden matka-ajan ennustamista lentodatan, lentosuunnitelmien, säädatan ja koneoppimisen avulla.

Avainsanat: koneoppiminen, ajan ennustus, ilmailu data, säädata, avoin data, boosting, neuroverkot, satunnaiset metsät, tukivektori, regressio

Preface

At first I would like to thank my supervisors for their patience and assistance during this project, while the research questions had to be adapted against the availability of usable data sets. Further, I would like to extend my gratitude to the three data sources that were utilized in the thesis. Especially, *ADS-B Exchange* for allowing me to use their data free of charge. And finally, I would like to thank my family and friends for their help and motivation while I was working on this thesis.

Jyväskylä May 11, 2020

Jarno Kiesiläinen

Glossary

AI	Artificial intelligence.
ANN	Artificial neural network
API	Application programming interface.
Flight plan	Proposed route plan for aircraft flight.
FMI	Finnish Meteorological Institute.
JSON	JavaScript Object Notation.
MAE	Mean absolute error.
ReLU	Rectified linear unit. Type of activation function used in artificial neural networks.
SVM	Support-vector machine.
SVR	Support-vector regression.
WGS 84	World Geodetic System 1984, standard earth centric coordinate system and coordinate system used by GPS.

List of Figures

List of Tables

Contents

1 Introduction

Ability to predict air traffic patterns is important for air traffic control and air space surveillance. Air traffic control is interested in making air traffic as efficient and safe as possible and air space surveillance is looking for unusual activity in the surveyed air space. Improved aircraft traffic prediction would allow airspace surveillance to better focus on interesting events, expending less effort on regular and normal traffic, therefore, increasing safety.

The growing amounts of available data have driven the rise of machine learning and artificial intelligence. Machine learning can be a very useful tool for many problems like data mining, automation, outlier detection and analytics. It is used in multiple applications, like for example web searches, drug discovery, advertising, manufacturing and many others. Machine learning has therefore become a part of many information systems (Witten and Frank 2005; Domingos 2012).

Machine learning has been applied to many aviation tasks like trajectory and aircraft climb prediction (Leege, Paassen, and Mulder 2013; Alligier, Gianazza, and Durand 2015) and Finnish defense forces technology strategy concludes that information systems have a very central role or even the most important role in the defensive capabilities of the armed forces. Systems for decision making and data mining are included in the most critical technologies to be developed by the Finnish Defense Forces (Puolustusvoimat 2012). Could machine learning and historical data be used to make better aircraft travel time predictions?

1.1 Research problem and questions

The novel work done in the course of this thesis are the feature extraction from the selected data sources and the use of this data to estimate aircraft travel times. The feature extraction is interesting because multiple data sources are used in combination to produce features such as weather along the aircraft travel route. Also the travel times are estimated to other route waypoints than just the final destination. The work is split into three parts, data collection, feature extraction and evaluation of the four machine learning methods for the travel time estimation. Following research questions have been identified as the most relevant for this

thesis:

1. Can aircraft travel times be reliably predicted from the available information?
2. How to preprocess and combine very different kinds of data from multiple sources?
3. How to vectorize the data for machine learning methods and what features to select?
4. Which machine learning methods could work best for this type of problem?

1.2 Structure of the thesis

The first chapter contains an introduction to the application area of this thesis and the motivation for the work. The main research questions are also found in this chapter.

Chapter 2 introduces machine learning in general. It contains an overview of different approaches and uses of machine learning methods, common problems faced in machine learning and a short history of artificial intelligence and research in this field. The chapter also describes the details of the machine learning methods used in the empirical part of the thesis.

Chapter 3 is about the dataset that was created for this thesis. The first part of the chapter is about the open data sources and the data collection process. The second part describes, how the preprocessing steps were used for the collected data to transform and combine it to be used by the machine learning methods to make predictions.

The 4th chapter has a description of the model search method and the model comparison methods. Also the model search parameters and results of the four methods, boosting, SVR, ANN and random forest method are presented in this chapter.

The last chapter has discussion and reflection on the work done in this thesis. An interpretation of the model search results and details about the practical implementation of the empirical research task are included. Concluding, remarks about future research ideas and improvements for the work done in this thesis are discussed.

2 Machine learning and artificial intelligence

Artificial intelligence (AI) is tightly tied to machine learning. Most AI-systems use one or multiple machine learning methods as parts of their implementation (Bini 2018). There is not a single clear-cut definition of artificial intelligence but generally artificial intelligence tries to build entities that can understand and manipulate an environment that is more complicated than the entities themselves are.

Artificial intelligence is often measured by comparing it to human intelligence when performing specific tasks. Arguably the most famous test of machine intelligence is the Turing test invented by Alan Turing in the 1950s. The test's idea is very simple: a human questioner asks questions in written form and the machine answers them in text. After questioning the machine passes the test if the human cannot tell whether the respondent is human or a machine. This test is not interested in how artificial intelligence works internally but just if its behavior is sufficiently close to human behavior and that it can fool the human. Another way of approaching AI is to mimic the way humans think but it would require us to understand how the brain works or at least how humans think. The third approach is through rationality with rational rules that are set to achieve the best result in the task (Russell and Norvig 2009).

The first actual limited work on artificial intelligence was done in the 1940s like Donald Hebb's rule for updating and modifying connections between neurons nowadays called Hebbian learning and McCulloh's and Pitts' work "A Logical Calculus of the Ideas Immanent in Nervous Activity". Research on artificial intelligence started more prominently in the 1950s with the influential figures like John McCarthy and Alan Turing. A single considerable step in AI was when Arthur Samuel wrote a checkers program that was able to learn and improve its gameplay by playing against human and machine opponents. AI research continued to develop gradually as applications like stock price prediction and playing poker emerged. Some applications like machine language translation stayed elusive targets and in the late 1980's skepticism and lack of merits of artificial intelligence resulted in significantly less research funding in AI. This period of slower research was called *the AI Winter*. Two things can be generally be seen after the 1980s in successful AI methods: solid mathematical theory behind the method and an increasing amount of data to create robust models (Russell

and Norvig 2009; Buchanan 2006; McCordyck 2004).

In the late 90s interest in AI increased with the growth of computing power and AI found use in narrow applications like data mining and Web bots. With the renewed interest in AI researchers started to look again into more general intelligence. This resulted in a realization that AI sub-fields like speech or image recognition in themselves are not enough but a multidisciplinary approach that can take uncertainties to account is required. One long-standing goal of AI research is Human-level AI which is inspired by Herbert Simon's words: "machines that think, that learn and that create" (Simon and Newell 1958). Another goal or research sub-field of AI is Artificial General Intelligence (AGI). The target of the AGI research is to find a universal agent or algorithm that can learn any task in any environment. Artificial General Intelligence is related closely to Human-level AI as "any task in any environment" can be thought to be any task that a human can learn. Still, most research and application of AI are in narrow applications like autonomous driving cars or drug discovery (Russell and Norvig 2009; McCordyck 2004).

Especially now that most AI systems use machine learning methods, the quality and amount of the input data is very important. Most of AI and machine learning research history the focus has been in the algorithms and methods but it has been shown that a good algorithm with a modest amount of data will perform worse than a mediocre algorithm with a lot of data (Domingos 2012; Russell and Norvig 2009; Buchanan 2006; McCordyck 2004).

In machine learning the goal is to find a model based on available data that generalizes the studied phenomena so that the model can be used to make predictions with new samples of data. Witten and Frank (2005) describe learning in the machine learning context as: "Things learn when they change their behavior in a way that makes them perform better in the future". This definition is not based on knowledge like learning is usually understood but in performance which is more appropriate in the computing context. Most machine learning methods are based on an iterative automatic process of improving the model's performance where different performance metrics can be used to measure the model's learning progression. This automatic model generation phase of machine learning and data mining is usually referred as "training" the model. Training is probably a more accurate description of what happens in these methods as it implies a repetitive process that does not necessarily include

thinking like learning does.

2.1 Types of machine learning

Machine learning can be divided into three groups by their learning type: supervised, unsupervised and reinforcement learning. The goal of supervised learning is to create a model that can produce input-output pairs based on known examples. Generally this task is called classification if the outputs are discrete values like labels or classes and regression if at least one of the output values is continuous (Bishop 2006). These training values are used to train a model to make predictions for input values with unknown outputs. Four supervised methods selected to be tested in the thesis are described in more detail in section 2.3.

Ensemble methods (sometimes called committees) are a subset of supervised methods. In ensemble methods multiple different learners are combined to make predictions that are much better than the learners could make on their own. The results are often aggregated by some method to produce one prediction. A common example of an ensemble method is boosting, where a collection of weak predictions is improved with new weak predictions in iterations called boosts (Bishop 2006; Liaw and Wiener 2002; Drucker 1997; Polikar 2012).

In unsupervised learning the correct labels for input values are unknown and the learning algorithm is used to discover hopefully useful classes from the dataset. An example of unsupervised learning is clustering methods where the idea is to partition the data points to nonempty disjoint clusters which are taken to be the classes of the dataset. Since the sample's location in the feature space relative to other samples is the only information available, the distance metric that is used is a very important parameter in clustering methods. Clustering methods are often used in data exploration and data discovery to gain insights into how the data is structured (Zaki and Wagner Meira 2014).

In the third kind of learning, reinforcement learning, the model is trained by reinforcement signals given by an outside system. This usually done by a system that measures the learner's performance and it is up to the learner to discover how to produce better results. Instead of looking for correct input-output pairs the focus is on the performance of the model. Generally there are two different approaches. Some search techniques and genetic models search the

model space to find models or individuals that perform better. The other approach is to use statistics and dynamic programming techniques to evaluate the utility of actions that can be taken to improve the model. An important concept of reinforcement learning is ”exploration vs. exploitation”. As model evaluation is often expensive it can be thought of as the cost of the learning process. This means that the algorithm has to decide if to ”explore” and find new solutions or to ”exploit” and improve current solutions (Kaelbling, Littman, and Moore 1996; Kotsiantis 2007).

2.2 Regression

In this thesis the objective of the machine learning system is to predict the time an aircraft will take to reach its next waypoint. This means that the machine learning problem is a regression problem and not a classification problem as the targets are not discrete values. Regression analysis is part of statistical modeling and an important tool in many applications for estimating relationships within data. By creating a regression model one can use regression for prediction and this is exactly what some machine learning methods do.

Simplest kind of regression model is a linear regression model. The simple linear regression model is a line that has the form of $\beta_0 + \beta_1 x = y$ where β_0 and β_1 are the estimator variables. The line represents the relationship between the variables x and y. In a two-dimensional case if one knows a sample value of x and not the value of y the model can be used to estimate the value of y. The estimator variables β_0 and β_1 are found by minimizing a cost function with known x_i, y_i pairs. Probably the most popular type of cost function is the ordinary least-squares:

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2 . \tag{2.1}$$

Geometrically in two dimensions, this means that the regressions residuals $y_i - (\beta_0 + \beta_1 x_i)$ i.e. the vertical distances from the regression line to the actual y-values are minimized (Weisberg 2005; Bishop 2006).

Since most parameters have a more complicated relationship than can be linearly modeled, the obvious next step is to move to polynomial and other non-linear models. The nth degree two-dimensional polynomial model is $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n$. Polynomial regres-

sion can model much more complicated relationships in the data and a polynomial model with a high enough degree can fit perfectly to all training data. This means that polynomial regression is prone to overfitting especially if one uses higher-order polynomials (Weisberg 2005).

2.3 Machine learning methods

Four different popular machine learning methods were selected to be tested in this thesis: artificial neural networks, support vector regression and two ensemble methods called random forests and boosting. In this section the basic principles of the chosen methods are detailed.

The following notation is used in the descriptions of the machine learning methods. y or y_i refers to the target value belonging to the input data element $\mathbf{x}$ or $\mathbf{x}_i$. f or f_i is the predicted value produced by a prediction model where the target value was y or y_i. Vector variables are marked with bold typeface like $\mathbf{x}$, compared to scalars with normal typeface like x.

2.3.1 Artificial neural networks

Original inspiration for neural networks comes from the structure of our brains. Our brains are a network of interconnected neurons that have the ability to organize so that they can perform computations. Current artificial neural networks (ANN) are magnitudes of simpler than the network in our brain but some key concepts remain the same. ANNs are constructed with artificial neurons and the simplest and most popular (artificial) neuron has inputs, output, sum junction and an activation function. The inputs are collected to a single value with the sum junction which is usually a weighted sum. The summed input is then given to the activation function which is the output of the neuron. An overview of this structure is displayed in figure 1. When choosing an activation function one has to make sure that its derivative is available since it is required in the network training. The simplest type of activation function is of course the identity function but rectified linear unit (ReLU) and sigmoid functions are generally seen to perform better in practice (Glorot, Bordes, and Bengio 2011; Haykin 2009).

Commonly neurons are organized to layers where the neuron's inputs are only connected to

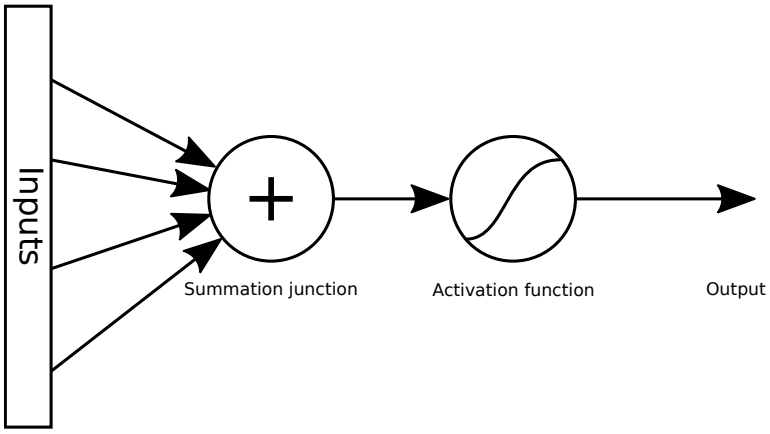

Figure 1. A simplified diagram of a neuron with four inputs and a single output connection.

the outputs of the previous layer. A layer is said to be fully connected if all neurons in a layer are connected to all neurons of the adjacent layers and partially connected if some of the connections are missing. A graph of a fully connected neural network is portrayed in figure 2. One type of ANN is the feed-forward network where layers are connected to one direction so that none of the connections form loops. Opposed to feed-forward networks in concurrent neural networks layers can form loops that allow the network to have a ”memory” so temporal relations existing between the inputs can be accounted like for example with audio or video data (Haykin 2009).

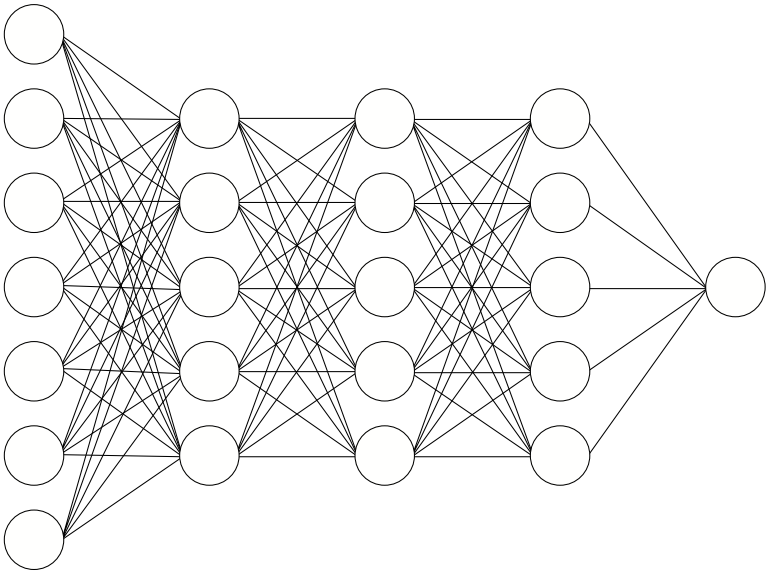

Figure 2. A graph of an artificial neural network with seven inputs, three hidden layers with five neurons each and one output.

Training neural networks is done by an algorithm called backpropagation. The idea of backpropagation is to first do a forward pass with a training data sample to find the error of the network for that sample. Then, using the error gradient the weights of the neurons can be adjusted toward correct values in reverse order. This is called the backward pass and these two passes are executed multiple times to train the network.

Defining a single neuron the input of neuron j in layer l is:

$$v_j^{(l)}(n) = \sum_i w_{ji}^{(l)}(n) f_i^{(l-1)}(n), \tag{2.2}$$

where $w_{ji}^{(l)}(n)$ is the weight corresponding to output of ith neuron in layer $l-1$, $f_i^{(l-1)}(n)$ is the output of neuron i in layer $l-1$ and n is the iteration number. Note that the first layer is the input layer where $f_i^{(1)}(n) = x_i^j$, where x_i^j is the ith value of the training data element $\mathbf{x}_j$. The final output of the neuron j then is:

$$f_j^{(l)}(n) = \varphi_j(v_j(n)), \tag{2.3}$$

where φ is the activation function of the neuron. Since the outputs of neurons of the previous layer are required for calculating the neuron's output, the output values of the neurons have to be computed a layer at the time from the first hidden layer to final network output layer. This is called the forward pass of the network.

In the backwards pass the weights of the neurons are adjusted with gradient descent. Weights are updated with the delta rule:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha \left[\Delta w_{ji}^{(l)}(n-1)\right] + \eta \delta_j^{(l)}(n) f_i^{(l-1)}(n), \tag{2.4}$$

in which l is the layer number, j the neuron number, i the neuron number of the previous layer, $\delta_j^{(l)}(n)$ are the local error gradients of the network, α is momentum constant and η is the learning-rate. Momentum constant is usually a positive value used to control the feed-back loop of the weight change. Learning rate is used to slow the gradient descent and limit oscillation. Small learning rate makes the training steady and predictable but also raises the computation cost since more iterations are required for the network to converge (Haykin 2009). For the output layer the gradient can be calculated with the error signal and the derivative of the activation function:

$$\delta_j^{(L)}(n) = e_j^{(L)}(n) \varphi' \left(v_j^{(L)}(n)\right), \tag{2.5}$$

where $e_j^{(L)}(n) = y_j - f_j^{(L)}(n)$ in which the y_j is the target output and for the hidden layers the local gradients of the next layer are required:

$$\delta_j^{(l)}(n) = \varphi_j'\left(v_j^{(l)}(n)\right)\sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n). \tag{2.6}$$

This leads to the backward pass as the new weights have to be calculated in the direction from the last layer to the first layer. Steps of the training for a single training data element are listed in the algorithm 1. The steps of the algorithm should be repeated for all of the training data (Haykin 2009).

Algorithm 1 Steps of backpropagation training for one training data element $\mathbf{x}_i$ without dropout.

1. Calculate the network output f_i for the data element $\mathbf{x}_i$ with the equations 2.2 and 2.3.
2. Calculate the gradient for the output layer with equation 2.5.
3. Update the weights of the output layer with the equation 2.4.
4. For each of the hidden layers:
   (a) Calculate the gradient with equation 2.6.
   (b) Update the weights with equation 2.4.

Dropout is a technique that can be applied to a neural network in the training phase. The idea of the dropout is to disable a neuron randomly with a configurable probability during a training iteration. This has been shown to increase the network performance because in essence it makes the network structure change for every iteration. Thus the final network is an ensemble of many different learners (Ba and Frey 2013).

Artificial neural networks have an exceptionally high number of hyper-parameters. The structure of the network is one parameter that has an infinite number of possible structures in the number of layers and neurons. This is made even more difficult by the fact that many of the parameters can be different for each layer or neuron. Each neuron could in principle have a different activation function or dropout probability.

2.3.2 Boosting

Boosting is an ensemble method. In boosting the classification or regression is based on an ensemble of weak learners that are trained one by one consecutively. The boosting algorithm used in this thesis is a modified version of the AdaBoost algorithm introduced by Freund and Schapire (1997). In every iteration training data is weighted so that training of new learners is prioritized to the data elements that had the largest error with the previous learners. The training data is selected with replacement meaning that some of the data elements can appear multiple times or not at all in a training set for a learner. Weak learners can be any type of regression machines but the usual choices are regression decision trees and neural networks (Friedman 2001). New learners are trained until the loss value of the latest learner trained is over a threshold. Then predictions can be made for new unknown data with the ensemble regression machine. To make a prediction each weak learner gives its prediction for the data element. Then a weighted median of the predictions is chosen as the result (Drucker 1997; Bishop 2006).

The regression boosting algorithm defined by Drucker (1997) is as follows:

1. Assign initial weights $w_i = 1$ for the data samples.
2. Repeat steps 3 to 7 until $\bar{L} > \frac{1}{2}$ ($\bar{L}$ defined in equation 2.8).
3. Select n samples where each sample has probability $p_i = \frac{w_i}{\sum_{i=1}^{N} w_i}$ of being selected.
4. Grow a regression decision tree with the selected training data.
5. Make predictions $\hat{f}_i$ with the regression tree for all of the training data.
6. Calculate measure of confidence for the new tree

$$\beta_t = \frac{\bar{L}}{1-\bar{L}}, \tag{2.7}$$

   where the average loss is

$$\bar{L} = \sum_{i=1}^{n} L_i p_i \tag{2.8}$$

   and loss for single data sample is

$$L_i = \frac{|\hat{f}_i - y_i|^2}{\sup_{i\in\{1\ldots N\}} |\hat{f}_i - y_i|^2}. \tag{2.9}$$

7. Update weights $w_i = w_i \beta_t^{1-L_i}$ for the training data.

Additional learning rate parameter can also be introduced during the training phase. This parameter scales the contribution of each new learner. The algorithm results in T regression machines. To make a cumulative prediction f_i with the ensemble calculate the weighted median

$$f_i = \inf \left\{ \hat{f}_t : \sum_{i:\hat{f}_i \leq \hat{f}_t} \log\left(\frac{1}{\beta_i}\right) \geq \frac{1}{2} \sum_{i=1}^{T} \log\left(\frac{1}{\beta_i}\right) \right\}. \tag{2.10}$$

To interpret the weighted median sort the predicted values $\hat{f}_i$ from smallest to largest. Then sum the items $\log\left(\frac{1}{\beta_i}\right)$ in the order of the $\hat{y}_i$s until the inequality is satisfied. Then $\hat{f}_t$ is the weighted median f_i that is the output of the model. Note that the weighted median is the regular median if the weights β_i are all equal in the equation 2.10 (Drucker 1997).

2.3.3 Random forests

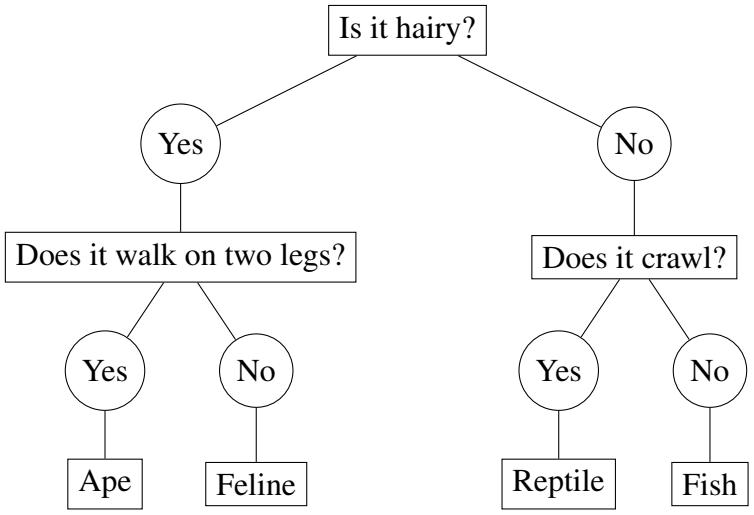

Figure 3. An example of a decision tree for classifying an animal.

Random forest is an ensemble learning method that combines multiple simpler learners. The learners are decision trees (hence the name forest) that are constructed with the help of training data. Decision trees are a tree structure used to recursively partition data based on hierarchical rules. Data is split on every node until it reaches a leaf node that represents the class label of the value (see figure 3 for a simple example). Every internal node has at least two child nodes and a split rule that splits the input value to one of the child nodes. Even though decision trees can be created by hand with knowledge of the application area,

automatic generation or automatic rule induction is used in machine learning. Decision trees can also be used for regression by replacing the class label with a constant value which in essence partitions the input space to discrete output values.

The ensemble tree learners are trained with a training algorithm called bagging (bootstrap aggregation). In bagging the training dataset is split randomly and the subsets are used to generate regression trees. Random forest augments this algorithm by adding another layer of randomness by selecting a random subset of features in every tree node rule induction. The steps of the algorithm are

1. Select n_{tree} subsets from the training data.
2. For each subset grow a regression tree with m features used for rule induction in every node. Special case is when m is same as the number of features. Then the algorithm is the standard bagging algorithm.
3. To predict values of unknown data, average of the n_{tree} trees predictions is taken.

To estimate the error of the random forest the training data elements that were not used for a single tree can be used to measure its error. The errors can be aggregated in the same way as the regression result to gain error rate for the random forest (Liaw and Wiener 2002; Breiman 2001).

To grow the decision trees in the random forest method CART algorithm is used (Breiman et al. 1984). The root node of the tree contains the training data subset and a binary split is always used in the nodes. In the original CART algorithm the tree is grown without stopping rule until no more splits can be made but some implementations have a max depth stopping rule. For selecting the splitting rule in a regression tree node the least-squares cost function can be used.

After the tree has been grown to maximum depth pruning can be performed. Pruning is done with a test dataset by minimizing the tree misclassification cost and the number of tree leaves. The objective function in pruning is

$$Ra(T) = R(T) + a|T|, \tag{2.11}$$

where $R(T)$ is the training data misclassification cost and $|T|$ is the number of leaf nodes in

the tree. The variable a can be chosen and it is used to control if to prioritize in misclassification cost or the size of the tree. Leaves are pruned from the tree one by one as long as the pruning objective improves (Steinberg 2009).

2.3.4 Support-vector regression

Support-vector regression (SVR) is based on Support-vector machines (SVM) (Cortes and Vapnik 1995; Drucker et al. 1996). Support-vector machines are a binary classification method and SVR is an extension to the SVM that allows it to be used for regression. The principal idea of SVM is to find a hyperplane that maximally separates the two classes. A subset of data points from the training data is used to construct such a hyperplane. These points are called support-vectors giving the method its name. The original SVM finds a linear hyperplane to separate the classes but as it is not always possible to separate the classes linearly a soft error margin is introduced to allow some training samples to remain on the wrong side of the plane and kernel method can be used to transform the input space so that the classes become separable.

SVR takes this idea and applies it to regression. The objective function of SVR is

$$U \sum_{j=1}^{N} L\left[y_j - F(\mathbf{x}_j, \mathbf{w})\right] + \|\mathbf{w}\|^2, \tag{2.12}$$

where $\mathbf{x}_j$s are the training sample vectors and y_js their target values, L is the loss function and F is the representation of the SVR parametrized by vector $\mathbf{w}$. The regularization term $\|\mathbf{w}\|^2$ is added to the objective function to get better generalization since it promotes the use of all of the features. The constant term U is used to control the focus of the minimization to the loss function or the regularization term. The function f is usually defined as

$$F(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) K(\mathbf{x}, \mathbf{w}) + b, \tag{2.13}$$

where N is the number of training samples, α_i^*, α_i and b are values to choose.

A method called ”kernel trick” is often applied in support vector machines. It is useful when the data points are not linearly separable in the classification case or when linear regression can not approximate the data well enough. The idea of the kernel method is to transform

the input data to a feature space in which the input data is linearly separable. In practice the actual input-feature space mapping is not required since solving the support-vector problem requires only the dot-product of the input vectors in the feature space. Therefore only a kernel function calculating the dot-product is required which is the function K in the equation 2.13 (Hofmann 2006). Generally four variations for the kernel function are used:

$$K_{linear}(\mathbf{x}, \mathbf{w}) = \mathbf{x}^t \mathbf{w}, \tag{2.14}$$

$$K_{polynomial}(\mathbf{x}, \mathbf{w}) = (\mathbf{x}^t \mathbf{w} + 1)^d, \tag{2.15}$$

$$K_{RBF}(\mathbf{x}, \mathbf{w}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{w}\|^2}{\sigma^2}\right), \tag{2.16}$$

$$K_{neural}(\mathbf{x}, \mathbf{w}) = \tanh(\kappa \mathbf{x}^t \mathbf{w} + \theta), \tag{2.17}$$

where σ, κ and θ are constants that can be chosen, $(\cdot)^t$ signifies transpose and d is a polynomial exponent (Suykens and Vandewalle 1999).

The function L in equation 2.12 is the ε-error margin loss function that is used to define the soft-margin

$$L = \begin{cases} 0 & \text{if } |y_i - F(\mathbf{x}_i, \mathbf{w})| < \varepsilon \\ |y_i - F(\mathbf{x}_i, \mathbf{w})| - \varepsilon & \text{otherwise.} \end{cases} \tag{2.18}$$

In essence the loss function creates a band around the target values where the loss is zero. The problem is a quadratic programming problem and it ends in a solution where either or both of the α_i, α_i^* variable pair from the equation 2.13 are zero. The training samples $\mathbf{x}_i$ corresponding to non-zero α_i or α_i^* are called the support-vectors (Drucker et al. 1996).

2.4 Challenges in machine learning

2.4.1 Curse of dimensionality

Many classification and regression tasks can be seen as a curve-fitting problem. This means that the learner's good generalization of input-output pairs is just a good interpolation of the training data. To find a good decision boundary or an interpolation, a dense sample of the data is needed. In high dimensional data this is a problem as the number of dimensions increases, the volume of the space increases exponentially. Therefore more data is required

for a dense training dataset. The three-dimensional intuition that we humans have does not work in higher dimensions. Phenomena like the ”mass” of multivariate Gaussian distribution is increasingly more in a shell around the mean and not close to the mean like one is used to in lower dimensions. These effects mean that finding a good interpolation becomes difficult in high dimensions as any similarity-based reasoning does not work in the same fashion as they do in a lower-dimensional space. This phenomenon is called the curse of dimensionality and it is the main reason why one should try to limit the number of features, as finding good generalizations in lower dimensions is usually easier. Thankfully, in many applications the data is spread in the space non-uniformly, meaning that the data is close to some lower-dimensional manifold that can be taken advantage of when designing the learner (Domingos 2012; Haykin 2009; Kanevski, Pozdnoukhov, and Timonin 2009).

2.4.2 Hyper-parameters

The performance of many machine learning methods depend on parameters that have to be chosen by a human before training the model. For example the number of trees in the random forest or the kernel of the SVM method. These parameters are sometimes called hyper-parameters. Selecting these parameters is often a difficult problem in itself since using grid search is often infeasible because training an individual classifier might take a significant amount of time and in methods with multiple hyper-parameters one might end up with a large number of combinations. Therefore many heuristics and methods have been devised for hyper-parameter selection. In this thesis the random grid search is used to find good hyper-parameters (Klein et al. 2016; Kanevski, Pozdnoukhov, and Timonin 2009; Bergstra and Bengio 2012).

2.4.3 Overfitting

A major problem with many supervised machine learning methods is overfitting. Overfitting happens when the model is fitted too close to the training data and the model performs exceptionally well when tested against the training data but much worse with samples outside the training set. This behavior is easiest to see with polynomial curve fitting. Fitting a lower order polynomial to a set of observations produces a simple predictor with some

error. Higher-order polynomial will allow the predictor to have a smaller error but will cause the curve to oscillate between the observations which usually does not reflect the underlying phenomenon being modeled. The problem boils down to selecting the number of parameters in the model. Increasing the number of parameters increases the fit of the model but Occam's razor suggests that you should use only what is necessary. The opposite of overfitting is underfitting. This is when the model fails to capture some of the effects that were supported by the data (Burnham and Anderson 2002; Bishop 2006; Haykin 2009; Kanevski, Pozdnoukhov, and Timonin 2009).

A useful tool borrowed from statistics that can be used in the model selection is called cross-validation. The idea of cross-validation is to randomly split the sample dataset into multiple parts: training datasets and validation datasets. The idea behind this split is to train or generate the model with the training set and then validate the model with data that was not used in the training. If the model starts to overfit we notice this by much better prediction accuracy in the training set compared to the validation set. If the training is iterated multiple times it is possible that the model starts to overfit to the validation set too and then a third test set can also be kept aside to test the final model (Haykin 2009; Bishop 2006).

3 The research dataset

The ideal dataset for this study would be one that has features generated only from data that would also be available for an air traffic surveillance system. This way the features could be calculated from the data live by the system and predictions could be made as soon as the data becomes available. The dataset produced for this thesis tries to follow this principle and the most significant deviation is the flight plan data, since no official flight plan data was available at the time of this thesis. A real surveillance system would of course have an access to official flight plans, but since such a source is not available, a service primarily intended to provide flight plans for flight simulators was used. This also leads to a need to associate flight plans with the flights. Association is not as reliable of a system as using the flight identification information which would be possible with official flight plan data.

An interesting direction also chosen for this thesis is the use of weather radar images along with the other flight information, even though forecasts and other more processed weather information would be available. The radar images were chosen as a data source because it is a relatively novel approach in this application and as such data could also be available in a real surveillance system.

3.1 The open data sources

Three data sources are used in this thesis: *ADS-B Exchange*, *Flight Plan Database* and *Finnish Meteorological Institute*'s open data API. All three data sources can be queried with HTTP (Hypertext Transfer Protocol) GET-method (Fielding et al. 1999) with request URI that consists of the server name, a resource path and additional parameters like authentication. For example a request URI could be `https://api.flightplandatabase.com/nav/airport/EFHK` which would request server api.flightplandatabase.com for resource nav/airport/EFHK i.e. information about Helsinki-Vantaa international airport. *Finnish Meteorological Institute*'s API works without any authentication, *ADS-B Exchange* API requires an API key for all request and *Flight Plan Database* requires authentication for some special requests.

3.1.1 ADS-B Exchange


```
{
  "postime": "1573733981595", "icao": "4614A3",
  "reg": "OH-HVF", "type": "AS32", "wtc": "2",
  "spd": "80.8", "altt": "0", "alt": "750",
  "galt": "476", "talt": "", "lat": "60.27063",
  "lon": "24.975807", "vsit": "0", "vsi": "0",
  "trkh": "0", "ttrk": "", "trak": "192.1",
  "sqk": "0012", "call": "FNG200", "gnd": "0",
  "trt": "4", "pos": "1", "mlat": "0", "tisb": "0",
  "sat": "0", "opicao": "", "cou": "Finland",
  "mil": "0", "interested": "0", "dst": "6.15"
}
```


Figure 4. A JSON sample of one aircraft observation from *ADS-B Exchange*.

The *ADS-B Exchange* service (`https://adsbexchange.com/`) is used as an aircraft flight information data source. ADS-B Exchange uses a community of people that host feeders with ADS-B receivers that listen to the SSR (secondary surveillance radar) reply pulses from aircraft and send them to ADS-B Exchange servers. Then on the server multilateration (Zhou, Jun Li, and Lamont 2012) is used to calculate the position of the aircraft. The aircraft location along with other auxiliary information is then published on the website.

Aircraft data can be accessed through JSON API from the *ADS-B Exchange* web service. Data is requested with an HTTP GET -request. Multiple types of requests can be made to the service. For example one can request all aircraft known to the service, find all aircraft from a geographic location or search aircraft by tag like 'military'. Returned data is a single JSON object which has a list of aircraft information corresponding to the query. There are altogether 32 different features that might not all be present. The fields that are of interest in this thesis are of course the aircraft location fields: `lat` (latitude coordinate), `lon` (longitude coordinate) and `alt` (aircraft altitude). Other important fields are the `reg`-field which is the aircraft tail number (i.e. the ICAO unique identifier), `postime` (time at which the plane was observed at the aforementioned coordinates) and the `to` and `from` fields which are the

departure and arrival airports and their ICAO codes. Latitude and longitude coordinates are in the WGS 84 (EPSG:4326) coordinate system and altitude is in feet. See a sample of the data in figure 4.

3.1.2 Flightplan database

Flight plan is a document that indicates a proposed flight time and route for an aircraft. Flight plans are usually submitted to air traffic services well before the aircraft departs and they contain relevant information for the air traffic service authorities. For example aircraft identification, route, cruising speed and level, fuel endurance and others (International Civil Aviation Organization 2005). Flight plans provide an initial guess for the timetable of the flight but since flight plans are often submitted hours before the actual flight departs and as weather, traffic and other causes affect the progress of the flight, flight plans are not reliable.

Flight Plan Database (`https://flightplandatabase.com/`) has a large collection of flight plans meant primarily for flight simulation use. For this reason most of the plans do not have any flight identification or time information. This presents a problem since one can not simply associate a flight plan to a flight with any unique identifier.

Flight Plan Database includes a JSON API. The API has multiple functions like submitting and editing flight plans but for this thesis the most useful function of the API is the flight plan search. The search has multiple parameters but here only two are needed: ”toICAO” and ”fromICAO”. When looking for a flight plan for a flight we get the flight’s departure and arrival airport’s ICAOs and use those to search for flight plans. The results usually contain many flight plans but most of them are duplicates. Luckily the result JSON contains an ”encodedPolyline” field which represents encoded polyline for the flight plan route. Because the route is the same for any duplicate plans present, also the ”encodedPolyline” will be the same, so it can be used filter out the duplicates. Figure 5 has a sample of a single flight plan returned from the API search query.

```
{
 "createdAt": "2019-09-11T20:35:38.000Z",
 "cycle": {"id": 19, "ident": "FPD1909", "release": 9, "year": 19},
 "distance": 219.547516013406, "user": None, "waypoints": 5,
 "downloads": 3, "flightNumber": None,
 "encodedPolyline": "oucoJssjwC~tu@vazLf{a@rpbGjsi@v{aHzyv@n{xJ",
 "fromICAO": "EFHK", "fromName": "Helsinki Vantaa Intl",
 "id": 2202730, "likes": 0, "tags": ["generated"],
 "maxAltitude": 24900, "popularity": 1568752538,
 "toICAO": "ESSB", "toName": "BROMMA",
 "updatedAt": "2019-09-11T20:35:38.000Z",
 "route": {"nodes": [
  {"ident": "EFHK", "name": "Helsinki Vantaa Intl",
   "lat": 60.3172, "lon": 24.9633, "alt": 0,
   "type": "APT", "via": None},
  {"ident": "UMUGI", "name": None,
   "lat": 60.0372, "lon": 22.6947, "alt": 24900,
   "type": "FIX", "via": None},
  {"ident": "USITU", "name": None,
   "lat": 59.8586, "lon": 21.3658,"alt": 23200,
   "type": "FIX", "via": {"ident": "Y360", "type": "AWY-HI"}},
  {"ident": "LUPET", "name": None,
   "lat": 59.6403, "lon": 19.8764, "alt": 13100,
   "type": "FIX", "via": {"ident": "Y360", "type": "AWY-HI"}},
  {"ident": "ESSB", "name": "BROMMA",
   "lat": 59.3544, "lon": 17.9416,
   "alt": 0, "type": "APT", "via": None}
 ]}
}
```


Figure 5. A sample flight plan search from the *Flight Plan Database* API. The query returns a lot of values of which many are not of interest for us.

3.1.3 Finnish Meteorological Institute

Finnish Meteorological Institute (FMI) is a government facility responsible for producing weather services in Finland. FMI website provides an open data API (`https://en.ilmatieteenlaitos.fi/open-data`) with a lot of different services like current weather data and data provided by weather prediction models. In this thesis we are interested in the wind and precipitation data.

The API includes access to radar GeoTIFF images. There are images for radar reflectivity, rainfall intensity, precipitation, rain classification, cloud top height and radial velocity. For us the most interesting are the precipitation and radial velocity images. Radial velocity is a measure of wind towards the radar. Since velocity is measured from reflected radar pulses only radial movement of particles can be detected and orthogonal movement can not be measured. This also means that velocity can only be measured reliably when there are enough particles in the air like snow. There are multiple precipitation accumulation images for different time periods.

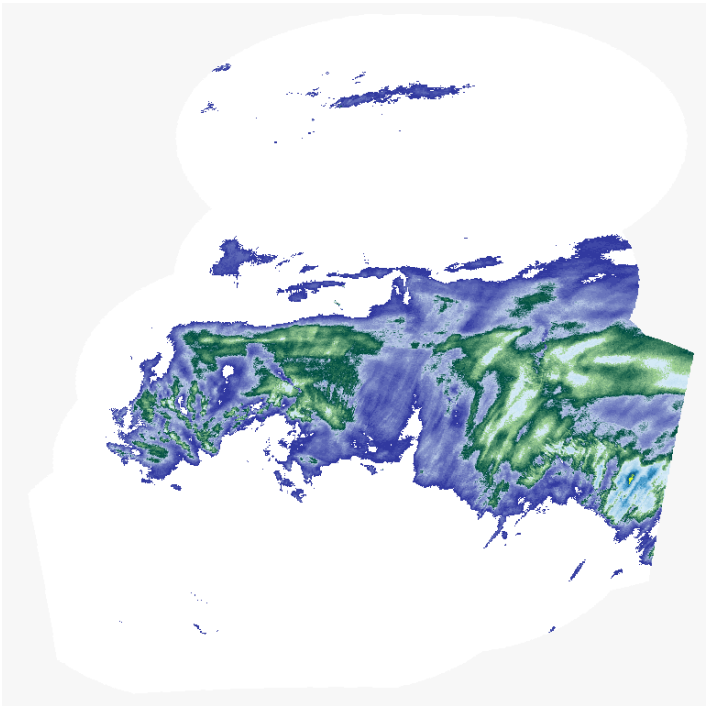

Figure 6. A sample composite image of multiple radar precipitation maps from the FMI Open Data API. The image covers Finland completely.

3.1.4 Data collection

The data is collected with a *Python* script in intervals of 30 seconds. Aircraft data is requested from the *ADS-B Exchange* API and weather radar data from the *Finnish Meteorological Institute*'s open data API. Aircraft data is requested from inside a circle around Southern Finland and appended to a JSON file. Because *ADS-B Exchange* does not directly track the aircraft but relies on user observations of the aircraft SSR reply pulses, the API can only provide the latest data that it has available. This means that even though the data is sampled at even intervals the actual observations are usually not from that exact moment.

The weather data is radar images from the *Finnish Meteorological Institute*'s open data API. The radar data is also requested every 30 seconds. The radar images are not updated every 30 seconds and therefore the images are hashed with the *sha256* hash algorithm. The hashes are used to check if two consecutive images from the same radar are the same image. If so the image is not stored to save the device disk space. Flight plans are not yet included at this point because the flight plans are assigned with a heuristic that is easier to apply after all the flight data has been collected.

3.2 Data generation

After all of the *ADS-B Exchange* -aircraft data and the weather images have been collected, the aircraft observation data is processed to flight objects. The flight objects contain multiple consecutive observations of an aircraft. Then based on the flight objects, flight plans are searched from the *Flight Plan Database* API and assigned to the flight objects.

The dataset generation is a relatively complex process and takes a lot of processor time. Therefore as soon as it can be deduced that a flight object cannot be processed to a dataset element it should be skipped. This means that flights that do not have a flight plan or if the flight does not reach any of the waypoints given in its plan it will be skipped. Also all observations that are far away from any of the radar maps are skipped since then no weather features could be generated.

The flight objects are split into sets of four consecutive observation samples. Each of these

sets of four are used to generate one dataset element. The general steps for generating a dataset element are the following:

1. Find the next waypoint that the aircraft will reach and calculate the time and distance to that waypoint.
2. Find radar maps that are the latest for the observation time.
3. Calculate precipitation and wind velocity value for the data element.
4. Calculate the median velocity for the aircraft during those four observations.
5. Calculate the time of day value (seconds since midnight).
6. Delta-encode the samples (for a description of delta encoding see 3.2.7).

The input data for one data element includes a series of four observations from *ADS-B Exchange*, the locations of the waypoints of the selected flight plan and a list of radar images. Processing the weather data results in arrays of radar image pixel values. Sometimes multiple pixel arrays can result if the flight crosses multiple radar images. These values are converted to precipitation and wind values with the color map method (described in 3.2.5). For precipitation the sum of the array values is given as the final result and for wind values their mean is used. The values of one random data element from the final dataset are listed in the table 1.

Description	value
Time (ms) to the next waypoint (target value)	1349681
Distance (km) to the next waypoint	146.8259520486311
Time of day in hours	21.3167
Mean of the wind speed along the route	2.001
Sum of the precipitation values	0.0
Mean velocity of the aircraft	0.203512
Wake turbulence class	2
Longitude	26.31015
Latitude	58.907822
Altitude	34075.0
Milliseconds since January 1st 1970	1581189476062
Longitude delta 1	-0.065896
Latitude delta 1	0.050494
Altitude delta 1	-1150
Time delta 1	32815
Longitude delta 2	-0.057867
Latitude delta 2	0.044132
Altitude delta 2	-1025
Time delta 2	28503
Longitude delta 3	-0.051144
Latitude delta 3	0.038778
Altitude delta 3	-900
Time delta 3	26654

Table 1. One data element from the dataset.

3.2.1 Selecting flights from observations

ADS-B Exchange API queries return a list of latest aircraft observations. To be useful in this research we need a time series of aircraft locations. This means that the list of observations has to be combined to form a time series. Since the ADS-B data contains the aircraft's unique tail number it is the easiest property to use to group the observations of the aircraft. If the recording covers a longer time the same aircraft can have easily flown multiple flights so further splitting by departure and destination has to be done. Even further some aircraft fly the same route often so the flight data has to be split if there is a long enough time between the observations meaning that they belong to different flight instances.

3.2.2 Flight plan association

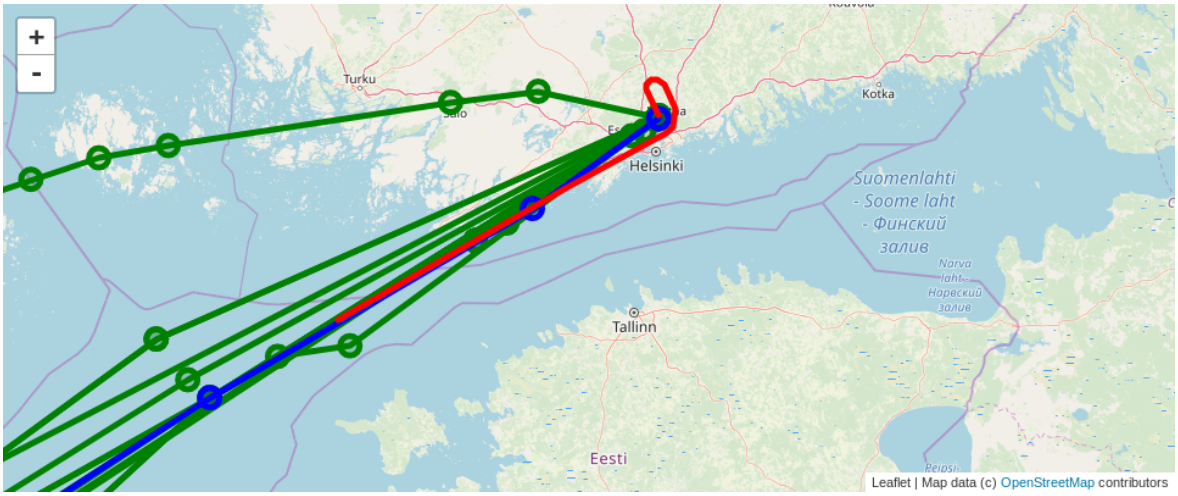

Figure 7. A visualization of flight plan selection for a flight from Amsterdam Airport Schiphol to Helsinki-Vantaa airport. Red polyline is formed from the flight observations. Green polylines are flight plans with the same departure airport and destination airport as the flight data has and the circles are flight plan waypoints. The selected flight plan is colored blue.

When flight plans are submitted to the aviation authorities they have the aircraft's "tail number" i.e. the ICAO registration number included for connecting the actual aircraft and the proposed flight plan so that air traffic control does not have to guess which aircraft a flight plan is assigned to. *Flight Plan Database* does not have any identification information as-

signed to the flight plans even though the plans are real-world flight plans. This is because the data is supposed to be used for flight simulation and not for real aviation purposes so any real-time information is removed to discourage their use in real aviation. Therefore some heuristic has to be used to select a plausible flight plan for a flight from many possible plans. The heuristic selected in this thesis is to find the flight plan that is geometrically ”closest” to the flight path. Closeness is measured by calculating the distance of the flight observations from the polyline defined by the flight plan.

The flight plan finding heuristic is defined in the algorithm 3 (appendix A). In the algorithm, in equation 6.1 the log of the distance is taken because aircraft do not always follow the line segments defined by the flight plan especially close to airports. Using the distance directly excessively punishes any aircraft departing too far away from the planned route. Without the log operation flight plan crossing the aircraft flight path gets selected over a plan running parallel to the flight path. Flight plan selection is illustrated in figure 7.

3.2.3 Kinematic features

Multiple features are extracted from the aircraft observations and flight plans. The simplest feature is the distance from the aircraft's position to the next flight plan waypoint. The distance is calculated with the Haversine formula (see 3.2.4). Another feature is the mean velocity of the plane for the last four observations. The mean velocity is calculated by summing the distances between the observations and dividing them with the total time. The distances are again calculated with the Haversine formula.

The third feature generated from the data is delta encoding (see subsection 3.2.7) of the last four observations. Fourth feature, wake turbulence class, is taken directly from the *ADS-B Exchange* data. The wake turbulence class is a rough measure of how much air the aircraft displaces in its wake which is an indication of the aircraft's size.

3.2.4 Haversine formula

Haversine formula is an important equation used originally in naval navigation to find one's position. If two positions on a sphere are known the formula can be used to calculate the

great circle distance of the two points. The great circle means the circle that is found with the cross-section of a sphere and a plane that goes through the center point of the sphere. This is useful because calculating the norm of the difference of two polar coordinates does not produce a distance on the sphere surface like it does on a flat surface.

The derived Haversine formula for distance is as follows:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right), \tag{3.1}$$

where λ_1, λ_2, φ_1, φ_2 are longitude and latitude of the two points, d is the distance and r is the radius of the sphere which is the mean radius of the earth in this thesis (Alam et al. 2016; Chopde and Nichat 2013; Brummelen 2013).

3.2.5 Weather features

The GeoTIFF format images retrieved from the *Finnish Meteorological Institute* are not directly available as the values that the radar measured but a color mapped image so the values have to be converted to real floating-point values. The image data is a 768×768 array of color map index values. The color values can be matched with the color bar index provided with the API giving one the actual value for any of the index values. The radial data has a range of negative 48 m/s to positive 48 m/s of radial velocity and the precipitation map has values ranging from 0 mm of rainfall to 250 mm of rainfall.

Wind velocity and precipitation are selected from the radar images with a line algorithm. The aircraft observation location and the next waypoint location are connected with a line. This line is then rasterized to the radar image coordinates with the Bresenham algorithm to gain a list of radar image indexes. If any of the points are within bounds of the radar image the values at the points are stored. Then a sum or a mean of those values is used as the feature.

3.2.6 Bresenham algorithm

An algorithm is needed for selecting values from weather radar images. A line rendering algorithm called the Bresenham algorithm is selected for this task because it is a simple way to get grid points connecting the aircraft and the next waypoint. The algorithm works in two-

Algorithm 2 The Bresenham algorithm to draw a line from (x_0, y_0) to (x_1, y_1).

1. Calculate the deltas: $d_x = x_1 - x_0$ and $d_y = y_1 - y_0$.
2. Set $y_i = 1$
3. **If** $d_y < 0$: Set $y_i = -1$ and $d_y = -d_y$.
4. Set $D = 2d_y - d_x$ and $y = y_0$
5. Let $P = \emptyset$ be a set containing the pixel coordinates.
6. **For** x from x_0 to x_1 **do**:
   7. Add (x, y) to P.
   8. **If** $D > 0$: Set $y = y + y_i$ and $D = D - 2d_x$.
   9. Set $D = D + 2d_y$.
10. **Return** List of pixel coordinates P.

dimensional integer space by stepping values over one axis and deciding if to move along the axis or diagonally to the next pixel. Depending on the slope of the line either x- or the y-axis is stepped (Pitteway and Watkinson 1980; Wright 1990).

The Bresenham algorithm is described in algorithm 2. The default version of the algorithm only handles gradients between -1 and 1 but if one swaps the x and y all gradients are possible. Vertical and horizontal gradients should also be checked before as in those cases the pixel array is easy to generate.

The algorithm has a single deficiency in this application. Since the coordinates are sphere surface coordinates a line between two points is not always the shortest route. The shortest route is always along a great circle of the sphere and the error between a Cartesian line and a great circle line is bigger closer to the poles of the sphere. This error presents itself in this thesis in that ”wrong” indices might be selected from the radar images. But since we are interested in the weather close to the aircraft and its destination location this error is thought to be small enough to be ignored.

3.2.7 Delta encoding

The four time-position coordinates of the aircraft are converted to a delta list. In delta encoding the actual values are not used but the difference of the values. Delta encoding of a series of data $\{v_1, v_2, \ldots, v_n\}$ can be calculated with the formula 3.2.

$$\hat{v}_i = \begin{cases} v_i, & \text{if } i = 1 \\ v_i - v_{i-1}, & \text{otherwise.} \end{cases} \tag{3.2}$$

For example the list $[4,5,7,8,5]$ would be converted to $[4,1,2,1,-3]$. Delta encoding is used to compress data when the changes of the values in a series contain less information to store than the complete values (Smith 1997). In this thesis the delta encoding is not used to compress the data but to extract the movement of the aircraft from the time-location series. Even though the aircraft observation location relative to the earth is important the way the aircraft moves is even more important and this is why delta encoding is used. When the observations are delta encoded only the first coordinate and time is left to the dataset and the rest of the observations are converted to deltas which represent the movement of the aircraft.

3.3 Resulting dataset

#	Description
1	Distance to next waypoint in kilometers
2	Time of day in hours
3	Wind speed on the route
4	Precipitation on the route
5	Mean velocity of the aircraft
6	Wake turbulence class of the aircraft
7-22	Delta encoding of the last four observations time positions

Table 2. Description of the final dataset features.

The input data for the data generation consist of 49099 radar images, 13147 flight plans and 2610993 ADS-B observations that result in 2976 flight objects between the 29th of December 2019 and the sixth of March 2020. After processing the input data the result is 14258 data

elements which each have 22 features and a target value. 338 of the elements have target values that are over 2 hours or a mean velocity that is over 1200 km/h. Those elements are considered outliers and are removed leaving 13920 elements to the final dataset.

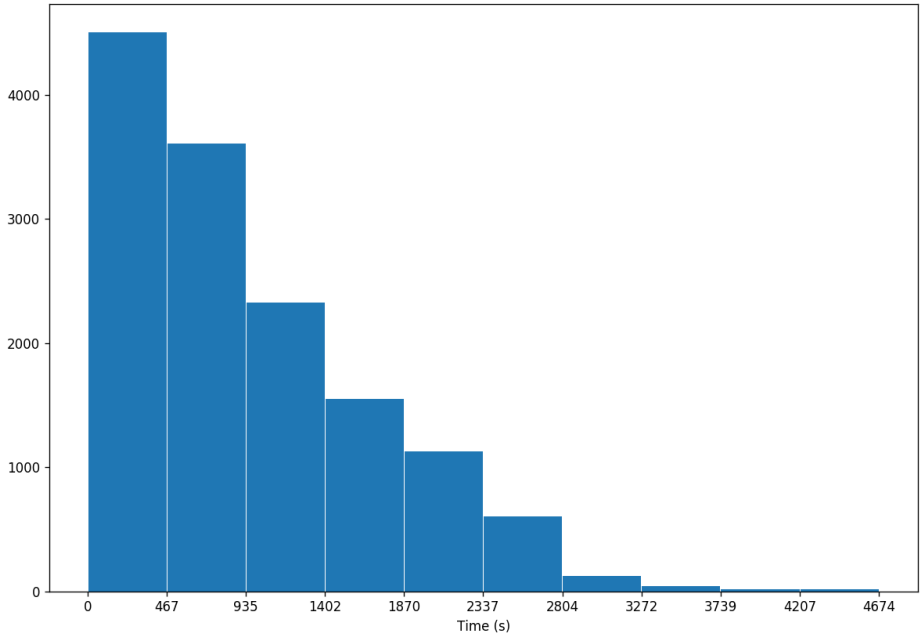

Figure 8. Distribution of dataset target values.

Histogram of the target values can be seen in figure 8. From the figure can be seen that smaller values (aircraft reaching next waypoint sooner) are more common. The 22 features are described in table 2 and an example data element is given in table 1. A more comprehensive description of how the values were generated can be found in section 3.2. Histograms of the features (excluding the delta feature) are plotted in figure 9.

As seen from figure 9 the most common precipitation and wind velocity values are close to zero with only a fraction of the elements having larger values. Bad weather should increase the flight time as the aircraft speed should be lower. The wake turbulence has 4 possible values: none, light, medium and heavy. No data elements correspond to the light class, most aircraft being in the medium or heavy class with some having no classification. Heat map of the distance and time values of the dataset is plotted in figure 10. From the figure a diagonal trend can be seen which indicates that the distance feature is probably a good predictor of the travel time left. This is reasonable because as the aircraft gets closer to the waypoint the travel time should also decrease.

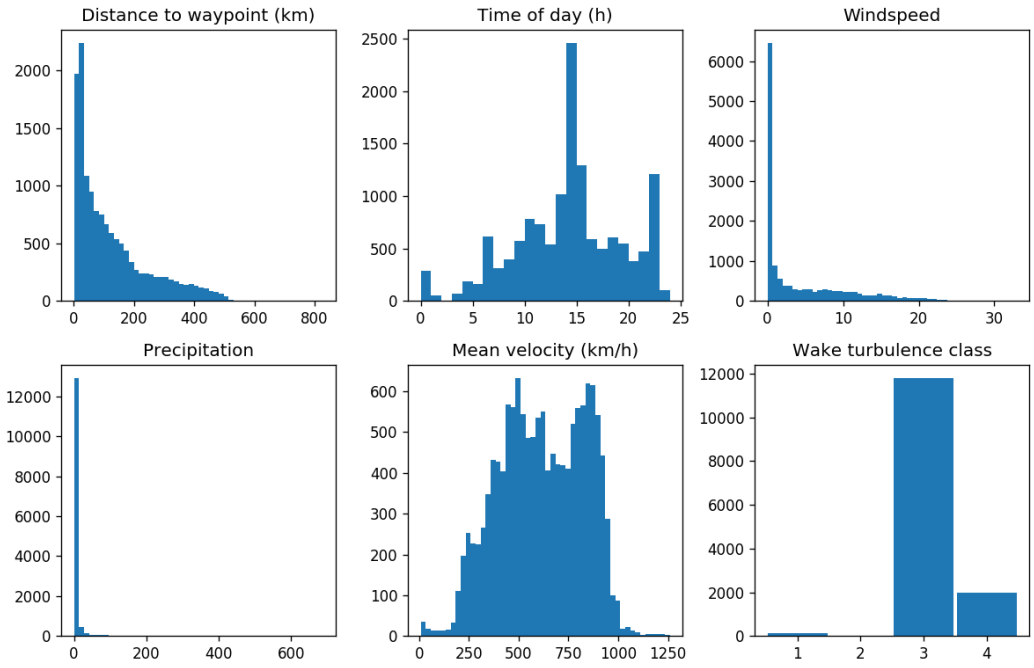

Figure 9. Distribution of dataset feature values.

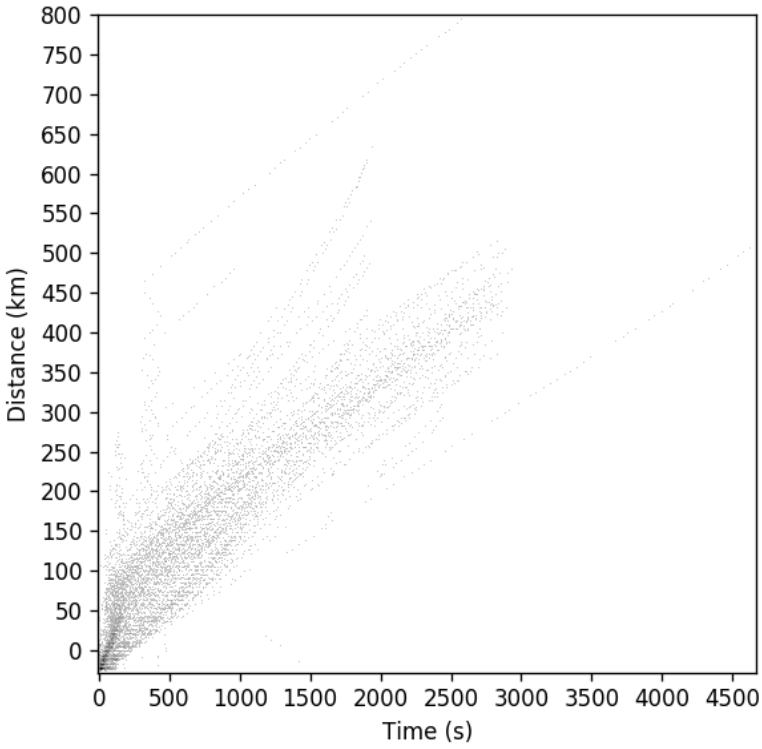

Figure 10. A heat map of time to the next waypoint in the horizontal axis and distance to the next waypoint in the vertical axis.

4 Model search parameters and results

Three datasets are derived from the complete data for testing: all of the data, data without the precipitation and wind speed features and data without the delta feature. The different datasets are used to gain insight if the features actually contribute to the models having better predicting ability. 5% of the dataset (696 of 13920 samples) was set aside for testing and was not used in the model training.

Two metrics are used to measure the models' performance: the mean absolute error (MAE) and the r^2-score. Mean absolute error is defined as

$$\frac{\sum_{i=1}^{N} |y_i - f_i|}{N}, \tag{4.1}$$

where y_i is the target value, f_i is the value predicted by the model and N is the number of samples. Mean absolute error is an interesting metric because it is often easy to interpret. As the target values are in seconds the absolute error of the model for that element is also in seconds. Therefore the mean absolute error tells how many seconds did the model err on average.

The r^2 is also called the coefficient of determination. It is defined as

$$r^2 = 1 - \frac{\sum_{i=1}^{N} (y_i - f_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2}, \tag{4.2}$$

where y_i, f_i and N are the same as in equation 4.1 and $\bar{y}$ is the mean of the y_i. It is often used as a measure of goodness of the fit of a linear model and can be interpreted to estimate the percentage of the variance of the target values explained by the feature variables (Renaud and Victoria-Feser 2010; Barrett 1974).

Before the dataset is used in the model training it is scaled. Scaling the features often improves the performance of the machine learning methods and some methods like support-vector methods actually require scaling because the method is not scale-invariant. Scaling is done so that the training data mean is moved to zero and then the features are divided by the feature's standard deviation. Features are scaled with the following formula:

$$k_i' = \frac{k_i - \bar{k}}{s_k}, \tag{4.3}$$

where k_i is the kth feature of the ith data element, $\bar{k}$ is the mean of the kth feature and s_k the standard deviation of the kth feature for the training set. An unfortunate side effect of the scaling is that all input values have to be scaled before they can be used by the model (Bring 1994).

The testing environment selected for the thesis is the *Python*-programming environment because of the number of ready-made libraries and implementations. An important Python module used in this thesis is the *Jupyter notebook* module. It allows easy testing and visualization of different models and data exploration. For the machine learning methods tested in this thesis the *scikit-learn* library was used except for the artificial neural networks for which the *Keras* library was used.

4.1 Grid search and random search

Grid search is a brute force approach to the hyper-parameter problem (see 2.4.2). It has multiple advantages that keep it relevant after years of research in alternative ways of finding good hyper-parameters. Its key advantages are that it is easy to understand and easy to implement in any programming environment and since the machine learning instances are independent of each other parallelization is usually trivial. The idea of the grid search is to select a set of parameter values for each of the hyper-parameters that one wants to try and then run the machine learning training with every combination of those selected values. This means that also grid search optimization suffers from the curse of dimensionality which is the method's greatest weakness. If $\left(L^{(1)}, L^{(2)} \dots L^{(K)}\right)$ is the set of selected values for parameters $1, 2 \dots K$ the number of variables in the grid search optimization is $\Pi_{k=1}^{K} |L^{(k)}|$. The number of combinations grows quickly when the number of parameters K is increased making the problem difficult (Bergstra and Bengio 2012).

A natural extension to grid search is the random search. The parameter values are selected randomly from a distribution of values instead from a grid. Even though the random values are not distributed as evenly as in a grid the random search seems to perform better in practice than the grid search. One can select how many times the parameters are sampled from the distribution to control the number of models tried. This becomes very advantageous

especially when the number of possible combinations in grid search is large and it is not feasible to try them all with limited resources (Bergstra and Bengio 2012).

In this thesis the random search is favored to the grid search. The implementation that is used is the *scikit-learn* library's implementation called "RandomizedSearchCV". The library is given a trainable model and a list of parameter distributions to be tried. Then the library deals with calling all the required functions for selecting the parameters and also training and evaluating the models.

The library uses cross-validation in the training of the models. Cross-validation works by splitting the training dataset to S different subsets. The model is trained with $S-1$ number of the subsets and then validated with the subset that was left out. Then a different subset is left out and the process is repeated until all subsets are left out once resulting in S iterations. Finally mean of the validations scores is taken as the final model performance score (Bishop 2006; Haykin 2009).

4.2 Polynomial and linear regression models

Parameter	Values
Degree	$\{2,3,4,5\}$
Bias	yes / no

Table 3. Parameters for the polynomial regression model search.

Polynomial regression is used as a baseline model that the other models can be compared to. 24 different polynomial models and a simple linear model was tested. The linear model with only the distance feature used had a mean absolute error of 243.5 seconds and r^2-score of 0.7915. The parameters of polynomial models are in table 3. Performing grid search yields eight different combinations of the parameters to be used in the three datasets so random search is not necessary in this case. The degree refers to the maximum degree of the polynomial model and bias selects if the bias term is used in the model.

The results are in table 4. The best result was achieved with no delta features with MAE of about three minutes. The dataset without the delta feature has the lowest number of features

	All features	No weather	No delta
MAE	201.2 s	207.4 s	**183.4 s**
r^2-score	0.5159	0.4877	0.6981
degree	2	2	5
bias	yes	yes	no

Table 4. The polynomial regression model search results.

in total and it seems that the polynomial models suffer from a high number of features so the best model was found with the lowest number of features.

4.3 Random forest

Parameter	Values
Number of estimators	$\{2^i : i = 3,4,\ldots 9\}$
α	Uniformly sampled from range $[0,2]$
Maximum depth of the tree	$\{8,9,10,11,\ldots,100\}$ or no limit
Maximum number of features used for a split	All, square root of the number of features or logarithm of the number of features

Table 5. Parameters for the random forest model search.

Parameters used in the random forest model search are listed in table 5. The number of estimators refers to the number of trees generated and the maximum depth is the stopping rule for the tree growing. The parameter α is the pruning parameter that controls the pruning process where zero value means no pruning. The maximum number of features sets the number of features used in the splitting. 250 candidates for each dataset were considered totaling 750 candidates.

Parameters and results of the best models found are in table 6. With all of the features the mean absolute error is a bit over one and half minutes. Without the weather feature the MAE was 2.6 % worse and without the delta feature 16.6 % worse.

	All features	No weather	No delta
MAE	**94.0 s**	96.4 s	109.6 s
r^2-score	0.9511	0.9501	0.9433
α	0.6251	0.4991	0.2407
# of est.	512	512	256
Max feat.	all	all	sqrt
Max depth	95	41	39

Table 6. The random forest model search results.

4.4 Boosting

Parameter	Values
Number of estimators	$\{2^i, i \in 3 \dots 8\}$
Loss function	linear, squared or exponential
Learning rate	uniformly sampled from range $[0.01, 1.0]$
Tree depth and alpha[1]	$\{(10, 0.0), (20, 0.1), (40, 0.5), (50, 1.0)\}$

Table 7. Parameters for the boosting model search.

The parameters used in the boosting model search are in table 7. The number of estimators is the number of trees used for model creation. The loss function is the function used to measure the model error when updating the weights. The learning rate is used to control the contribution of new regressors to limit overfitting. Tree depth and alpha are parameters for the decision tree growth similar to the random forest method. 500 different models were evaluated for each dataset totaling 1500 candidates. The results for the model search can be found in table 8. The best model using all of the features had a 4.27 % better score than without the weather features and 28.39 % better score than without the delta features.

	All features	No weather	No delta
MAE	**39.8 s**	41.5 s	51.1 s
r^2-score	0.9940	0.9933	0.9858
# of est.	256	256	256
Loss function	squared	squared	linear
Learning rate	0.5765	0.4694	0.8329
Tree depth and α	20 and 0.1	50 and 1.0	20 and 0.1

Table 8. Results of the boosting model search.

4.5 Support-vector regression

Parameter	Values
Kernel	RBF: $K_r(x,x') = \exp(-\gamma\|x-x'\|^2)$
	Sigmoid: $K_s(x,x') = \tanh(\gamma\langle x,x'\rangle + r)$
	Polynomial: $K_p(x,x') = (\gamma\langle x,x'\rangle + r)^d$
ε	uniformly sampled from range $[0.1, 60.0]$
C	uniformly sampled from range $[0.1, 10.0]$
d	$\{2,3,4,5\}$
γ	uniformly sampled from range $[0.1, 1.0]$
r	0.0

Table 9. Parameters used in the SVR model search.

Parameters for the support-vector regression model search are in table 9. The kernel parameter is the K-function used in the equation 2.13. The ε is the error margin that the regressor is not punished for. d, γ and r are parameters of the kernel functions. d is used only for the polynomial kernel as it is the maximum polynomial power. γ is a coefficient used for scaling the input in the kernels. The r-parameter is an independent term used in the kernel. Parameter C controls the optimization to prefer either the estimation or the regularization.

The results and parameters of the best models are listed in table 10. Using all of the features produced a 2.1 % better model when measuring with the MAE than without the weather feature and 5.2 % than without the delta feature. The results are not very good since the best polynomial model has a performance very similar to the best SVR models found.

	All features	No weather	No delta
MAE	**183.9 s**	187.8 s	193.5 s
r^2-score	0.7983	0.8246	0.8140
Kernel	poly	poly	poly
C	7.948	3.060	6.116
d	2	2	4
r	1.714	1.727	1.701
γ	0.353	0.631	0.382
ε	5.65	19.28	17.43

Table 10. Results of the SVR model search.

4.6 ANN

Parameters for the artificial neural network search are in table 11. The dropout parameter is the dropout probability of neurons in the training phase. Layers define the network's structure where each number represents a layer with that number of neurons in the layer. The learning rate is the η parameter of the back-propagation algorithm (see 2.3.1). The activation function is the activation function used in the neurons. The epoch controls the number of training iterations for the network. For each dataset 150 different models were tried totaling 450 different models. Parameters and error scores of the model search are in table 12. Using all features produced 30.2 % better mean absolute error than without the weather features and 11.0 % better error score than without the delta features.

Parameter	Values
Dropout probability	$\{0.0, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$
Learning rate	$\{0.01, 0.05, 0.1, 0.15, 0.2, 0.25\}$
Activation function	$\text{ReLU} = \varphi(x) = \begin{cases} x, & x > 0 \\ 0, & \text{otherwise} \end{cases}$
	$\text{linear} = \varphi(x) = x$
	$\text{sigmoid} = \varphi(x) = \frac{1}{1+e^{-x}}$
Layers	$\{30, 20, 10\}$ or
	$\{100, 100, 100\}$ or
	$\{30, 35, 20, 15, 10\}$ or
	$\{50, 50, 50, 20, 20, 10\}$ or
	$\{30, 40, 50, 40, 30, 10, 5\}$ or
	$\{100, 100, 100, 100, 100, 100\}$

Table 11. Parameters used in the ANN model search.

	All features	No weather	No delta
MAE	**112.4 s**	146.4 s	124.8 s
r^2-score	0.9180	0.9129	0.9009
Activation	sigmoid	ReLU	ReLU
Drop out	0.2	0.01	0.0
Epochs	206	263	326
Learning rate	0.025	0.05	0.025
Layers	$\{100, 100, 100\}$	$\{50, 50, 50, 20, 20, 10\}$	$\{30, 35, 20, 15, 10\}$

Table 12. Results of the artificial neural network model search.

5 Discussion

The research questions posed in this thesis were:

1. Can aircraft travel times be reliably predicted from the available information?
2. How to preprocess and combine very different kinds of data from multiple sources?
3. How to vectorize the data for machine learning methods and what features to select?
4. Which machine learning methods could work best for this type of problem?

With the polynomial model the mean absolute error was around three minutes and with a simple linear model about four minutes setting the baseline results. Three minutes is not an impressive result as the mean for the target values is around 900 seconds and the median bit under 800 seconds. The machine learning regression model search results are promising and indicate that it is possible to predict the travel times with such an accuracy using these methods to give a positive answer to the question 1.

	All features		No weather		No delta	
	MAE	r^2	MAE	r^2	MAE	r^2
Polynomial	201.2 s	0.5159	207.4 s	0.4877	183.4 s	0.6981
Boosting	**39.8 s**	0.9940	41.5 s	0.9933	51.1 s	0.9858
Random forest	94.0 s	0.9511	96.4 s	0.9501	109.6 s	0.9433
SVR	183.9 s	0.7983	187.8 s	0.8246	193.5 s	0.8140
ANN	112.4 s	0.9180	146.4 s	0.9129	124.8 s	0.9009

Table 13. A summary of the model search results.

Looking at the question 4 the best model was a boosting model with a test set mean absolute error of well under one minute. If the model could make predictions of this accuracy in practice the model performance would be sufficient for real-world usage. Interestingly boosting performed much better than the random forests method even though both methods are based on the same regression tree learner. This may be attributed to the boosting methods ability to focus on the ”difficult” data. The best random forest model had a mean absolute error of about one and half minutes and the artificial neural networks performed a bit worse with an

MAE of about two minutes. Support-vector regression performed the worst of the four with an error of about 3 minutes making perform about as good as the polynomial model. The summary of the results can be found in the table 13.

5.1 Data preprocessing

Another objective of the thesis was to find how to prepare the data from multiple sources for the machine learning methods. Answering the research questions 2 and 3 required a considerable amount of effort and multiple heuristics were created to convert the separate data so that it could be used by the machine learning methods. Both combining the ADS-B data with the flight plan data and weather radar images consisted of multiple steps.

Flight plans were required to know when an aircraft has reached its waypoints. Since no official flight plan source was available an alternative source with identification information missing had to be used. This meant that there was no direct way to know which plan belongs to a flight if any. An algorithm had to be developed to select plans that fit the flight data in a realistic fashion.

The weather data was utilized to see if it would increase the prediction accuracy as the prevailing weather should affect air traffic. The input data used was weather radar images of radial wind velocity towards the radar and one hour precipitation detected by the radar. There were two problems to be solved with using the weather images. The first one was that the data was color mapped images without the original observation values. Thankfully a legend for the color map was provided which could be used to convert the color values back to the values measured by the radar. The second problem was how to combine the values of the images with the flight data. The selected solution was to use a line connecting the aircraft and the target waypoint to sample the data from the radar image.

The delta and weather features improved the regression with all of the methods except with the polynomial model which performed best with the lowest number of features. It seems clear that the delta feature is very important as it often improved the result significantly. The weather feature was also beneficial for the regression problem improving the result moderately compared to the delta feature.

5.2 Practical notes

Even though the goal of this thesis was not to evaluate these methods technically a small evaluation of the machine learning implementations and tools available was done before selecting the methods. This was done so that the effort required to test the methods would not be overwhelming since trying and comparing the methods was the goal. The *scikit-learn*-library provided a great set of tools for testing and using different kinds of machine learning methods and processes. Implementations for all of the methods except the artificial neural network were available in the library which made testing them much more effortless.

A few problems were countered while testing the methods. The non-linear support-vector regression with polynomial kernel seemed to often not to converge or it converged really slowly. Therefore iteration cap of 500 000 iterations was set for the method. None of the other SVR kernels had this problem. Another problem was with using the artificial neural networks with the random search method. When concurrently testing multiple models, the search would run into concurrency error that would make the whole search fail. The problem seemed to be operating system-related because when the search was run on a Windows system it failed and with a Linux system it succeeded.

Of the three data APIs used the *ADS-B Exchange* API required the most effort. The API is available to anyone freely if one hosts an ADS-B feeder for the service and does not use the data for commercial purposes. Hosting a feeder requires a small computer like Raspberry Pi with an ADS-B antenna that is capable of listening for the aircraft SSR-pulses. After acquiring permission to use the API, its use is quite straight forward. The *Flight Plan Database* API is completely free except for a daily limit on some requests. Also the API was documented very clearly with examples of all of its capabilities. The *Finnish Meteorological Institute* API is also free and open for anyone. The problem with the data API was that it has a lot of data available with all kinds of different types of requests. There exists some documentation of the API online on the FMI website in Finnish and English with the content sometimes differing between the languages. The documentation is split to multiple pages with some of the documentation being integrated in the data queries itself making the use of the API a bit confusing at times.

5.3 Improvements and future research

A possible deficiency in the dataset produced in this thesis might lay in the recording of the data. The time of day, season of the year and many other factors affect the amount and type of air traffic. Recording for short periods might not capture all kinds of aviation as widely as one would hope, were the recording period longer. To give one example, no observations for the summer period are present in the dataset. This might affect the prediction quality during summer, since most of recreational aviation with small aircraft takes place during the summer when the flying weather is better. Thankfully the data was recorded before the COVID-19 crisis, which halted much of the civil air traffic.

Another obvious problem in this thesis was the lack of access to complete flight plan data. There was no comprehensive analysis of how well the flight plan association heuristic worked. It was just assumed that the easy cases were captured and more difficult associations were lost. Improving the research in this front would require either contacting the authorities having control of the complete flight plan data or improving the flight plan association algorithm. This was left out of this thesis because of time constraints and because an alternative source was available.

If the research was continued improvements and chances to the data features could be made. More weather features like thunderstorms or cloud cover could be easily derived from the available *Finnish Meteorological Institute* data which could improve the results even more. In addition to weather features, also new kinematic features could be developed like aircraft flight level or difference of aircraft travel direction and direction to the next waypoint. Also the possibility of other, additional data sources could be explored.

To find better prediction models, firstly new types of models could be researched and tried. Moreover each of the methods selected in this thesis could be explored more closely. Even the methods that did not perform optimally could be given a second change because transformations of the input data and better hyper-parameters could improve the performance of the models. Artificial neural networks would have especially many options to explore as the network structures tried in this work were relatively simple and not too many training iterations of the networks were executed. But this was out of the scope of this thesis since the

goal was to test the viability of this kind of system to gain insight into what works, if anything. Randomized grid search was used to search for the models, which is a well tested and established method to find hyper-parameters for the models. A next step in improving the model search would be to try genetic methods briefly mentioned in the section on machine learning to optimize the hyper-parameters. Also a more formal approach to the number of models could be used as now the amount of models tried was loosely based on the time it took to execute the search.

Instead of estimating the remaining travel time the data could be also used to make travel delay predictions. The overall traffic amount and the data that was used in this thesis could be used to estimate the change in travel time compared to average. Estimating the delay or arrival ahead of time could be more useful than knowing the remaining flight time in some applications.

Finally a prototype of the prediction system could be built. The prototype should be made in a way that it is possible to change the machine learning model. This prototype would be fed data as it becomes available and predictions would be made. Later when the aircraft reaches their waypoints the predictions could be checked.

6 Conclusion

Air traffic control and air space surveillance are time-consuming and important tasks. Better predictability of air traffic would allow resources to be directed better. Historical flight data and weather data could be used to estimate future aircraft flight durations.

Machine learning methods can be used to create regression models that can estimate relationships between variables. Many different methods exist that have their strengths and weaknesses and selecting the best method for the required task can be difficult.

In this thesis data from three different sources for aircraft flight data, aircraft flight plans and weather data were combined to create a dataset for predicting aircraft flight durations. Then the data was used to generate regression models with four different machine learning methods: support vector regression, boosting, random forests and artificial neural networks. The different methods' performance in the flight duration estimation was evaluated. Also the effects of a weather- and a delta feature were evaluated.

Based on the regression model evaluation results it is possible to use these methods and data to estimate aircraft travel time with a reasonable accuracy. The results also show that the weather- and the delta feature improved the estimation accuracy. The improvement was substantial enough that it can be concluded that these features are beneficial for the estimation task.

Bibliography

ADS-B Exchange. https://adsbexchange.com/.

Alam, C. N., K. Manaf, A. R. Atmadja, and D. K. Aurum. 2016. "Implementation of haversine formula for counting event visitor in the radius based on Android application". In *2016 4th International Conference on Cyber and IT Service Management,* 1–6. doi:10.1109/CITSM.2016.7577575.

Alligier, R., D. Gianazza, and N. Durand. 2015. "Machine Learning and Mass Estimation Methods for Ground-Based Aircraft Climb Prediction". *IEEE Transactions on Intelligent Transportation Systems* 16, number 6 (): 3138–3149. ISSN: 1558-0016. doi:10.1109/TITS.2015.2437452.

Ba, Jimmy, and Brendan Frey. 2013. "Adaptive dropout for training deep neural networks". In *Advances in Neural Information Processing Systems 26,* edited by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, 3084–3092. Curran Associates, Inc. http://papers.nips.cc/paper/5032-adaptive-dropout-for-training-deep-neural-networks.pdf.

Barrett, James P. 1974. "The Coefficient of Determination - Some Limitations". *The American Statistician* 28 (1): 19–20. doi:10.1080/00031305.1974.10479056. eprint: https://doi.org/10.1080/00031305.1974.10479056. https://doi.org/10.1080/00031305.1974.10479056.

Bergstra, James, and Yoshua Bengio. 2012. "Random Search for Hyper-Parameter Optimization". *J. Mach. Learn. Res.* 13, number 1 (): 281–305. ISSN: 1532-4435.

Bini, Stefano A. 2018. "Artificial Intelligence, Machine Learning, Deep Learning, and Cognitive Computing: What Do These Terms Mean and How Will They Impact Health Care?" *The Journal of Arthroplasty* 33 (8): 2358–2361. ISSN: 0883-5403. doi:https://doi.org/10.1016/j.arth.2018.02.067. http://www.sciencedirect.com/science/article/pii/S0883540318302158.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning.* Springer-Verlag. ISBN: 0387310738.

Breiman, Leo. 2001. “Random Forests”. *Machine Learning* 45 (): 5–32.

Breiman, Leo, Jerome Friedman, Charles J. Stone, and R.A. Olshen. 1984. *Classification and Regression Trees.*

Bring, Johan. 1994. “How to standardize Regression Coefficients”. *The American Statistician* 38 (3). doi:10.1080/00031305.1994.10476059.

Brummelen, Glen Van. 2013. *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry.* Princeton University Press. ISBN: 9780691148922. http://www.jstor.org/stable/j.ctt1r2fvb.

Buchanan, B. 2006. “A (Very) Brief History of Artificial Intelligence”. *AI Magazine* 26 (4).

Burnham, K. P., and D. R. Anderson. 2002. *Model Selection and Multinmodal Inference.* Springer-Verlag. ISBN: 978-0-387-22456-5.

Chopde, Prof. Nitin R., and Mr. Mangesh K. Nichat. 2013. “Landmark Based Shortest Path Detection by Using A* and Haversine Formula”. *International Journal of Innovative Research in Computer and Communication Engineering* 1, number 2 ().

Cortes, C., and V. Vapnik. 1995. “Support-vector networks”. *Machine Learning* 20, number 3 (): 273–297. ISSN: 1573-0565. doi:10.1007/BF00994018.

Domingos, P. 2012. “A Few Useful Things to Know About Machine Learning”. *Commun. ACM* (New York, NY, USA) 55, number 10 (): 78–87. ISSN: 0001-0782. doi:10.1145/2347736.2347755.

Drucker, Harris. 1997. “Improving Regressors using Boosting Techniques”.

Drucker, Harris, Chris J.C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. 1996. “Support Vector Regression Machines”. *Advances in Neural Information Processing Systems.* ISSN: 1049-5258.

Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. 1999. *Hypertext Transfer Protocol – HTTP/1.1.* RFC. https://tools.ietf.org/html/rfc2616.

Finnish Meteorological Institute. https://en.ilmatieteenlaitos.fi/open-data.

Flight Plan Database. https://flightplandatabase.com/.

Freund, Y., and R. E. Schapire. 1997. "A Decision-Theoretic Generalization of On-Line Learning". *Journal of Computer and System Sciences* 55 (1): 119–139. ISSN: 0022-0000.

Friedman, Jerome H. 2001. "Greedy Function Approximation: A Gradient Boosting Machine". *The Annals of Statistics* 29 (5): 1189–1232. ISSN: 00905364. http://www.jstor.org/stable/2699986.

Glorot, X., A. Bordes, and Y. Bengio. 2011. "Deep Sparse Rectifier Neural Networks". In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistic (AISTATS).*

Haykin, Simon. 2009. *Neural Networks and Learning Machines: Third Edition.*

Hofmann, Martin. 2006. "Support Vector Machines — Kernels and the Kernel Trick An elaboration for the Hauptseminar " Reading Club : Support Vector Machines "".

International Civil Aviation Organization. 2005. *Rules of the air.*

Jupyter notebook. https://jupyter.org/.

Kaelbling, L., M. Littman, and A. Moore. 1996. "Reinforcement Learning: A Survey". *Journal of Artificial Intelligence Research* 4:237–285.

Kanevski, Mikhail, Alexei Pozdnoukhov, and Vadim Timonin. 2009. *Machine learning for spatial environmental data. Theory, applications and software.* 1st. 368. EPFL Press. ISBN: 978-2-940222-24-7.

Keras. https://keras.io/.

Klein, A., S. Falkner, S. Bartels, P. Hennig, and F. Hutter. 2016. "Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets". *CoRR.*

Kotsiantis, S. B. 2007. “Supervised Machine Learning: A Review of Classification Techniques”. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies,* 3–24. IOS Press.

Leege, Arjen de, Marinus van Paassen, and Max Mulder. 2013. “A Machine Learning Approach to Trajectory Prediction”. In *AIAA Guidance, Navigation, and Control (GNC) Conference.* doi:10.2514/6.2013-4782. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2013-4782. https://arc.aiaa.org/doi/abs/10.2514/6.2013-4782.

Liaw, Andy, and Matthew Wiener. 2002. “Classification and Regression by randomForest”. 2 (). ISSN: 1609-3631.

McCordyck, P. 2004. *Machines who think: A personal inquiry into the history and prospects of artificial intelligence.* 2nd. ISBN: 9780429258985.

Pitteway, M. L. V., and D. J. Watkinson. 1980. “Bresenham’s Algorithm with Grey Scale”. *Commun. ACM* (New York, NY, USA) 23, number 11 (): 625–626. ISSN: 0001-0782. doi:10.1145/359024.359027.

Polikar, Robi. 2012. “Ensemble Learning”. Chapter 1 in *Ensemble Machine Learning,* edited by Cha Zhang and Yungian Ma, 1–34. ISBN: 978-1-4419-9326-7.

Puolustusvoimat, Suomen. 2012. *Puolustusvoimien teknologiastrategia.* https://puolustusvoimat.fi/asiointi/aineistot/julkaisut-ja-esitteet.

Python. https://python.org/.

Renaud, Olivier, and Maria-Pia Victoria-Feser. 2010. “A robust coefficient of determination for regression”. *Journal of Statistical Planning and Inference* 140 (7): 1852–1862. ISSN: 0378-3758. doi:https://doi.org/10.1016/j.jspi.2010.01.008. http://www.sciencedirect.com/science/article/pii/S0378375810000194.

Russell, Stuart, and Peter Norvig. 2009. *Artificial Intelligence: A Modern Approach.* 3rd. Upper Saddle River, NJ, USA: Prentice Hall Press. ISBN: 0136042597, 9780136042594.

scikit-learn. https://scikit-learn.org/.

Simon, Herbert A., and Allen Newell. 1958. “Heuristic Problem Solving: The Next Advance in Operations Research”. *Operations Research* 6 (1): 1–10. ISSN: 0030364X, 15265463. http://www.jstor.org/stable/167397.

Smith, Steven W. 1997. *The Scientist and Engineer’s Guide to Digital Signal Processing.* www.DSPguide.com.

Steinberg, Dan. 2009. “CART: Classification and Regression Trees” (): 179–201.

Suykens, J.A.K., and J. Vandewalle. 1999. “Least Squares Support Vector Machine Classifiers”. *Neural Processing Letters* 9, number 3 (): 293–300. ISSN: 1573-773X. doi:10.1023/A:1018628609742.

Weisberg, Sanford. 2005. *Applied Linear Regression.* Third edition. ISBN: 0-471-66379-4.

Witten, I. H., and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Tecchniques.* Second Edition. ISBN: 0-12-088407-0.

Wright, W. E. 1990. “Parallelization of Bresenham’s line and circle algorithms”. *IEEE Computer Graphics and Applications* 10, number 5 (): 60–67. ISSN: 1558-1756. doi:10.1109/38.59038.

Zaki, Mohammed J., and Jr. Wagner Meira. 2014. *Data Mining and Analysis: Fundamental Concepts and Algorithms.* Cambridge University Press. ISBN: 9780521766333.

Zhou, Y., Jun Li, and L. Lamont. 2012. “Multilateration localization in the presence of anchor location uncertainties”. In *2012 IEEE Global Communications Conference (GLOBECOM),* 309–314. doi:10.1109/GLOCOM.2012.6503131.

Appendices

A Flight plan selection algorithm

Algorithm 3 Heuristic for finding the "closest" flight plan from flight plans F for an aircraft flight path P.

1. Let $D = \emptyset$ be the set of distances from flight path to the flight plans.
2. **For** each flight plan f_i in F **do**:
   3. Let $d_i = 0$ be the total distance for the flight plan f_i.
   4. **For** observation o_j in flight path P **do**:
      5. Let $e = \emptyset$ be the set of distance from observation o_j to the flight plan line segments.
      6. **For** Segment s_l in f_i **do**:
         7. Find the distance t between the observation o_j and the line segment s_l with the algorithm 4. Take logarithm of the distance:

$$e_l = \log(1+t) \tag{6.1}$$

            Add the distance to the list of segment distances $e = e \cup \{e_l\}$.
      8. Add the shortest segment-observation distance to the total flight path-flight plan distance $d_i = d_i + \min(e)$.
   9. Add the flight plan f_i's distance to the set of distances $D = D \cup \{d_i\}$.
10. **Return** the flight plan f_i that corresponds to shortest distance $\min_{d_i \in D} d_i$.

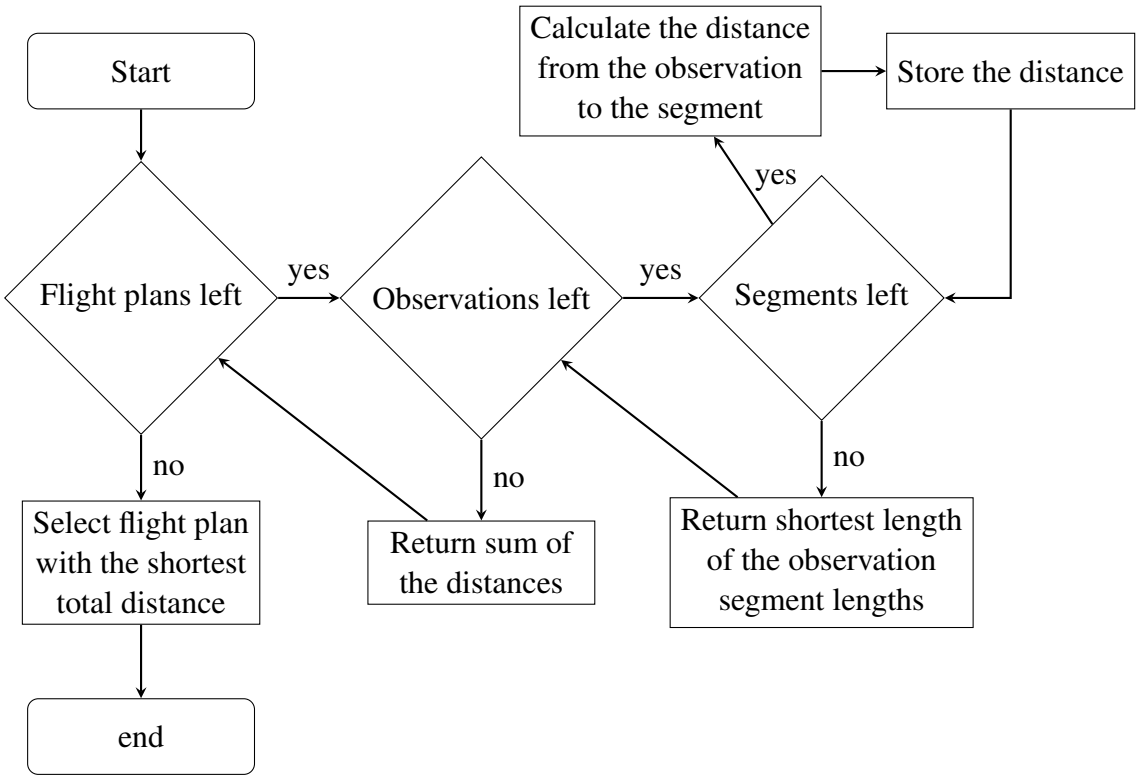

Figure 11. Flowchart of the flight plan selection algorithm.

Algorithm 4 Algorithm for shortest distance from point p to a line segment defined by two points v and w.

1. Project p to the line defined by the two points v and w:

$$t = \langle p - v, w - b \rangle \tag{6.2}$$

2. Project to the line segment capped by the points:

$$t' = \min(\max(0, t), 1) \tag{6.3}$$

3. **Return** the length of the difference vector of the original point p and the projected point t':

$$\|p - v + t'(w - v)\| \tag{6.4}$$

where $\|\cdot\|$ is the L2-norm.