CHALMERS
UNIVERSITY OF TECHNOLOGY

Calculation of estimated time of arrival using artificial intelligence

A comparative study of machine learning algorithms and their ability to determine estimated time of arrival

Master's thesis in Engineering Mathematics and Computational Science

KONSTANTINOS KONSTANTINOU

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019

Calculation of estimated time of arrival using artificial intelligence

A comparative study of machine learning algorithms and their ability to determine estimated time of arrival

KONSTANTINOS KONSTANTINOU

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
Gothenburg, Sweden 2019

Calculation of estimated time of arrival using artificial intelligence

A comparative study of machine learning algorithms and their ability to determine estimated time of arrival

KONSTANTINOS KONSTANTINOU

Supervisor: Mohammad Manjurul Islam, Volvo Group Trucks Technology, Sweden
Examiner: Stig Larsson, Chalmers University of Technology, Dept. of Mathematical Sciences, Sweden

Master's Thesis 2019
Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2019

Calculation of estimated time of arrival using artificial intelligence
A comparative study of machine learning algorithms and their ability to determine estimated time of arrival
KONSTANTINOS KONSTANTINOU
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Providing solutions for designing an optimal planning scheme is extremely important in transportation industry. A potential solution requires the development of models that are able to accurately predict the remaining travel time of trucks operating in a transport mission. Unfortunately this is not a trivial task as a transport mission is influenced by a variety of stochastic and impossible to predict factors. This study develops a variety of machine learning approaches and benchmark their ability to predict arrival times. In particular Support Vector Regression, Artificial Neural Networks, Gradient Boosting, Random Forest and Stacked Generalization models were developed for the aforesaid task. The proposed models are trained and evaluated using GPS and weather data for a transport mission between Malmö and Göteborg. The main objectives of the study are finding the variables that influence the total travel time of a vehicle and are optimal to be used as inputs to the prediction models, comparing the performance of different machine learning approaches and identifying the optimal approach among the proposed models. Study results verified that machine learning approaches have the ability to predict the arrival times of trucks. Even though all methods outperformed a historic data based model, results showed that the Random Forest and Stacked Generalization methods outperformed the other machine learning models in terms of Root Mean Square Error and Mean Absolute Percentage Error. In addition it was found that utilizing appropriate features as inputs to the prediction models dramatically increased the performance of the algorithms.

Keywords: Support Vector Regression, Feed Forward Neural Networks, Recurrent Neural Networks, Gradient Boosting, Random Forest, Stacked Generalization, Artificial Intelligence, Road Transport, Estimated time of arrival(ETA)

Acknowledgements

I would like to express my sincere gratitude to my industrial thesis supervisor Mohammad Manjurul Islam for suggesting the research topic and providing me with the appropriate hardware and software as well as for his continuous guidance throughout the project. I also want to thank Björn Mårdberg for providing me with the Volvo FH-180 log data and helping me with the technical details regarding the data.

Additionally, I would like to express my special appreciation and thanks to my academic supervisor and examiner, Prof. Stig Larsson for his academic guidance, advice and constructive feedback.

Last but not least I would like to thank my family for their encouragement and constant support. Without them this research work would not have been possible.

Konstantinos Konstantinou, Gothenburg, June 2019

Contents

List of Figures

List of Tables

1
Introduction

In this chapter, the background of the main problem motivating this thesis is presented and discussed. The purpose, scope, aims and the research questions the thesis trying to answer are also analyzed and clearly stated. In the end of this chapter, the overall structure of the thesis is explicitly described.

1.1 Background

Transportation is a vital part of the development of modern civilizations. Through transport, trade is being developed and effective distribution of goods is possible in countries, contributing to the improvement of the quality of life of citizens. Even though this can be achieved by different modes of transport such as road, rail, aviation and maritime transports, road transport is the main means of transporting passengers and goods in the modern world. As far as freight transport is concerned, road transport continues to dominate with continued growth. According to studies [12] almost half of the freight tonnage transported in Europe in 2010 was by road.

This thesis considers a specific transport mission operated by Volvo FH-180 trucks between Göteborg and Malmö. The aim of the study is to propose different machine learning models to predict the remaining travel time of a truck in the mission and evaluate their performance. In general, the prediction of the arrival time of trucks is not a trivial task since it is usually influenced by many stochastic factors such as weather and traffic congestion.

An optimal planning scheme requires the design and development of prediction models for the estimated time of arrival of the operating vehicles. A travel time estimation model is a fundamental element to decrease wastes related to idle times, for instance a truck arriving early and forced to be idle for a significant amount of time or arriving late and force other operational machines to be idle.

Unfortunately due to the complex non-linear relationship between various factors and the total travel time of a truck, no explicit formula is known that can give perfect predictions. This leads to a need of approximation models that can approximate this complex non-linear function. Moreover it was concluded in many studies that artificial intelligence models can predict the outcome of an unknown complex non-linear function with minimal errors [17]. In the last years artificial intelligence models managed to accomplish great achievements, and started to outperform humans in many tasks [28] [15].

1.2 Purpose of the study

A well-designed prediction model for the total travel time of trucks in a transport mission is an essential ingredient to make decisions that can optimize planning. This can be achieved by utilizing the proposed prediction models to make better planning in order to increase productivity.

The purpose of this thesis is to propose and evaluate the ability of machine learning algorithms to predict the arrival time of trucks. Furthermore this thesis studies which factors influence the total travel time of the vehicles in a mission. The research questions that this thesis tries to answer are the following:

1. What are the variables that can significantly influence the total travel time of a vehicle in a mission and are optimal to be used as inputs to minimize the errors of the prediction model?

2. What proposed Artificial Intelligence prediction model accomplished the best performance when a benchmark comparison of the ability of the models to determine the estimated time of arrival of a truck was performed?

1.3 Thesis structure

The thesis is structured into the following six chapters:

- **Introduction**
- **Literature Review**
- **Theory**
- **Methods**
- **Results**
- **Conclusions**

In the **Introduction** chapter the background of the main problem that motivated the study as well as the main goals and objectives of the study are explicitly analyzed, stated and described.

In the **Literature Review** chapter we present an extensive literature review that was performed with the research questions of the thesis in mind. In this chapter several results and models from studies related to predicting travel time of vehicle throughout the years with a specific mission are mentioned. From the literature review we noticed that even though there were many studies related to our goals, to design a reliable arrival time prediction model, the proposed models were mostly related to buses. Hence there is a need to study if the technologies that are widely used to deal with similar problems, can also be utilised to make predictions for the specific problem this thesis considers.

In the **Theory** chapter the theoretical background of the proposed models as well as the mathematical principles of the algorithms are analyzed and discussed. Furthermore frequently used techniques for improving the performance of the prediction models are analyzed and explained in detail.

In the **Implementation** chapter a detailed description of how the data were collected and processed as well as the libraries and software utilised to implement the models are presented.

In the **Results** chapter the experimental results of the study are described and analyzed. In the end of the chapter the outcomes of the thesis are compared with results of other studies in the literature.

In the **Conclusions** chapter the research questions this thesis considers are answered. In addition the thesis limitations as well as future projects proposals are stated.

2

Literature review

In the previous chapter the needs that motivated this thesis were analyzed and discussed. In order to achieve our goals, an extensive literature review was performed to determine what Artificial Intelligence models are more suitable to predict the remaining travel time of vehicles operating in a mission. Furthermore one of the main goals of the literature review was to discover what variables give more accurate results when used as inputs to a specific model. In this chapter a variety of proposed models based on past studies and their performance is presented. Moreover the input variables to the prediction models and also the technology used to collect the corresponding data are analyzed and discussed. In the end of this chapter a brief summary of our findings is presented.

2.1 Determination of estimated time of arrival

Researchers found that Artificial Network Networks (ANN) had a better performance than Historical data based Models and Regression Models in terms of the predicted time of arrival accuracy [17]. They believed that ANN were able to capture the complex non-linear relationship between travel time and the independent variables. This study used data such as arrival time, dwell time and schedule adherence at each stop collected by a Differential Global Positioning System(DGPS) receiver in Houston, Texas as the input variables to each proposed prediction model.

A dynamic algorithm based on the Kalman filtering technique [18] was developed to estimate buses arrival times using Automatic passenger counter (APC) data. The APC data consisted of many variables such as schedule arrival time, transit day and time of the day, latitude, longitude, stop distance, trip status(start/end), time point ID, dwell time and inter-stop travel time. The dynamic algorithm consisted of two parts, in the first part an ANN was predicting the bus travel time between time points and in the second part a dynamic Kalman-based algorithm was using bus location information to adjust the prediction for the arrival time of the bus. They claimed that the algorithm can be utilised to provide real-time bus arrival time predictions for each time point along the route [8].

Other studies proposed Support Vector Machine (SVM) models [10] such as Support Vector Regression (SVR) [11] to make predictions for the arrival time of buses. Their model considered current segment, travel time of current segment and the largest travel time of next segment as inputs to their prediction models. The proposed model outperformed an ANN model in terms of Root Mean Square Error(RMSE) in seconds. They stated that SVM could be trained thought a linear

optimization process and unlike ANN, this model is not affected by the overfitting problem [5].

More recent studies utilised a Support Vector Machine with Genetic Algorithm (GA-SVM) model to determine the estimate time of arrival. The GA-SVM algorithm used a Genetic Algorithm (GA) to find optimal hyper-parameters for a Support Vector Machine (SVM) model. Their input variables to the SVM consisted of weather related data, the character of the time period, the average speed, the rate of road usage and the length of the road. Results showed that GA-SVM model was superior to traditional Artificial Neural Networks(ANN) and SVM models in terms of prediction accuracy [35].

In 2018 Wang et al., [32] proposed an end to end Deep learning framework for Travel Time Estimation (DeepTTE) to estimate travel time of the whole path directly. According to them estimating the travel times of individual segments to calculate the travel time of the whole path produces inaccurate predictions. That is because this approach does not consider road intersections and traffic lights, hence local errors may accumulate. The model consisted of three components, the Spatio-Temporal Learning Component, the Attribute Component and the Multi-task Learning Component. For the Spatio-Temporal Learning Components, DeepTTE used Convolutional Neural Networks(CNN) [20] with Exponential Linear Unit (ELU) activation [9] to capture the spatial dependencies in the GPS sequence and recurrent layers to capture temporal dependencies among the local paths. The Attribute Component was used to process external factors and information of the path, for instance weather, driver, time and distance related data, and its output was used as input to the other Components. The Multi-task Learning Component combined the other components to give predictions for the travel time using residual fully connected layers with Rectified Linear Unit (ReLU) activation [14]. Experiments suggested that DeepTTE significantly outperformed other models.

Mehmet and Metin in 2013 [3] made a review for different computational models for arrival time prediction. According to them historical data models are only reliable when the traffic patterns in the area are stable since their accuracy relies on the similarity between real time and historic traffic patterns. In the case of regression models several studies showed that they are outperformed by other models. The study suggest that regression models showed poor performance since the variables are highly inter-correlated. Moreover they discussed the performance of Kalman filtering models and Machine learning models like ANN and SVM. They claimed that the Kalman filtering approach and Machine learning models can be used to give reliable results. They stated that even though no algorithm managed to produce robust results there is an increasing trend between researchers to utilise hybrid algorithms to improve a model's accuracy.

A study to predict the cycle time of trucks in an open-pit mining was performed by Chanda and Gardiner in 2010 [7]. Artificial Neural Networks (ANN) and Multiple Linear regression (MLR) prediction models were compared with the most commonly used model by that time, a simulation using TALPAC software [1] based on the Monte-Carlo technique. Their results showed that ANN and regression models were superior to the computer simulations techniques.

Xiaoyo et al., in 2018 [29], proposed three machine learning models to predict

real-time link travel time of open-pit trucks in Fushun West Open-pit Mine in China. More specifically k-Nearest Neighbour (kNN), Support Vector Machine (SVM) and Random Forest (RF) algorithms were used to predict travel time for every link road. The link roads were split into two categories, the fixed and the temporary roads. The road and truck related data were collected with open-pit automated truck dispatching systems (OPATDS) while the weather data were collected from the China Meteorological Administration (CMA). Further two different approaches were compared, namely the LTTP and RTTP approach. LTTP approach considered many models for every possible link in the route and summed up the best predictions of every route while the RTTP approach used a model to predict arrival time for the whole route. In terms of Mean Absolute Percentage Error (MAPE) LTTP approach outperformed the RTTP approach. Moreover their results indicated that RF and SVM methods outperformed the kNN approach. Also experiments showed that using weather related data as inputs into their models resulted in a dramatic increase in the prediction accuracy.

2.2 Summary of literature review

Many studies concluded that machine learning models outperform other approaches in terms of prediction accuracy. Most used methods for that purpose appear to be Artificial Neural Networks, Support Vector Regression and Random Forest. Through the literature review we identified a huge research gap related to models that can predict the arrival time of vehicles operating in a long mission. In particular most of the studies are focused in estimating the arrival time of urban buses. In those mission factors such as driver breaks are not existent. In this study we develop prediction models for the estimated time of arrival of a long transport mission with driver breaks.

3

Theory

In this chapter the theoretical background of the proposed machine learning prediction models is explicitly analysed and described. In the first part of the chapter the theory behind artificial neural networks and techniques that are usually used to optimize their performance are presented. In the following sections an overview of the theoretical background of machine learning algorithms such as the support vector regression and ensemble models such as random forest, gradient boosting trees and stacked generalization are briefly discussed. The purpose of this chapter is to provide the reader with enough knowledge to be able to understand how the proposed models produce their predictions and what mathematical principles are the basis of every model.

3.1 Artificial neural networks

Artificial Neural Networks (ANN) are computational networks consisting of artificial neurons and are inspired by the biological neural networks that constitute animal brains. Every artificial neuron can receive a signal, process it and sent a signal to other nearby artificial neurons. In the end the network produce an output value based on its inputs. Many studies found out that they can be used to predict the output of a complex non-linear function [17]. There are many different types of neural networks such as recurrent and feed forward neural networks. This study utilises both feed forward and recurrent neural networks to get predictions for the total travel time of trucks.

3.1.1 Feed forward networks

Feed-forward networks are artificial networks where neurons belong to a layer of neurons and signals are only allowed to be transferred forward from the input layer in the left side to the output layer in the right side. In other words no signal can be transferred from layer i to layer j, where $j \leq i$. Moreover the network is a fully connected network, meaning that every neuron is connected to every neuron to next layer. In addition to that no neuron is allowed to skip a layer, for instance a neuron in layer i is unable to sent a signal to another neuron in layer k for $k \geq i + 2$. An example of a feed-forward neural network with two hidden layers is illustrated in figure 3.1.

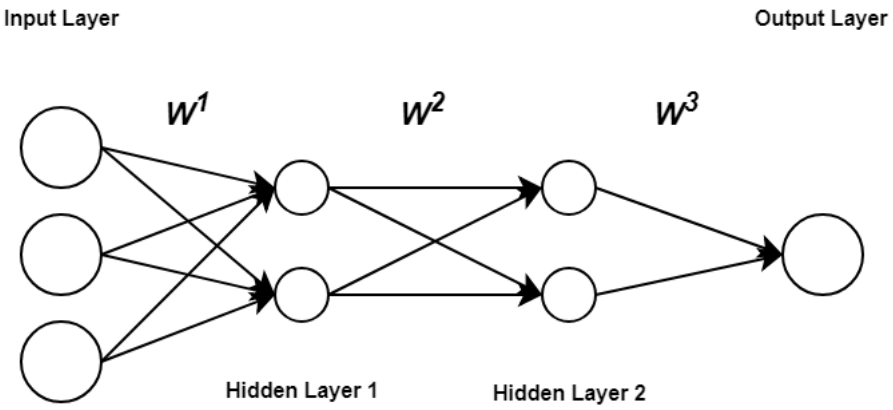


Figure 3.1: A multi-layer perceptron with three inputs, two hidden layers with two neurons and one output.

3.1.2 Training a network

After designing the architecture of a network, it is time to train the network to give accurate predictions. The training procedure is an iterative process consisting of two parts, the feed forward propagation and the back propagation algorithm. After obtaining a pair of inputs and outputs, the weight matrices and biases are randomly initialized and an error function is chosen. In the first part the inputs are fed forward to produce a prediction, which is expected to be random at first iterations, and in the second part the weights and biases are updated by utilizing an optimization algorithm on the error function. In other words the network initially makes a lot of mistakes and it begins to learn by capitalizing on its mistakes. Those parts as well as the mathematical principles of the algorithms will be discussed in detail in the next paragraphs.

3.1.2.1 Forward propagation

As it was previously stated forward propagation is the first part of the learning procedure of a neural network. Initially an activation function for every layer, except the input layer, is chosen. Some of the commonly used activation functions are

$$\text{The Sigmoid function :} \quad g(x) = \frac{e^x}{e^x + 1} \tag{3.1}$$

$$\text{The Hyperbolic Tangent :} \quad g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.2}$$

$$\text{The Rectified Linear Units :} \quad g(x) = \max(0, x) \tag{3.3}$$

The shape of those activation functions is shown in fig 3.2.

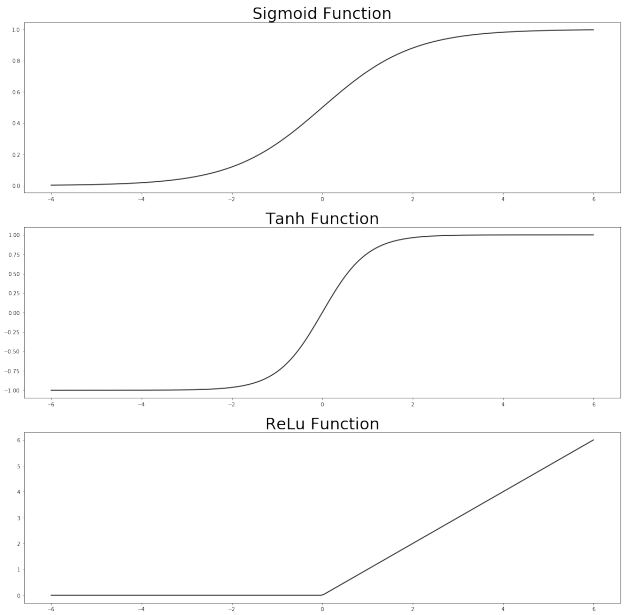


Figure 3.2: The shapes of the Sigmoid, tanh and ReLu activation functions.

Lets consider a feed forward neural network with N hidden layers. We define:

- g_j = activation function for layer j $\in \{1, ..., N+1\}$.
- V^j = the states of neurons in layer j $\in \{0, ..., N+1\}$.
- W^j = The weight matrix connecting layer j-1 with layer j $\in \{1, ..., N+1\}$.
- b^j = The biases in layer j $\in \{0, ..., N+1\}$.

Initially the weight matrices as well as the biases are randomly chosen. Then the inputs $V^0 \in \mathbb{R}^k$ are fed into the network to produce the outputs $V^{N+1} \in \mathbb{R}^m$. This is achieved by the following computations

$$V^j = g^j(\ W^j V^{j-1} + b^j\ ) \quad \text{for j} \in \{1, ..., N+1\}. \tag{3.4}$$

After the computations in equation (3.4) outputs were produced by utilising the data as inputs. However since the weight matrices and biases were randomly chosen the model's predictions are expected to be wrong. The idea of the second part of the algorithm is to apply optimization techniques on an error function to update weights and biases in order to produce more accurate results.

3.1.2.2 Gradient descent

One optimization algorithm frequently used for minimizing a neural network error function is the gradient descent algorithm. Lets assume that H is the function we want to minimize, in this study we consider the error function H to be the mean square error between the actual values of the training set and the predicted values. The Gradient Descent optimization algorithm utilises the fact that the

negative gradient direction of H is a descent direction. Then in order to find a local minimum the algorithm takes a step proportional to the negative gradient of the function evaluated at the current point. However since we do not want our algorithm to get stuck in a local minimum but to converge to a global minimum stochastic Gradient Descent is usually used to add noise to the algorithm.

3.1.2.3 Chain rule

Chain rule is a fundamental mathematical formula used to compute derivatives of the composition of two or more functions. According to [33] the formulation of the Chain rule is as follows:

- Let $w = f(x, y, w, ..., v)$ be a differentiable function of a finite set of variables $\mathcal{A} = \{x, y, w..., v\}$
- Let $x, y, w..., v$ be differentiable functions of another finite set of variables $\mathcal{B} = \{p, q, ..., t\}$.
- Then w is a differentiable function of the variables $p, q, ..., t \in \mathcal{B}$ and the partial derivatives of w with respect to a variable $k \in \mathcal{B}$ is given by

$$\frac{\partial w}{\partial k} = \frac{\partial w}{\partial x}\frac{\partial x}{\partial k} + \frac{\partial w}{\partial y}\frac{\partial y}{\partial k} + ... + \frac{\partial w}{\partial v}\frac{\partial v}{\partial k} \tag{3.5}$$

3.1.2.4 Back propagation

The second part of learning procedure is the back propagation algorithm. In this part the weight matrices and the biases are updated by applying an optimization algorithm in an error function. For the purpose of the thesis we consider the mean square error function H and we choose gradient descent as our optimization algorithm. Lets assume that (x^μ, y^μ) for $\mu \in \{1, ..., k\}$ are pairs of input and target vectors and O^μ is the output vector of the feed forward propagation algorithm with n hidden layers. Then the error function is defined as

$$H = \frac{1}{2}\sum_{i,\mu}(y_i^\mu - O_i^\mu)^2 \tag{3.6}$$

Applying the gradient descent optimization algorithm for the weights and learning rate η we get the following weight updates and biases updates

$$W_{m,n}^l = W_{m,n}^l - \eta\frac{\partial H}{\partial W_{m,n}^l}, \qquad \forall\ l \in \{1, ..., N+1\} \tag{3.7}$$

$$b_j^l = b_j^l - \eta\frac{\partial H}{\partial b_j^l}, \qquad \forall\ l \in \{1, ..., N+1\} \tag{3.8}$$

Then by applying the chain rule (see Appendix A) we calculate the partial derivatives

$$\frac{\partial H}{\partial W^l_{m,n}} = -\sum_{\mu} \Delta^{\mu,l}_m \; V^{\mu}_n \tag{3.9}$$

$$\frac{\partial H}{\partial b^l_j} = -\sum_{\mu} \Delta^{\mu,l}_j \tag{3.10}$$

where the errors for neuron m in the output layer are given by

$$\Delta^{\mu,N+1}_m = g'(\sum_n W^{N+1}_{m,n} \; V^N_n \; + \; b^{N+1}_m \;)(y^{\mu}_m - O^{\mu}_m) \tag{3.11}$$

and the errors for neuron m in the layer l are given by

$$\Delta^{\mu,l}_n = \sum_m \Delta^{\mu,l+1}_m \; W^{l+1}_{m,n} \; g'(\sum_k W_{j,k} V^l_k + b^{l-1}_j) \qquad \text{for } l = 1, ..., n \tag{3.12}$$

3.1.3 Adam optimizer

In this study we utilise the Adam optimizer as described in [19] to perform optimization on the selected error function. Adam optimizer is an efficient algorithm that combines two optimization algorithms, the gradient descent with momentum and Root Mean Square Propagation (RMSprop) algorithms. Experienced neural network engineers often suggest using Adam optimizer since it is an algorithm that requires minimal hyper parameter tuning and lower memory usage than other optimization methods.

3.1.4 Initialization of weights and biases

In the remainder of this section, techniques often used to increase the model performance are discussed. Previously we mentioned that our initial weights and biases are randomly initialized. According to [30] many deep and recurrent neural networks failed to learn due to poor initialization schemes. Therefore a good initialization strategy is essential to increase a prediction model performance. A common initialization strategy is to set biases to zero and weight matrices to normally distributed numbers with zero mean and variance proportional to $\frac{1}{N}$, where N corresponds to the number of incoming connections to a neuron. This initialization scheme guarantees that weights are small initially and speeds up the convergence of the optimization algorithm [23].

3.1.5 Normalizing data

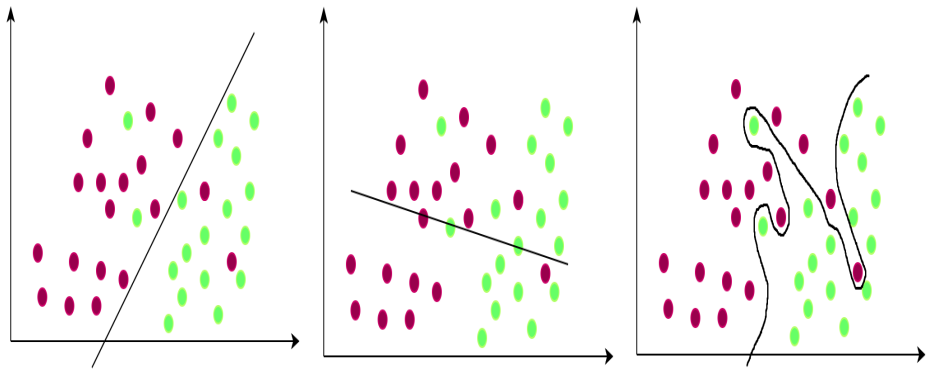
A common technique that speeds up the optimization process of the error function is to normalize the data before using them as inputs into the prediction model. Normalization of the data means transforming the data to data with zero mean and

unit variance. The normalization of the data transforms an ill-posed problem to a well-posed problem and therefore speeds up the convergence of the optimization algorithm. In this study the proposed neural network normalizes the train data and then the normalized train data are used as inputs to the prediction model. Furthermore for the evaluation of the model the test data also need to be standardized.

3.1.6 Overfitting and underfitting

The goal of the described algorithm is to give reliable predictions for unseen random data and learn to generalise the knowledge it learnt from the training data. Unfortunately usually networks start to "memorise" the patterns of the training data and are not able to generalize. This phenomenon is known as overfitting and is often observed when the architecture of the neural networks includes many neurons as well as when the network is "over-trained". This happens when the algorithm fits the training data too well and it starts capturing the noise of the data. Several techniques that reduce this phenomenon such as early stopping and regularization techniques are discussed and explained in the following sections.

In contrast networks are said to underfit when they are unable to capture trends in the data. Underfitting is usually observed when the prediction model is to simple, consists only of few neurons. Both overfitting and underfitting are problems that an optimal prediction model should aim to avoid in order to produce optimal predictions. Those phenomena and their negative effects in a specific classification problem are illustrated in figure 3.3.



(a) Optimal Fit (b) Underfitting (c) Overfitting

Figure 3.3: Examples of overfitting and underfitting. The prediction model can generalize and give reliable predictions (Left). The prediction model fails to capture the trend in the data (Middle). The prediction model captures all the noise of the train set(Right)

3.1.6.1 Early stopping

One of the previously mentioned techniques is early stopping. The idea behind this technique is simple but very effective. Before training the data-set at hand is split into two different sets, the training set and the validation set. The training

set consist of the data that would be used to train the model while the validation set consists of the data that would be used for validation instead of training. The early stopping stopping technique then keeps track of the error for both the training and the validation sets and aborts the training procedure when the error of the validation set becomes stable or increases over a number of iterations. The early stopping trick is presented in figure 3.4.

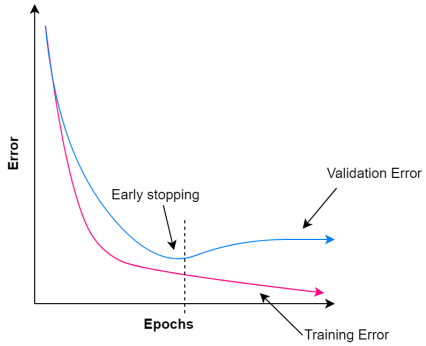


Figure 3.4: The early stopping technique. The red line represents the training error and the blue line the validation error. The training is stopped when the validation error, blue line, increases over a number of iterations.

3.1.6.2 Weight decay

In this part the L_1 and L_2 regularization techniques and how those can be used to prevent overfitting are analyzed and discussed. The idea of regularization is to add a regularization term in the error function H.
In the L_1 regularization scheme the new error function becomes

$$H' = H + \frac{\gamma}{2} \sum_{i,j} |W_{i,j}| \tag{3.13}$$

In the L_2 regularization scheme the new error function becomes

$$H' = H + \frac{\gamma}{2} \sum_{i,j} W_{i,j}^2 \tag{3.14}$$

where $\gamma \in \mathbb{R}_+$. Then the update rule for the weights are

$$W_{k,l} = W_{k,l} - \eta \frac{\partial H}{\partial W_{k,l}} - \epsilon W_{k,l} \tag{3.15}$$

$$W_{k,l} = W_{k,l} - \eta \frac{\partial H}{\partial W_{k,l}} - \epsilon \, \text{sign}(W_{k,l}) \tag{3.16}$$

for the L_2 and L_1 regularization scheme respectively where $\epsilon = \gamma\eta$.

This approach prevents overfitting in the two following ways. First, when weight decay is applied some of the effects of static noise are suppressed. Second, the model chooses the smallest weight vector that solves the learning problem, hence any irrelevant components of the weight vector are also suppressed [21]. Therefore an increase in the ability of the network to generalize is observed.

3.2 Recurrent neural networks

Unlike the Feed Forward Networks (see 3.1) where information is fed forward through the network, Recurrent Neural Networks are networks with an architecture that allows more complex information flows [26]. For instance the information can be looped though the network as seen in figure 3.5. This modification makes RNN suitable models to forecast time series data since the network not only uses information from one observation but also keeps into the network memory information of the previous time-steps. As a result those networks are an optimal choice for translation tasks, speech recognition tasks and are widely used in applications where inputs are given as a sequence of data. However since each neuron can give feedback to itself the back propagation algorithm discussed in 3.1.2.4 needs to be modified.

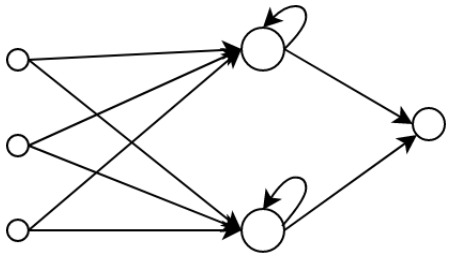


Figure 3.5: Example of a recurrent neural network structure.

3.2.1 Back propagation through time

In order to train a RNN , the network is expanded in time as seen in figure 3.6. Then the information are fed though the network as described in equations (3.17) and (3.18). In particular the hidden state of the neuron at time t, h_t is given by

$$h_t = g_1(Ux_t + Wh_{t-1} + b) \tag{3.17}$$

and the output at time t, O_t is then computed by

$$O_t = g_2(Vh_t + B) \tag{3.18}$$

where g_i are activation functions, for instance the hyperbolic tangent or the sigmoid function, and U, V, W are weight matrices and b, B are vectors containing the biases. Then the weights are updated using the back propagation algorithm discussed in 3.1.2.4. We consider again the mean square error function

$$H = \frac{1}{2}\sum_{i=1}^{T}(Y_i - O_i)^2$$

where Y_i are the actual response variable at the time step i. Then computing the gradient of error H with respect to the weights W using the Chain rule we obtain.

$$\frac{\partial H}{\partial W} = \frac{\partial H}{\partial O_k}\frac{\partial O_k}{\partial h_k}(\prod_{i=2}^{k}\frac{\partial h_i}{\partial h_{i-1}})\frac{\partial h_1}{\partial W} \tag{3.19}$$

Inserting equation (3.17) to equation (3.19) and calculating the product of derivatives $\frac{\partial h_i}{\partial h_{i-1}}$ we observe that the network learning rate tends to decay exponentially [4]. This phenomenon is also known as the Vanishing Gradient Problem. As a result RNN are computationally expensive to train and hence a lot of computation power is required.

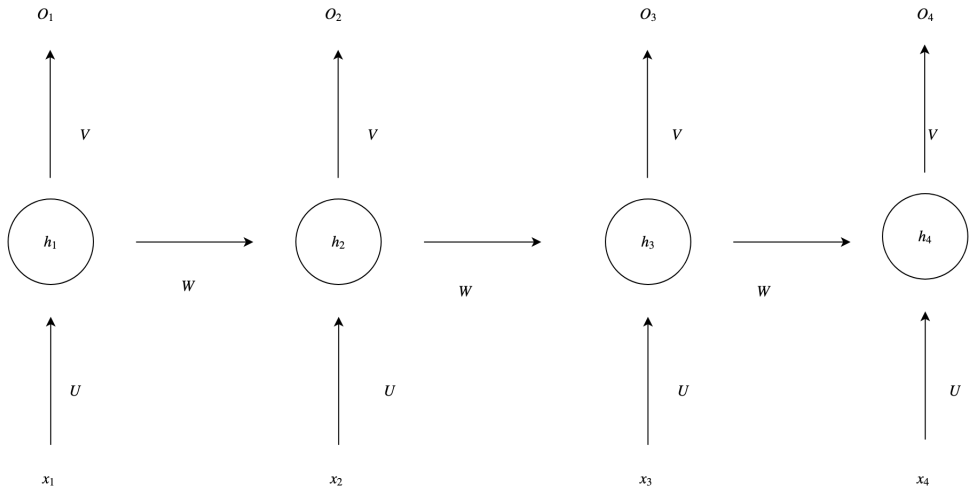


Figure 3.6: Plot demonstrating how the network is expanded in time and how information is then fed through the network.

3.2.2 LSTM

In the previous paragraph it was mentioned that RNN suffer from the Vanishing Gradient problem and are unable to handle dependencies over large time periods. Long Short-Term Memory [16] cells are RNN architectures that can deal with the vanishing gradient problem. The idea is to replace every neuron in the network with a LSTM cell. An LSTM cell consists of three gates, the input gate, the output gate and the forget gate. The input gate determines how the new information flows into the LSTM cell, the forget gate decides what needs to kept in the cell memory and the output gate is used to compute the output of the cell. The inputs to an LSTM cell are the input data at time t, x_t and the hidden state of the previous LSTM cell h_{t-1}. The outputs f_t, i_t, o_t of the forget, input and forget gates at time t respectively are given by.

$$f_t = \sigma(W_\mathrm{f} x_t + U_\mathrm{f} h_{t-1} + b_\mathrm{f}) \tag{3.20}$$

$$i_t = \sigma(W_\mathrm{i} x_t + U_\mathrm{i} h_{t-1} + b_\mathrm{i}) \tag{3.21}$$

$$o_t = \sigma(W_\mathrm{o} x_t + U_\mathrm{o} h_{t-1} + b_\mathrm{o}) \tag{3.22}$$

The hidden state of the cell h_t and the state of the cell c_t at time t are then given by

$$c_t = c_{t-1} f_t + i_t \tanh\left(W_\mathrm{c} x_t + U_\mathrm{c} h_{t-1} + b_\mathrm{c}\right) \tag{3.23}$$

$$h_t = o_t \tanh\left(c_t\right) \tag{3.24}$$

where W_j, U_j are matrices and b_j are biases for indices $j \in \mathcal{A} = \{f, i, o, c\}$. Using this architecture the derivative that caused the problem in the previous paragraph is approximately

$$\frac{\partial c_t}{\partial c_{t-1}} \approx f_t \approx 1 \tag{3.25}$$

This implies that the gradients do not vanish and the network can learn dependencies over large time periods. These designs are also known as Constant Error Carousels (CEC). In this study an RNN model utilising LSTM cells was developed to estimate the arrival time of trucks between Malmö and Göteborg. The architecture of an LSTM cell is visible in figure 3.7.

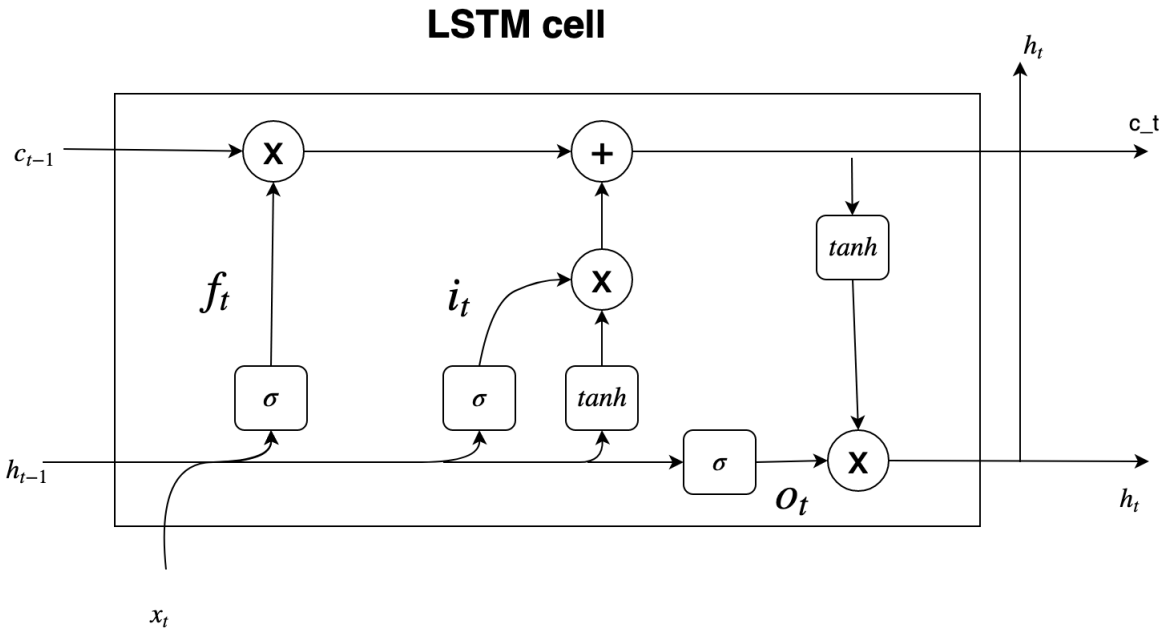


Figure 3.7: The Long Short-Term Memory unit Architecture. The flow of the information from all three gates is presented. The circular shapes represent element-wise multiplication and addition. The rectangular shapes represent application of sigmoid and hyperbolic tangent functions to an affine transformation of their inputs.

3.3 Support vector machines

Support Vector Machines (SVM) [10] are machine learning algorithms that can be used for classification as well as for regression. This section describes how the SVM algorithm works for classification problems. The basic idea of the SVM algorithm is to find the optimal hyperplane, often called maximal margin hyperplane, that maximizes the distance from the nearest data points on each side. A classification problem is said to be linearly separable if it is possible to find a hyperplane that separates the data into different classes. Moreover when the problem is linearly separable then two parallel hyperplanes that separate the classes and the distance between them is maximal can be selected. The region between those hyperplanes is called margin and points on the boundary of the margin are called support vectors. When the data are standardized and labeled as 1 and -1, for the first and second class respectively, those hyperplanes as well as the width of the margin can be described with the following equations.

$$\text{First Parallel Hyperplane} : Wx - b = 1 \tag{3.26}$$

$$\text{Second Parallel Hyperplane} : Wx - b = -1 \tag{3.27}$$

$$\text{Maximal Margin Hyperplane} : Wx - b = 0 \tag{3.28}$$

$$\text{Margin Width} : \frac{2}{|| W ||} \tag{3.29}$$

After receiving pairs of inputs and labels (x_j, y_j) $j \in \mathcal{A} = \{1, ..., M\}$ from the dataset the SVM algorithm is equivalent to solving the following linear optimization problem.

$$\min \frac{1}{2} || W ||^2 \tag{3.30}$$

Subject to

$$y_i (Wx_i - b) \geq 1, \qquad \forall i \in \mathcal{A} \tag{3.31}$$

The optimization problem of the SVM algorithm has a quadratic objective function and linear constraints, hence it is a Quadratic Programming (QP) problem. Furthermore, many algorithms exist that can efficiently solve this type of optimization problem. Figure 3.8 illustrates how SVM algorithm works for classification problems. The maximal margin hyperplane is represented by the red line and the support vectors, the points in the dashed line are also presented.

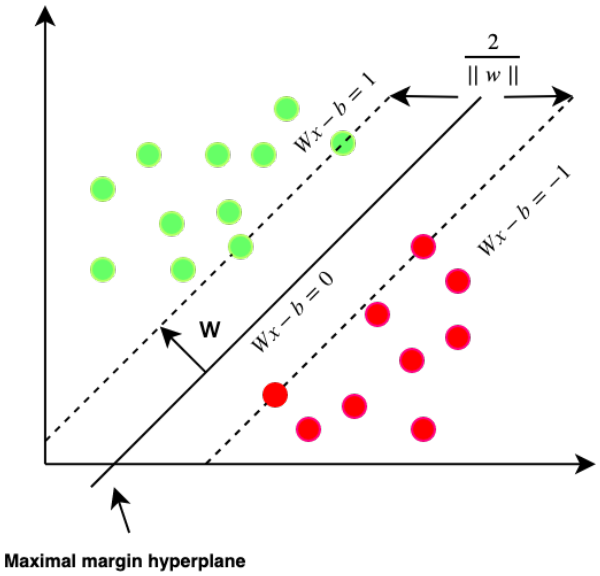


Figure 3.8: The Support Vector Machine algorithm. The maximal margin hyperplane as well as the support vectors, points in the dashed line, are displayed.

3.3.1 Kernel trick

In the previous paragraph we assumed that the data are linearly separable, in other words we assumed that the maximal margin hyperplane exists. Unfortunately most of the times this is not the case, therefore a technique known as the kernel trick is often utilised. The idea that motivates the kernel technique is that is easier to separate variables when you map them in a higher dimensional space (Cover's Theorem). However it can be quite hard and impractical to define such mappings and computationally expensive to implement. Therefore in order to overcome this difficulty the kernel functions are introduced. Kernel functions are functions that can do operations in higher dimensional spaces by computing the inner products of data in the original space without the need of defining and computing complex mappings. Commonly used kernels are presented in table 3.1.

Kernel Function		
Linear Kernel	$\mathcal{K}(x_i, x_j) = \mathrm{x}_i \cdot x_j$	
Radial Basis Function Kernel	$\mathcal{K}(x_i, x_j) = \mathrm{e}^{-\gamma\|\|x_i - x_j\|\|^2}$	$\gamma > 0$
Polynomial Kernel	$\mathcal{K}(x_i, x_j) = (x_i \cdot x_j + 1)^d$	$d \in \mathbb{N}$

Table 3.1: Popular kernel functions

Then the SVM algorithm is used to separate the variables produced by the kernel

mapping. The following section describes how a support vector machine algorithm can be modified in order to confront regression problems.

3.3.2 Support vector regression

This paragraph is focused on the Support Vector Regression (SVR) algorithm which is inspired by the SVM algorithm and is used to confront regression problems. The idea of SVR algorithm is to approximate the unknown real valued function $f(x)$ with a higher dimensional hyperplane by using the kernel function $\phi(.)$. In other words we assume that $f(x)$ can be expressed in the following way.

$$f(x) = w \cdot \phi(x) + b \tag{3.32}$$

However since we deal with a regression problem instead of a classification problem we introduce an ϵ-tube around our predicted function. Moreover in equation (3.32) we assumed that is feasible to approximate the function $f(.)$ with precision ϵ but since this is not usually the case, slack variables ξ_i and ξ_i^* are introduced . After receiving pairs of inputs and target variables (x_i, y_i) $\forall i \in \{1, ..., l\}$ the SVR algorithm is then equivalent to solving the following optimization problem.

$$\begin{aligned}
\min \quad & \frac{1}{2} \mid\mid w \mid\mid^2 + C \sum_i^l (\xi_i + \xi_i^*) \\
\text{Subject to} \quad & \\
& y_i - w \cdot \phi(x) - b \leq \epsilon + \xi_i \qquad \forall i \in \{1, ..., l\} \\
& w \cdot \phi(x) + b - y_i \leq \epsilon + \xi_i^* \qquad \forall i \in \{1, ..., l\} \\
& \xi_i, \xi_i^* \geq 0 \qquad \forall i \in \{1, ..., l\}
\end{aligned} \tag{3.33}$$

The objective function of the above optimization problem is a trade of between two terms, the regularized term and the empirical error. Minimizing the regularized term, $||w||^2$, will make the function as flat as possible. The empirical error term, $C \sum_i^l (\xi_i + \xi_i^*)$ penalizes the objective function if the prediction is outside of the ϵ-prediction tube. This implies that the SVR model can perform regression by solving an optimization problem with two parameters, the regularization constant C and precision parameter ϵ. Those parameters are controlled by the user and is often crucial to chose optimal hyperparameters to optimize the algorithms performance. For the scope of this thesis the optimization of the parameters is done by a trial and error grid-search. The SVR algorithm is illustrated in figure 3.9. In this study a Support Vector Regression algorithm is proposed to predict the arrival time of trucks in a transport mission between Malmö and Göteborg.

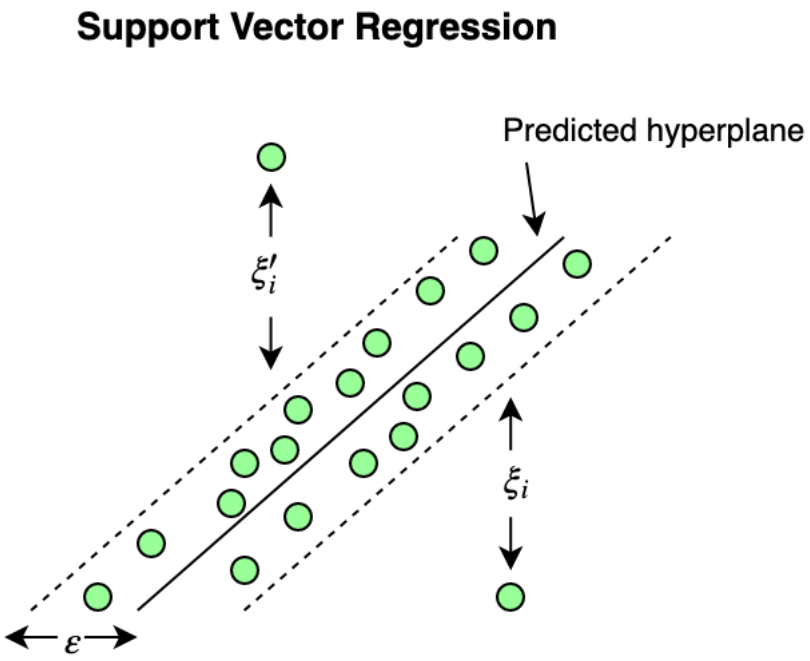


Figure 3.9: The modification of support vector machines for regression purposes. Points outside the ε region around the predicted hyperplane are penalized.

3.4 Decision tree based models

Decision trees are simple tree-structured decision tools that can be used for classification as well as for regression. A decision tree used for regression take as inputs the attributes of different observations and outputs a real valued prediction. A tree for classification take as inputs the attributes of many observation and outputs a prediction for the category the observations belong. A decision tree consists of internal nodes, branches and leafs. Internal nodes check if explanatory variables satisfy a certain condition, branches represent the outcome of those conditions and leaf nodes represent the predicted value or class. In other words the inputs space is split into many regions and a prediction for every region is produced based on the observations in the available data-set. In addition a greedy algorithm that minimize a cost function and a stopping criterion is usually used to determine the cut-points.

Initially, decision trees identify the region of the input space where an observation belongs. Then the average response for observations in that region is computed and given as a prediction. This study proposes random forest and gradient boosting algorithms for predicting the arrival time of trucks. Those algorithms utilise weak learners such as decision trees to produce a more accurate prediction. An example of how a decision tree splits the input space and produce its outputs is presented in figure 3.10.

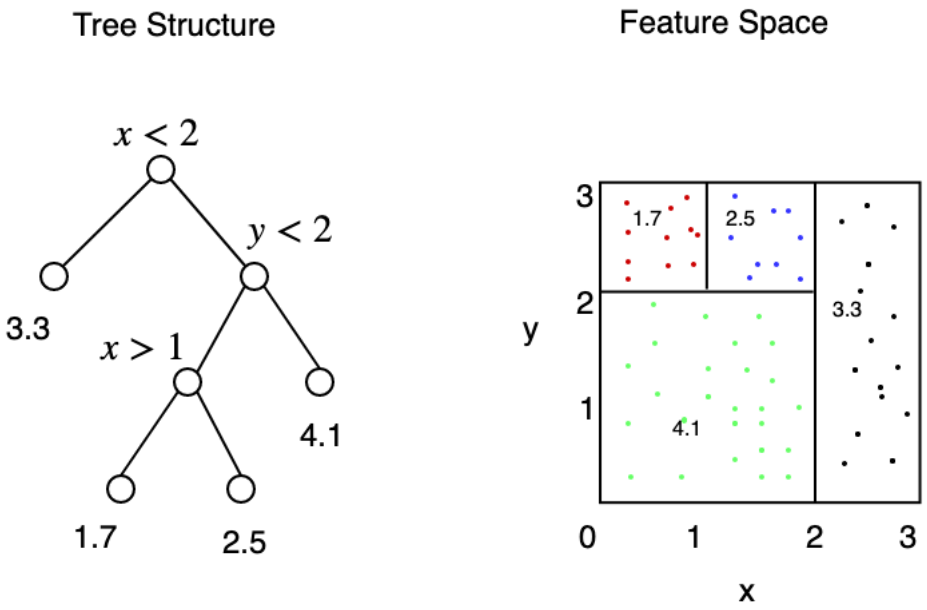


Figure 3.10: A simple example of how a decision tree separates the input space. The tree corresponds to a regression problem, in each region the mean value of all the observations are used as a prediction .

3.4.1 Random forest

The Random Forest algorithm [22] is an ensemble algorithm used for regression and classification. This method utilizes Decision trees (see 3.4) to produce its prediction. Even though the motivation of the algorithm is very simple the method most of the times produces satisfactory results and often outperforms other prediction models. Given a set of input variables and prediction (x_i, y_i) the method randomly samples with replacement to create a number of decision trees, this technique is known as bagging [6]. Moreover Random Forest applies a bagging method into the feature space as well. This randomness guarantees that all of the weak learners, decision trees, are uncorrelated. In other words, this method creates a forest of random and independent decision trees. Therefore the variance of the trees is reduced by sacrificing some bias. Then, in the case of regression problems, the average prediction of all decision trees is taken as the final prediction. In the case of classification problems the prediction is taken by a voting procedure. In other words the decision trees "vote" for a class and the winning class is the final prediction of the Random Forest algorithm.

In this study a Random Forest algorithm for regression is proposed to predict the arrival time of a truck in a mission between Malmö and Göteborg. A trial and error grid-search algorithm is utilised to optimise the parameters of the proposed random forest model. An example of the Random Forest algorithm with N decision trees for a regression problem is presented in figure 3.11.

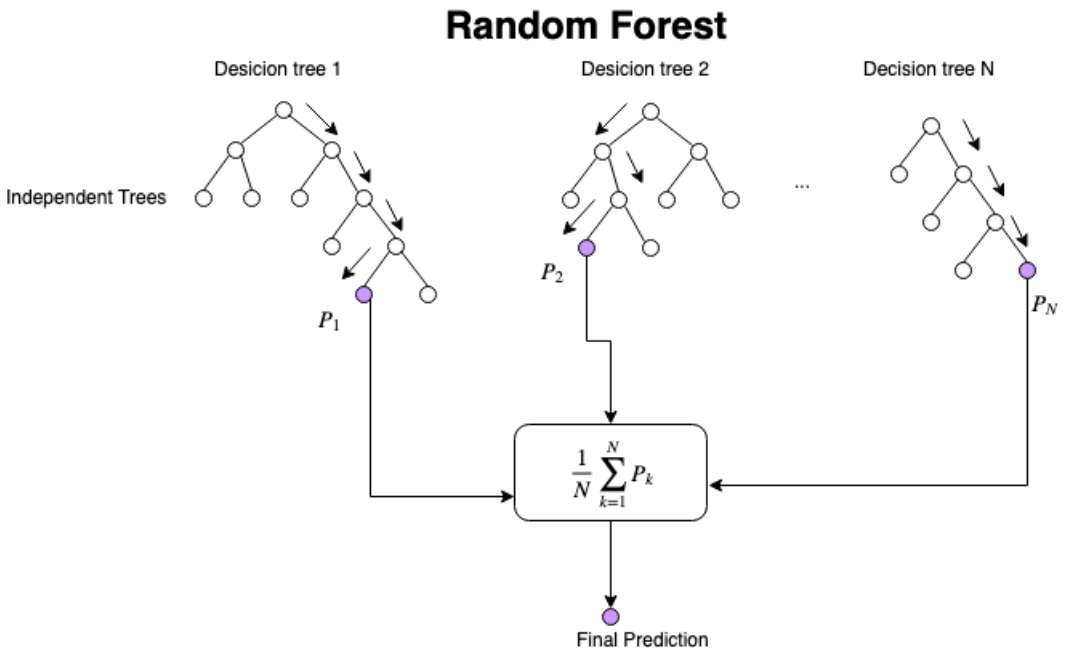


Figure 3.11: The random forest algorithm for a regression problem. N independent decision trees are created using the bagging technique both for feature space and data. In the end the average of the decision trees predictions is the final prediction of the random forest algorithm.

3.4.2 Gradient boosting

The Gradient Boosting algorithm [13] is an ensemble algorithm that can be used to deal with regression and classification problems. The Gradient Boosting algorithm utilises a technique called boosting [27] to obtain its predictions. The main idea of boosting is to utilise weak learners, for instance decision trees (see 3.4), and an optimization algorithm to optimize the predictions of the weaker model.

Unlike the previously discussed Random Forest model which produces decision trees in parallel Gradient Boosting is a sequential algorithm. The first step of this algorithm is to choose an error function and make an initial prediction using a decision tree. Since the decision tree model is a weak predictor, some errors are expected. Then a functional Gradient Descent algorithm is used to determine the next prediction of the algorithm. This algorithm is explicitly described below

- Let y be the target variable and $f_0(x)$ the initial prediction.
- Let $E(f_0(x), y) = \frac{1}{2}((y - f_0(x)))^2$ be the mean square error function.

Then the negative gradient of the error function with respect to $f_0(x)$ is given by

- $-\nabla E(f_0(x), y) = f_0(x) - y = h_1(x)$ is the residual of the model.

Then the same procedure is repeated to estimate the residual $h_1(x)$. In the end the prediction of the Gradient Boosting algorithm is given in equation (3.34)

$$F(x) = f_0(x) + f_1(x) + f_2(x) + ...f_N(x) \tag{3.34}$$

where $f_j(x)$ is the prediction given by a weak learner for the residual $h_j(x)$ $\forall j \in \{1, 2, ..., N\}$ and N is the iteration where a stopping criterion is met. In this study the Gradient Boosting algorithm was utilised to produce predictions for the total travel time of trucks operating in a specific mission. The algorithm is displayed in figure 3.12

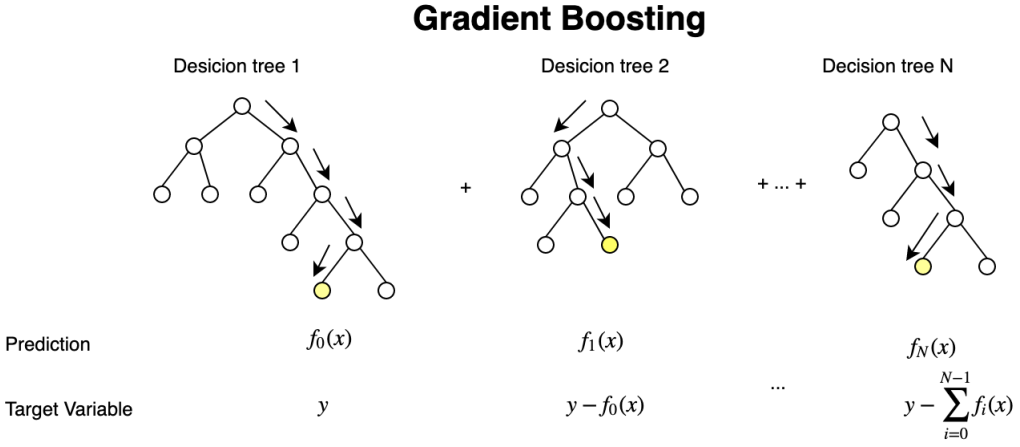


Figure 3.12: The Gradient Boosting algorithm. The first decision tree predicts the target variable y and the n_{th} tree predicts h_{n-1} $\forall n \in \{2, ..., N+1\}$. The sum of those weak predictors is the final prediction of the Gradient Boosting model.

3.5 Stacked generalization

The stacked Generalization method is an ensemble algorithm that can be applied to increase the accuracy of previously trained prediction models [34]. The main idea of this method is to combine the predictions of different machine learning models in order to achieve better prediction accuracy. This can be achieved by using a meta predictor algorithm with input variable the predictions of the previously trained models. This approach is also known as super learning [31]. The algorithm is explicitly described below.

- Let $P_i = f_i(x)\ \forall i \in \{1, .., N\}$ be the prediction of the prediction model i with input data x.
- Let g be the function describing the approximation of the meta predictor model.
- Then the final prediction W is given by

$$W = g(P) = g(f(x)) \tag{3.35}$$

where x are the input data and $P, f(.)$ are vector with elements the predictions P_i and approximation functions $f_i(.)$ respectively. In this study a Stacked Generalization method was implemented with a linear regression meta predictor to estimate the total travel time of trucks in a mission between Malmö and Göteborg. This machine learning method is presented in detail in the figure 3.13.

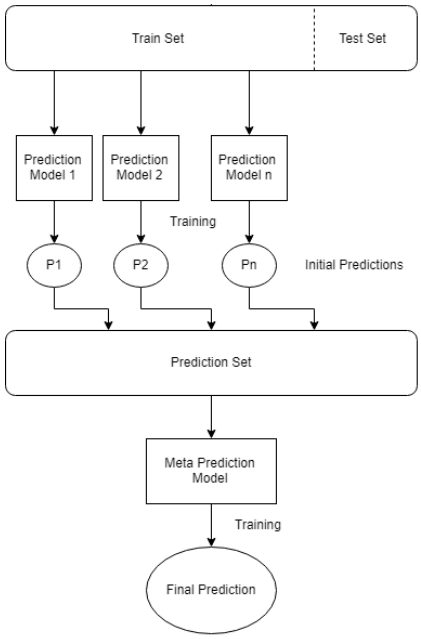


Figure 3.13: Flow chart demonstrating how stacking technique generally works. First, all of the initial prediction models are trained using the available data. Then a meta prediction model is trained to make a final prediction using all the predictions of the initial algorithms as additional inputs.

4
Implementation

In this chapter the methodology we utilized to approach the problem, the libraries used in the study as well as how the data are processed and used to design, implement and evaluate the prediction models are described in detail. The aim of this chapter is to clearly describe the implementation of our models so that any curious reader can reproduce the results of the study.

4.1 Libraries

The implementation of the study is made in Python language. Python was chosen as the preferred language since a variety of optimized libraries exist that can assist us with data processing, prediction model implementation and evaluation. Some of the libraries this study utilises are the Pandas, Tensorflow, Keras, NumPy and Scikit-Learn.

4.1.1 Pandas

Pandas is an open source, BSD-licensed library [24] for the Python programming language. Pandas library is one of the most used Python libraries for data manipulation and analysis. It is a trivial to use library that offers data structures and operations for manipulating numerical tables and time series. This study utilises this library to manipulate the input data and create dataframes that will be used to train and evaluate the prediction models.

4.1.2 Tensorflow and Keras

Tensorflow is an open source library used for machine learning applications such as neural networks [2]. It is an optimized library developed by Google that utilises dataflow graphs to represent computations. Keras is an open-source neural-network library for neural networks. It provides a high-level neural networks API and is capable to run on top of Tensorflow. This study uses Keras on top of Tensorflow to build a neural network that predicts arrival times of trucks.

4.1.3 NumPy and Scikit-learn

NumPy is a library for Python that supports large, multi-dimensional arrays and matrices. It also supports a plethora of high-level mathematical functions that

can operate on these arrays and are useful in many scientific and engineering applications. Scikit-Learn is an open source, BSD-licensed library for the Python programming language. It is built on top of the NumPy library and it provides variety of machine learning algorithms for classification, regression and clustering problems. This study utilises Scikit-Learn on top of NumPy to build the Random Forest, Gradient Boosting, Stacked Generalization and Support Vector Regression prediction models.

4.2 Data processing

In this section the functions created to clean and process the position related data from the GPS sensors as well as the weather related data obtained from an online dataset are analyzed and discussed. The position related data were collected from different transport missions between Malmö and Göteborg between 16^{th} of December 2015 and 25^{th} of June 2018.

4.2.1 GPS data

For the purposes of the study spatio-temporal truck related data were collected by GPS sensors installed in Volvo FH-180 trucks during the period between 16^{th} of December 2015 and 25^{th} of June 2018. The GPS sensors provided us with 116 truck features but only 6 features were selected for the study. The selected variables are displayed in table 4.1.

Variable Name	Variable Description
Computer Time(yyyy:mm:dd hh:mm:ss)	Date and time of the record.
Vehicle Speed (kmph)	Truck speed in Km per Hour.
GPS heading (°)	GPS heading in degrees.
Longitude (°)	Longitude of the truck.
Latidute (°)	Latitude of the truck.
Gross Weight (Kg)	Gross weight of the truck.

Table 4.1: Description of the truck related variables collected from GPS sensors.

4.2.2 Cleaning the data

The following changes to the data were made

- Observations with missing values were removed.
- Observations with no logical values were removed.
- Observations corresponding to the truck idle hours were removed.
- Observations corresponding to missions that lack information about the arrival of the truck were removed.
- Day variable was amended in the data-set. For example when this variable value is 1, it corresponds to a transport mission on Monday. To compute this variable we utilised the dates of the records.
- Mission's starting and ending points coordinates were chosen as the coordinates that the truck remained idle for a long period of time with the hand brake enabled.

4.2.3 Haversine formula:

In order to compute the distance from the mission's starting and ending point the Haversine Formula was utilised. Haversine formula computes the distance between two points in a sphere given the points latitudes and longitudes. The formula is displayed below

- Let ϕ_1, ϕ_2 = latitude of point 1 and latitude of point 2
- Let λ_1, λ_2 = longitude of point 1 and longitude of point 2
- Let r be the radius of earth = 6,371km
- Distance d is then given by

$$d = 2r \arcsin\left(\sqrt{\sin^2(\frac{\phi_2 - \phi_1}{2}) + \cos(\phi_1)\cos(\phi_2)\sin^2(\frac{\lambda_2 - \lambda_1}{2})}\right) \tag{4.1}$$

For this study a function that takes as inputs the geographical coordinates, longitudes and latitudes, of two points and return their distance on earth was developed.

4.2.4 Weather data

It was previously mentioned that arrival time of trucks is influenced by stochastic factors such as weather. In order to get weather data for different missions and specific times we used the Wunderground weather dataset that is available online. Specifically `http://oco-carbon.com/wunderground-weather-data-downloader/` website provides a tool to download Wunderground weather data in CSV format for given weather station and date range. The weather stations chosen for this study as well as a short description for their location is briefly presented in table 4.2.

Weather Station Name (ID)	Weather Station Description
ISKNEMAL1	Weather station collecting weather related data for the Malmö city
ISKNELDD2	Weather station collecting weather related data for the region between Malmö and Helsingborg.
IHELSING2	Weather station collecting weather related data for the Helsingborg city.
ISHALLAND39	Weather station collecting weather related data for the area near Halmstad city.
ISHALLAND64	Weather station collecting weather related data for a region near Göteborg city.

Table 4.2: The weather stations chosen to collect the weather data

Moreover since the weather data are generated in five minute intervals and the data from the trucks GPS sensors are generated every five seconds a function was created to synchronise the weather variables with the truck related variables. To achieve this, the study assumes that weather variables are constant in the five minute interval between every weather observation and hence the weather only changes every five minutes . This assumption is realistic since weather behaviour cannot rapidly change and is most of the times constant. In addition to that weather data were spatially synchronised. In other words only weather related data from the nearest weather station and corresponding five minute interval were considered. The weather related variables and a short description of every variable are briefly shown in table 4.3.

Weather Related Variables	Weather Variable Description
Date (yyyy: mm: dd)	Date of the recorded weather.
Time (hh: mm: ss)	Time of the day for the recorded weather.
Temperature (°F)	The atmospheric temperature in a region around the weather station.
Dewpoint (°F)	The dewpoint in a region around the weather station.
Humidity (%)	Humidity in a region around the weather station.
Wind Direction (°)	Wind Direction in a region around the weather station. Degrees are measured by the angle between the North and the wind direction. For example ninety degrees corresponds to wind direction towards the East and 270 degrees corresponds to wind towards the West.
Wind Speed (mph)	Wind Speed in Miles per hour.
Pressure (in)	Pressure in a region around the weather station.

Table 4.3: Description of the weather related variables. Date and Time variables were used to synchronise the weather related data with the truck related data.

4.2.5 Computing wind advantage feature

We created a new feature named Wind Advantage using the input variables wind direction and GPS heading. This variable is a measure on how wind direction influence the vehicle movement. For instance this variable is positive when the angle θ between wind direction and GPS heading is $\theta \in [-90, 90]$ and negative otherwise. This feature is computed as follows

- Let ϕ be the wind direction and
- Let ζ be the GPS heading. Then the wind advantage is given by

$$Wind\_Advantage = \cos(\phi - \zeta) = \cos(\theta) \tag{4.2}$$

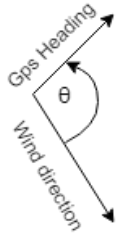


Figure 4.1: An example displaying how the wind advantage feature is computed. Wind advantage feature is given by $\cos(\theta)$.

4.3 Train and test data

In previous paragraphs the functions that helped as clean and process the data from two year worth of truck data log files were discussed. The main purpose of the mentioned implementations was to create a train data set to train our prediction models and a test data set to evaluate the accuracy of the proposed models. For the creation of the train set, fifty different missions were randomly selected and processed. For the creation of the test data set twenty different transport missions not included in the train set were randomly selected. Then weather related data for the dates of the missions were amended into both datasets.

Moreover the dates of missions in the test set were randomly selected. However the dates of those missions were carefully checked to be different from the mission dates of the missions in the training set. This check was performed to ensure that missions in the test set do not have any correlations with missions from the train set since the purpose of the study is to evaluate the models in completely new missions.

4.4 Error definitions

For evaluating the ability of the models to predict arrival times we define the following error functions.

Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{100}{N} \sum_{i=1}^{N} \mid \frac{P_i - O_i}{O_i} \mid \tag{4.3}$$

Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (O_i - P_i)^2} \tag{4.4}$$

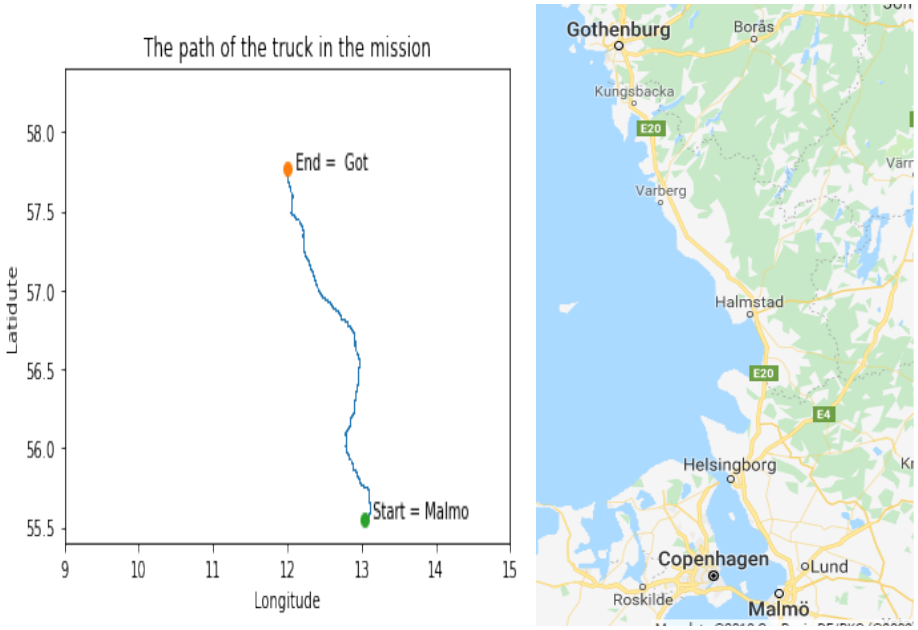
where O_i is the observed value of the response variable and P_i is the prediction of that variable.

4.5 Data exploration

This section provides a visualisation of the spatio-temporal data obtained from the trucks GPS sensors. More specifically the mission path and speed profiles of different mission are displayed in the following paragraphs. This analysis is essential to guarantee the correctness of the given data.

4.5.1 Mission route

The mission path of the route is created when longitudes are plotted against latitudes. The axis limits are manipulated to deal with the fact that longitude values $\lambda \in [-180, 180]$ and latitude values $\phi \in [-90, 90]$. The simulated mission path is compared with the real mission path in figure 4.2.



(a) Longitudes plotted against latitudes to create the mission route path.

(b) Figure from google maps

Figure 4.2: A comparison between the simulated mission path with the real path.

4.5.2 Speed profiles of missions

The speed profile of three different mission are displayed in figure 4.3. From the figure it is observed that the speed is almost constant at 80 $\frac{Km}{h}$ when the truck is in the highway. Moreover speed profile fluctuations are observed when the truck passes through cities. This phenomenon agrees with the initial study assumptions that uncertainties related to traffic congestion and traffic lights can influence the speed profile and hence the total travel time of the operating truck. In addition to that it is observed that the truck stops at certain locations for a long periods of time. The factor that causes this behaviour is discussed in the next paragraph.

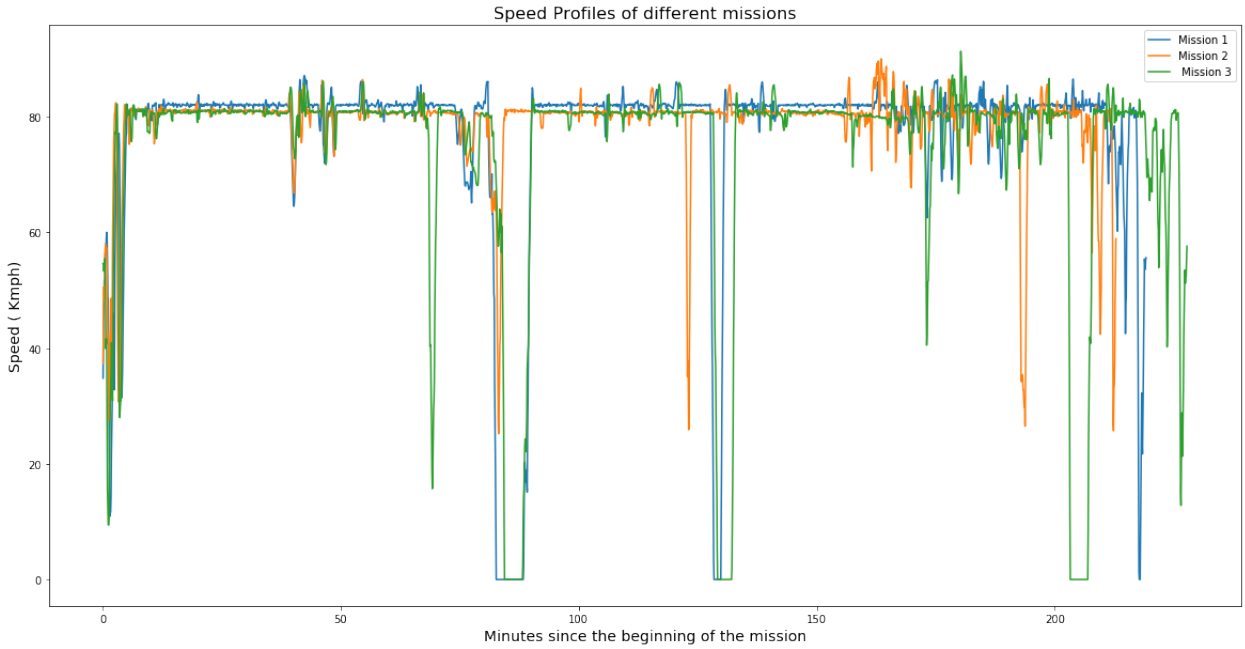


Figure 4.3: Speed profile of three different missions. Every time point corresponds to one minute interval. Driver break duration as well fluctuations in speed due to traffic congestion are also visible.

4.5.3 Driver breaks

The duration of the studied mission is very long and hence it is very common for the driver to take a break. Unfortunately the duration of the driver breaks are impossible to predict. Hence missions with no breaks and missions with very long driver breaks are uncommon and are considered as outliers in the data. This statistical analysis of the data helped as identify and remove outliers from the data which resulted in a decrease of the prediction error of the proposed models. This finding makes the study different than the studies found in the literature since this factor only influence extremely long missions. Most of the studies in the literature are related to predicting arrival times of city buses, hence this impossible to predict factor is not existent. The distribution of the total mission's duration before removing the uncommon missions from the data is displayed in figure 4.4. To create the empirical distribution in the figure missions from both test and train sets were used. Moreover since the total duration of the missions are normally distributed we computed the standard deviation of that distribution. The standard deviation of the empirical distribution corresponds to the Root Mean Square Error of the prediction model that uses as prediction the mean of the distribution. In the evaluation section this result is compared with the errors of the proposed prediction models.

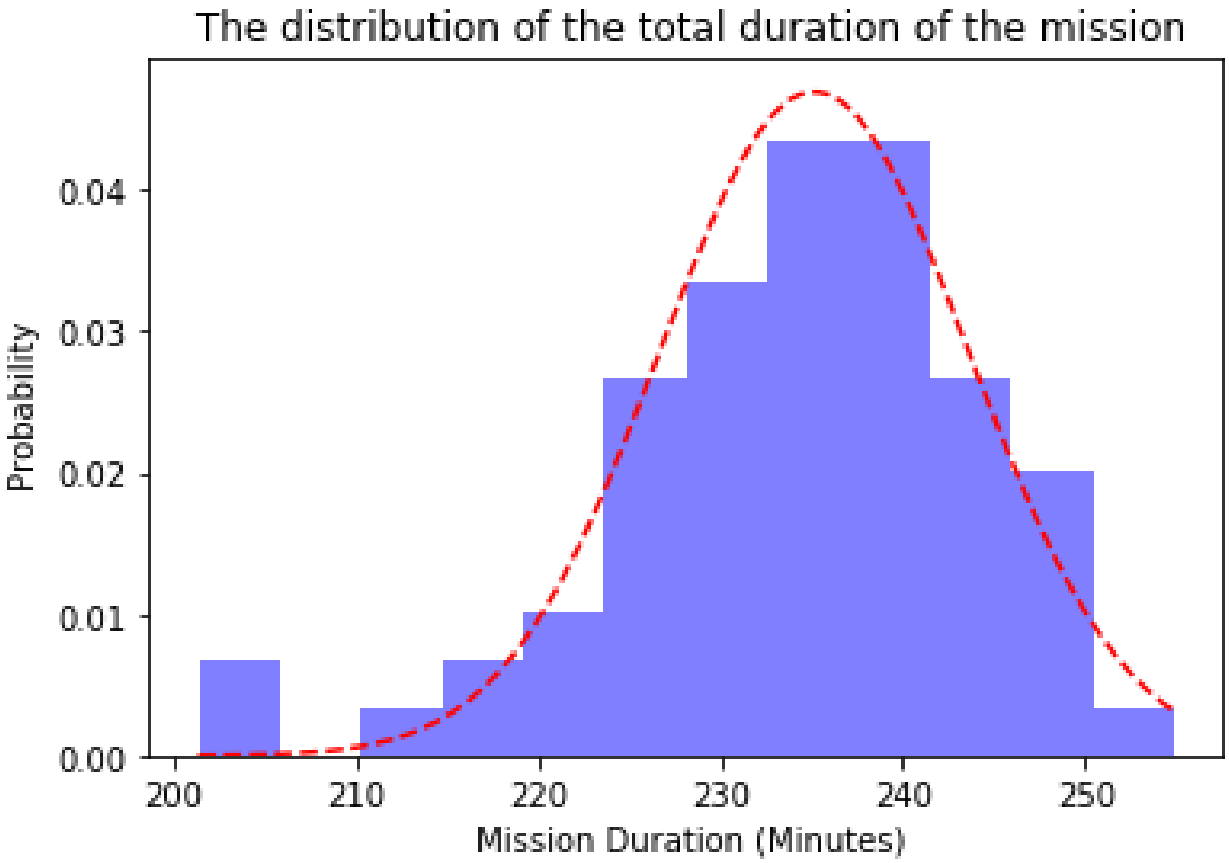
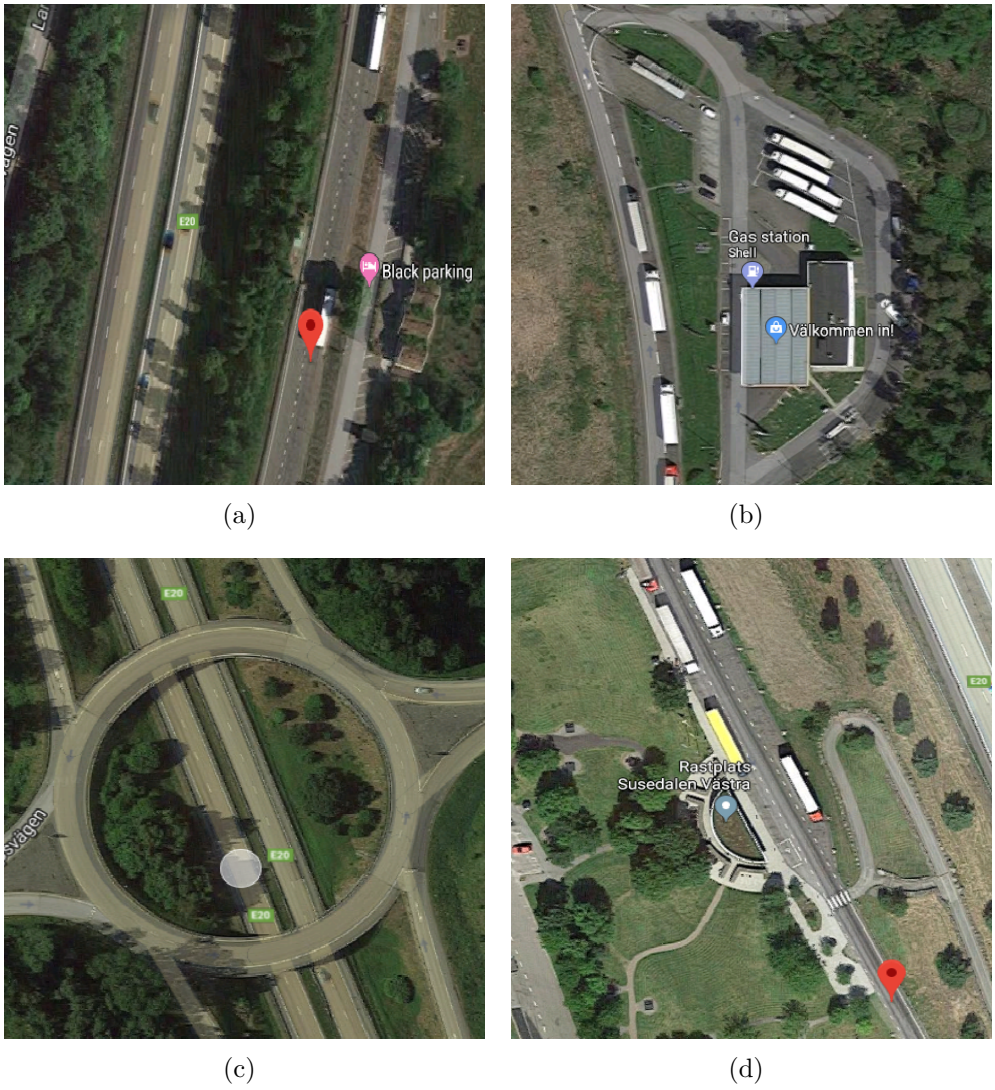


Figure 4.4: The empirical distribution of the total mission duration is compared with a normal distribution with the same mean and variance. Outliers can easily be identified from the figure.

4.5.4 Locations of the driver breaks

To justify our claims, that the mission is greatly affected by driver breaks, the locations where the truck remained idle for more than ten minutes were manually found using google maps and the longitude and latitude variables from the available data. Figure 4.5 demonstrates some of the resting places the drivers usually take their breaks. Some of those locations are gas stations, resting places, cafeterias and in some cases those location are roads where queuing problems might have occurred due to an accident or traffic congestion caused by road construction.



(a) (b)

(c) (d)

Figure 4.5: Four locations where drivers tend to stop. (a) a resting place (b) a gas station (c) a road affected by traffic congestion, and (d) a resting place. Figures are from Google Maps.

4.5.5 Data correlation

Lastly, in order to identify which features influence the arrival time of the truck we computed and visualised the correlation matrix of the data. According to our findings, the variables that are correlated with the arrival time are the time of the day, the geographic coordinates, the vehicles speed, the distances from starting and ending points, the GPS heading and humidity. A heat-map displaying correlations between the input variables is displayed in 4.6. This result is used to eliminate 'bad' features, features that are not correlated with the response variable, from the models.

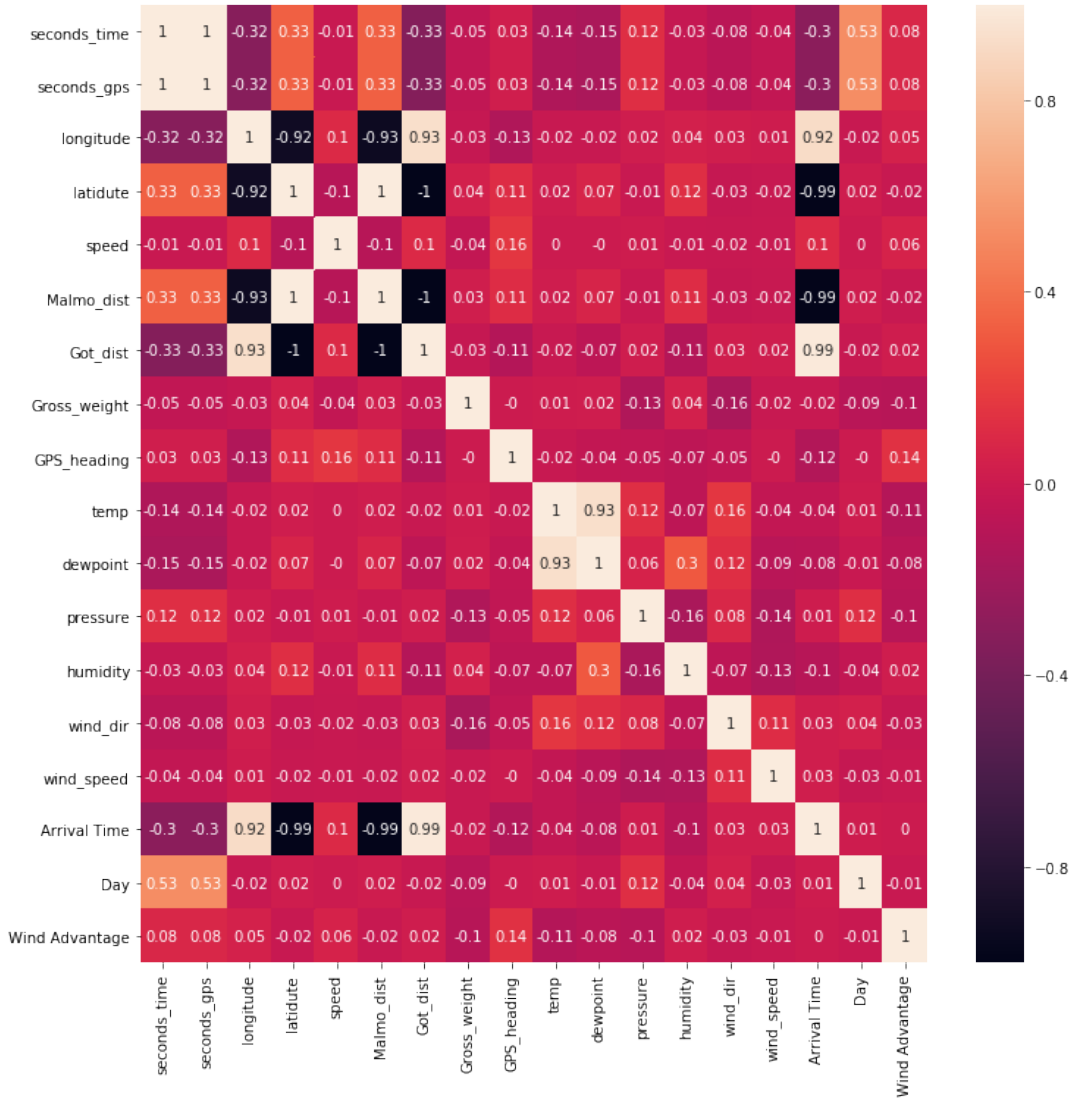


Figure 4.6: Correlation heat-map of the input variables.

4.6 RNN data processing

As it was previously mentioned recurrent neural networks are able to learn a dynamical system unlike feed forward networks that can learn input to output mappings. Hence additional processing of the data is necessary. Up to this point we have a dataset of M observations and N features for some integers M and N. However the inputs to a RNN should be three dimensional which implies that our two dimensional inputs require further processing. In order to obtain the correct format for inputs to the RNN models we first have to define how many previous time steps our network will keep into its memory. In other words the format of the inputs into the RNN model should be three dimensional. Eventually what we feed into the network should have K inputs containing information of the previous L time-steps with F features, where K, L and F are integer numbers. To obtain such format for our data we used the Numpy library to reshape our datasets. By reshaping the dataset

further cleaning of the data was necessary. This happens since when moving from one mission to another there is a danger that the network will remember information from the previous mission which will cause large errors. To guarantee that this never happens we created functions to process and clean the data to obtain suitable input for the RNN network.

4.7 Random forest model

The first proposed prediction model is a Random Forest (see 3.4.1) model implemented in the Scikit-learn library (see 4.1.3). The parameters of the model were tuned with a trial and error grid search. Table 4.4 shows the optimal parameters chosen by the grid search algorithm.

Parameter Name	Value
bootstrap	True
max_depth	120
max_features	sqrt
min_samples_leaf	1
min_samples_split	5
n_estimators	200

Table 4.4: The final parameters of the Random Forest algorithm after tuning them with a grid search algorithm.

4.8 Gradient boosting model

The second proposed prediction model is a Gradient Boosting (see 3.4.2) model implemented in the Scikit-learn library. The parameters of the model were tuned with a trial and error grid search. Table 4.5 shows the optimal parameters chosen by the grid search algorithm.

Parameter Name	Value
learning_rate	0.05
min_samples_split	0.2
min_samples_leaf	0.1
max_depth	3
max_features	log2
subsample	0.95
n_estimators	300

Table 4.5: The final parameters of the Gradient Boosting algorithm after tuning them with a grid search algorithm.

4.9 Support vector regression model

The third proposed prediction model is a Support Vector Regression (see 3.3.2) model implemented in the Scikit-learn library. The parameters of the model were tuned with a trial and error grid search. Table 4.6 shows the optimal parameters chosen by the grid search algorithm.

Parameter Name	Value
Kernel	rbf
C	250
epsilon	0.31
gamma	0.01

Table 4.6: The final parameters of the Support Vector Regression algorithm after tuning them with a grid search algorithm.

4.10 Feed forward network

The fourth proposed prediction model is an Feed Forward Neural Network (see 3.1) implemented in the Keras library (see 4.1.2). The parameters of the model were manually tuned. The model consisted of two hidden layers with ten and five neurons respectively. The activation function chosen for hidden layers was ReLU and the initial weights were sampled from a normal distribution. Also since the model should output a real value the activation function for the output layer was chosen to be linear. Moreover in every layer a L_1 regularisation scheme was applied. Table 4.7 shows some of the parameters chosen for the final model.

Parameter Name	Value
Learning Rate	0.1
Loss	MSE
Decay	e-5
Batch size	128
Epochs	30000
Callbacks	EarlyStopping
Optimizer	Adam
Regularization Constant	0.01

Table 4.7: The final parameters of the Feed Forward Neural Network mode after manually tuning the parameters.

4.11 Stacked generalization model

The fifth proposed prediction model is a Stacked Generalization (see 3.5) model. The implementation was performed using the Scikit-learn library. The meta predictor model was chosen manually after experimenting with a variety of prediction models. After experimenting with different meta predictor models we chose a linear regression model as the final meta predictor model.

4.12 Recurrent neural network model

The final proposed prediction model is a Recurrent Neural Network (see 3.2) implemented in the Keras library (see 4.1.2). This network consists of a layer of LSTM cells and an output layer with linear activation function. Table 4.8 shows some of the parameters chosen for the final model.

Parameter Name	Value
Learning Rate	0.1
Loss	MSE
Decay	e-4
Batch size	128
Epochs	30000
Callbacks	EarlyStopping
Optimizer	Adam
Regularization Constant	0

Table 4.8: The final parameters of the Recurrent Neural Network model after manually tuning the parameters.

5
Evaluation

This chapter presents all the findings and results obtained by the experiments performed in the study. Initially the performance of historic data models is discussed and then the predictions of the machine learning models are evaluated and discussed.

5.1 Predictions using historic average

In the first experiment we used the historical average mission duration of missions in the train set as the prediction. This approach, as expected, produced very poor results. More specifically the Root Mean Square Error of this model was approximately 508 seconds. Therefore historic data models are not suitable and definitely not the optimal choice to give predictions to missions with many uncertainties. This result ultimately confirms Mehmet and Metin [3] claims that models based on historic data are unreliable when traffic conditions are unstable since their accuracy relies on similarities between real time and historic traffic patterns. This outcome also agrees with Jeong et al. [17] claims, that machine learning models outperform historic data based models. The studied mission is a very long transport mission that is greatly influenced by traffic congestion and driver breaks which makes those models incapable of producing accurate predictions.

5.2 Machine learning based predictions

This paragraph describes the performance of the machine learning prediction models. These models are trained using three different training schemes. Every training scheme utilises different variables as input to the prediction models. In particular in the first training scheme the models utilise every available variable, in the second only truck related variables are considered and in the third only the variables correlated with the travel time are used. In addition to that all models are trained with observations sampled with certain frequency. More specifically, since for the most part of the thesis we did not have access to a computer to run complex models, we reduced the complexity of the models using observation sampled every one, three and five minutes. We did not notice any significant difference regarding the accuracy of the models when this modification was used, while in the same time the complexity of the models dramatically decreased. In the next paragraphs the errors of the models are presented.

5.2.1 Training scheme 1

The performance of every algorithm in terms of Root Mean Square Error (RMSE) in seconds and Mean Absolute Percentage Error (MAPE) is displayed in table 5.1. Those errors are obtained by evaluating the predictive capability of the models using the test set.

Model	Frequency	MAPE (%)	RMSE (Seconds)
RF	5 minutes	6.79	315
	3 minutes	5.88	311
	1 minute	5.43	316
	5 seconds	5.24	321
FFNN	5 minutes	14.40	375
	3 minutes	14.78	391
	1 minute	13.18	390
	5 seconds	12.02	400
SVR	5 minutes	16.97	457
	3 minutes	16.27	458
	1 minute	15.56	447
	5 seconds	9.14	392
RNN	5 minutes	13.45	490
	3 minutes	12.46	460
	1 minute	11.52	465
	5 seconds	12.96	480
GB	5 minutes	16.31	384
	3 minutes	15.91	384
	1 minute	15.74	385
	5 seconds	19.06	373
SG	5 minutes	5.68	315
	3 minutes	6.91	322
	1 minute	6.51	353
	5 seconds	4.96	325
Best Error		**4.96**	**311**

Table 5.1: The MAPE and RMSE of the models trained on observation sampled with certain frequency using all the available input variables. The errors correspond to prediction errors of missions in the test set.

5.2.2 Training scheme 2

The performance of every algorithm in terms of RMSE in seconds and MAPE is displayed in table 5.2. Those errors are obtained by evaluating the predictive capability of the models using the test set.

Model	Frequency	MAPE (%)	RMSE (Seconds)
RF	5 minutes	5.29	314
	3 minutes	6.15	320
	1 minute	6.13	333
	5 seconds	4.70	324
FFNN	5 minutes	13.80	359
	3 minutes	12.43	376
	1 minute	12.48	376
	5 seconds	9.42	379
SVR	5 minutes	10.40	385
	3 minutes	9.11	386
	1 minute	8.90	398
	5 seconds	8.75	383
RNN	5 minutes	13.46	490
	3 minutes	11.25	480
	1 minute	6.8	439
	5 seconds	8.85	404
GB	5 minutes	18.40	364
	3 minutes	13.79	365
	1 minute	13.32	367
	5 seconds	19.03	359
SG	5 minutes	5.51	312
	3 minutes	6.21	320
	1 minute	6.49	336
	5 seconds	4.96	325
Best Error		**4.70**	**312**

Table 5.2: The MAPE and RMSE of the models trained on observation sampled with certain frequency using only truck related variables. The errors correspond to prediction errors of missions in the test set.

5.2.3 Training scheme 3

The performance of every algorithm in terms of RMSE in seconds and MAPE is displayed in table 5.3 Those errors are obtained by evaluating the predictive capability of the models using the test set

Model	Frequency	MAPE (%)	RMSE (Seconds)
RF	5 minutes	6.53	304
	3 minutes	6.26	302
	1 minute	6.48	311
	5 seconds	5.21	311
FFNN	5 minutes	11.32	372
	3 minutes	11.52	374
	1 minute	12.78	330
	5 seconds	10.35	396
SVR	5 minutes	10.42	385
	3 minutes	13.29	393
	1 minute	10.91	384
	5 seconds	8.80	373
RNN	5 minutes	11.46	400
	3 minutes	12.08	405
	1 minute	10.54	449
	5 seconds	8.99	385
GB	5 minutes	18.40	364
	3 minutes	12.04	343
	1 minute	13.61	338
	5 seconds	17.47	363
SG	5 minutes	6.67	309
	3 minutes	5.91	302
	1 minute	6.80	321
	5 seconds	6.25	317
Best Error		**5.21**	**302**

Table 5.3: The MAPE and RMSE of the models trained on observation sampled with certain frequency using the variables correlated with the response variable. The errors correspond to prediction errors of missions in the test set.

5.3 Performance evaluation

The last part of our experimental design consists of the evaluation of the performance of the developed models. The models are compared in terms of Root Mean Square and Mean Absolute Percentage Errors.

5.3.1 Benchmark comparison

For the benchmark comparison of the predictive capabilities of the models, it is essential to visualise the errors of different models. In this paragraph we present plots demonstrating the Root Mean Square Errors and Mean Absolute Percentage Errors of the developed models. In particular, in figure 5.1 the RMSE of different experiments are presented while in figure 5.2 the MAPE are displayed.

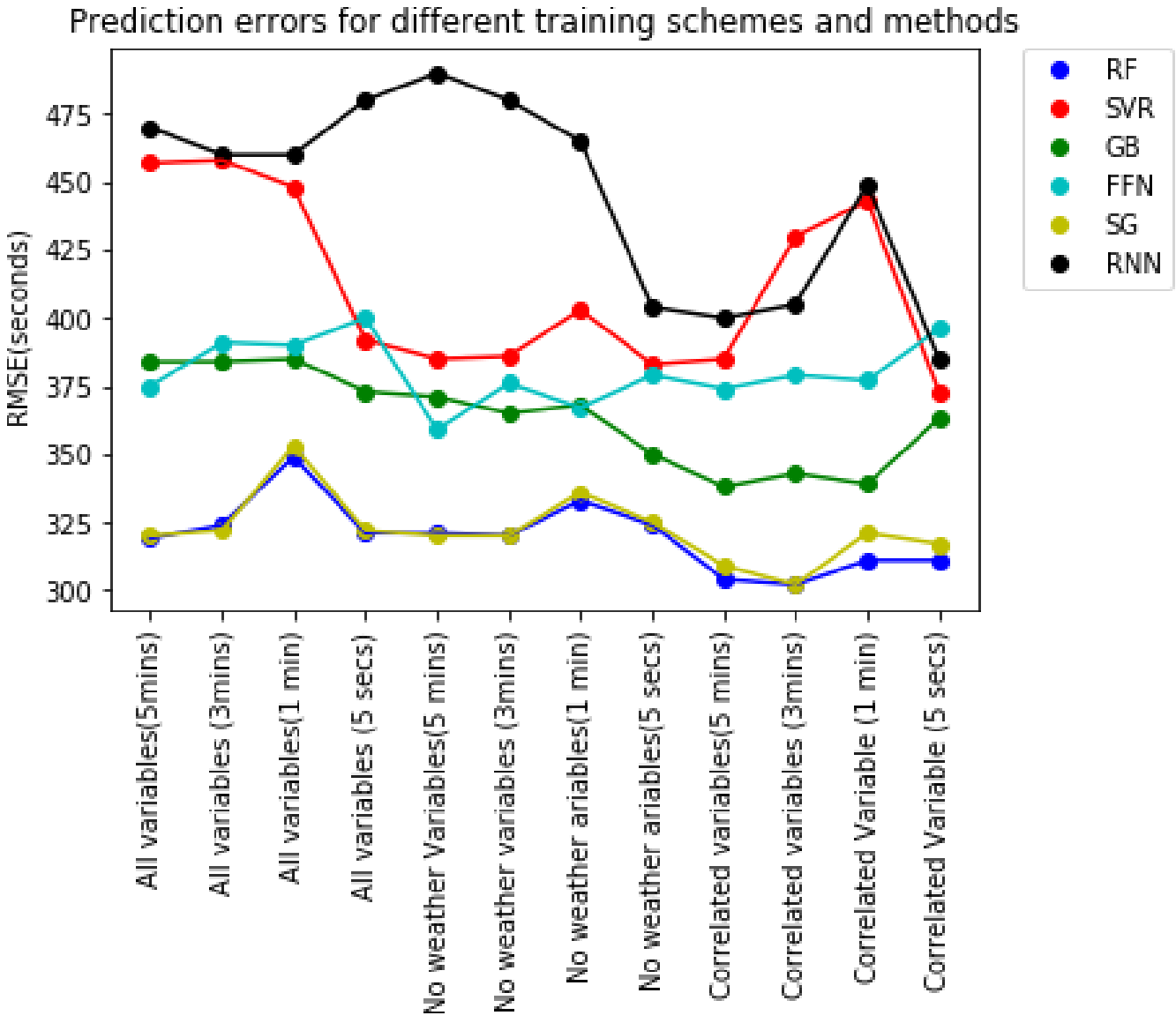


Figure 5.1: The RMSE of every model for different training schemes.

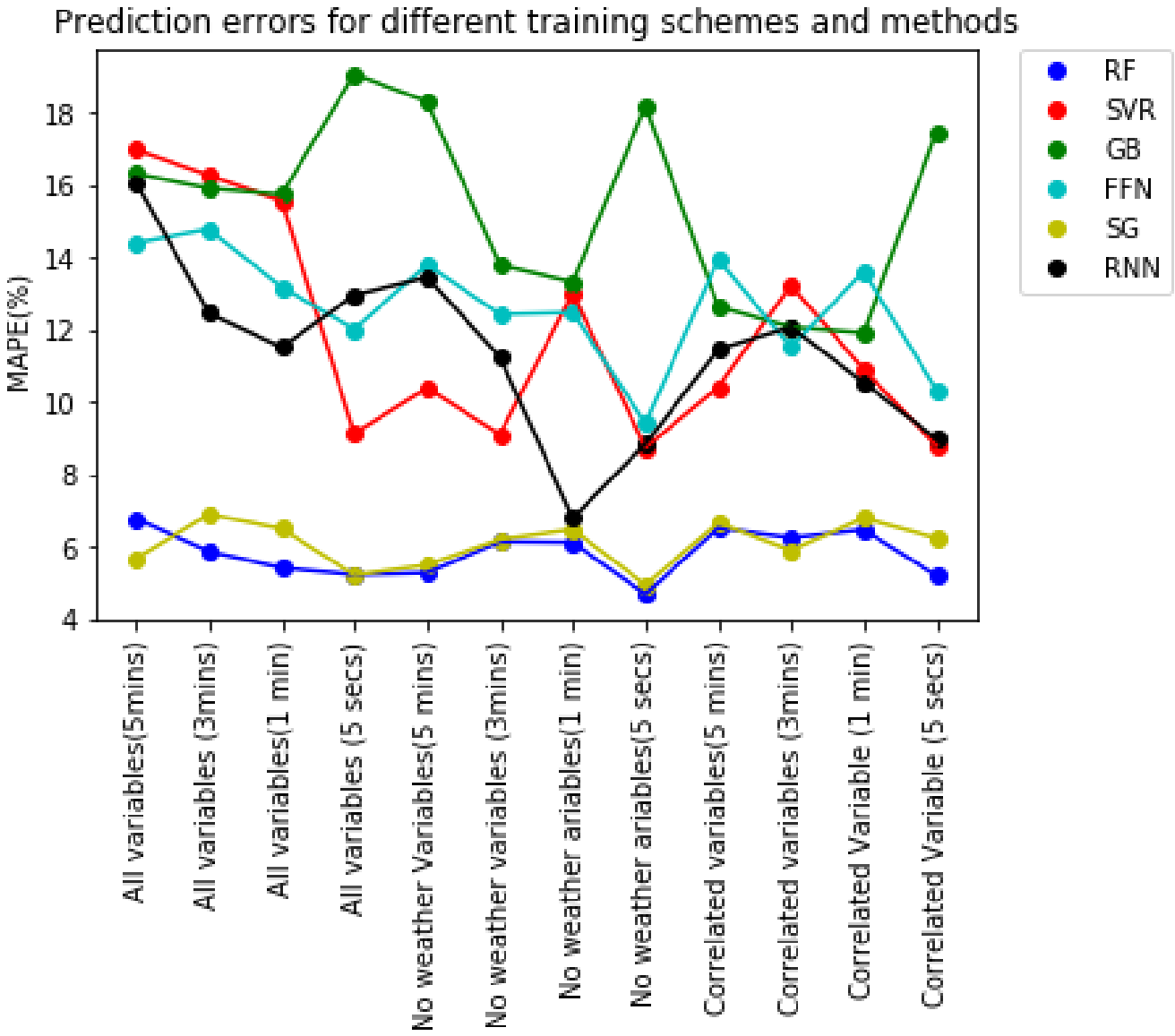


Figure 5.2: The MAPE of every model for different training schemes.

5.3.2 Visualizing the predictions

Figure 5.3 demonstrates the ability of the models to predict the remaining mission time at the last hour of a mission. The blue line corresponds to the observed arrival time and the other lines are predictions of the models. This plot undoubtedly illustrates the predictive capability of the models. Moreover it is easily observed that the Random Forest predictions, orange line, is closer to the actual arrival time than the other models which confirms the study findings that this model showed a superior performance compared to other models. Moreover in the last part of the mission, random forest manages to completely negate the errors which result in very low MAPE errors. In contrast, the gradient boosting model, red line, which showed a balanced performance for most part of the mission, failed to capture the trend in the last minutes of the mission. As a result, this model gave the poorest

performance in terms of Mean Absolute Percentage Error.

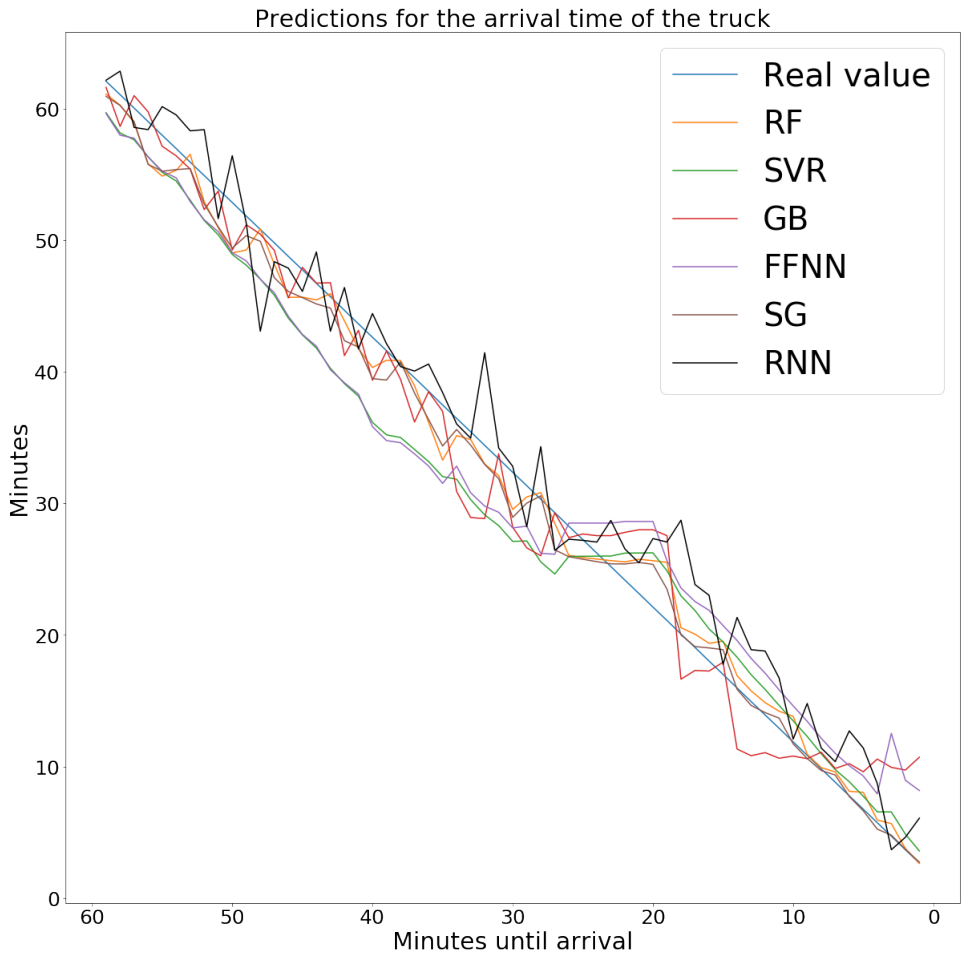


Figure 5.3: The models predictions for the last hour of a mission for models trained using all the variables sampled every five minutes. Blue line corresponds to the actual arrival time and rest of the lines correspond to the predictions given by the models.

5.4 Discussion of the results

According to our results using historic data based model, such as using the historic mean travel time as a prediction results does not provide accurate predictions (RMSE = 508 seconds). This argument can be easily justified by comparing this error with the RMSE of the proposed models. In particular the RMSE of most of the models was very close to 350 seconds and even the worst performing model had an error of 490 seconds. This finding confirms [17] claims that machine learning models can capture the complex non-linear relationship between the explanatory variables and the total travel time. This result can also be connected to [3] conclusions that historic average models are not suitable when traffic conditions are unstable.

In terms of Root Mean Square Error of the machine learning proposed models, all models managed to achieve acceptable performance. In particular Random Forest model had the lowest RMSE of all models for all training schemes and training frequencies. The Stacked Generalization that utilises the predictions of the models as inputs to a meta predictor displayed similar performance. However as figure 3.8 shows the differences between this approach and Random Forest are not significant which makes RF the winning model since SG model requires development of many other models. The lowest RMSE, 302 seconds, was observed when Random Forest was trained using only correlated variables with observations sampled every three minutes. Similar results were observed in terms of Mean Absolute Percentage Error with Random Forest demonstrating a superior performance compared to other methods. The lowest MAPE (4.70%) was observed when Random Forest was trained using only truck related variables with observations sampled every five minutes. The Gradient Boosting model that was the second best model after Random Forest in terms of RMSE had the worst performance in terms of MAPE and failed to capture the trend in the last minutes of the mission. The RF model completely negated the errors in the last minutes and displayed the lowest MAPE. This behaviour can be easily explained since most of the models solve certain optimization problems to obtain their predictions and may not converge to solutions that can accurately predict the last minutes of the mission. On the other hand Random Forest take into account only predictions from decision trees in the correct region of the feature space and hence it provides very accurate predictions in the last minutes of the mission.

In regards to the appropriate choice of explanatory variables it was found that truck related data from GPS sensors are sufficient to provide accurate predictions. This result verifies Patnaik et al. [25] claims, that weather variables are insignificant when predicting arrival times of urban buses. This finding does not agree with [29] conclusions that utilizing weather data dramatically increased the prediction accuracy of their models. However [29] studied a mission operated in Fushun West Open-pit Mine in China while we considered a transport mission in Swedish highways and roads. Therefore, further study is required to decide if weather variables should be considered when developing such models for trucks operating in rough environments.

One of the main contributions of the thesis is the study of a long transport mission, approximately four hours long, that is greatly influenced by human behaviour. More specifically, the travel time of the mission does not only depend on traffic and

weather behaviour but also on the duration of the driver break. In contrast, most studies in literature are related to urban buses where this factor is not existent. This is an important element to take into consideration when designing similar models for fully autonomous vehicles having in hand data that are greatly disturbed by human behaviour.

6
Conclusion

The main objectives of the study were to design, develop and evaluate the ability of machine learning models to predict the arrival time of a truck operating in transport mission as well as to identify which features are optimal to be used as inputs to the predictions models.

6.1 Answers to the research questions

For the fulfilment of the main aims of the study we try to give answers to two research questions (see 1.2). Those questions are restated and answered below.

Research Question 1:

What are the variables that can significantly influence the total travel time of a vehicle in a mission and are optimal to be used as inputs to minimize the errors of the prediction model ?

Answer: Using as input variables the features that are correlated with the response gave the lowest prediction errors and hence those variables are the optimal choice of input variables to the models. Results were similar when training with no weather variables. On the other hand when weather data were used the model's errors increased. Therefore truck related data are sufficient to provide accurate predictions.

Research Question 2

What proposed Artificial Intelligence prediction model accomplished the best performance when a benchmark comparison of the ability of the models to determine the estimated time of arrival of a truck was performed?

Answer: Even though most of the models acquired a good level of predictive accuracy the models that are more suitable to predict the arrival times of trucks are the Random Forest and Stacked Generalization models. Since Stacked Generalization model uses the prediction of the Random Forest as input to the meta predictor it achieved low errors. However the difference between the two models is not significant and the winning model can be considered to be the Random Forest algorithm,

since the Stacked Generalization model requires predictions from four different other models. The lowest RMSE error of the Random Forest algorithm was 302 seconds. In terms of Root Mean Square Error, Recurrent Neural Network showed the poorest performance (RMSE = 490 seconds) out of the prediction models but still gave better predictions than models using the average travel time as their prediction (RMSE = 508 seconds). Random Forest also displayed the lowest error in terms of MAPE error while Gradient Boosting model had an inferior performance compared to other models (MAPE = 19.03%).

6.2 Limitations

The main limitations of the study were hardware and time related. For the most part of the thesis we did not have access to a high-performance computer to run our models. This combined with the fact that some models require a lot of time to train, did not allow us to develop new models such as Convolutional Neural Network. Moreover for the same reason we were not able to explore in depth the predictive capabilities, tune the hyperparameters, of models such as Recurrent Neural Networks. Another limitation of the study is the unavailability of data related to important factors such as traffic flow information. One of the main assumptions of the study is that traffic greatly influences the arrival of trucks but unfortunately no information about the traffic situation was available to us.

6.3 Future work

Lastly we express our opinion about future projects that can potentially improve the thesis results and contribute to the development of more accurate models. Those ideas are stated in the next paragraphs.

6.3.1 Augmentation of the datasets

Collecting and processing new truck related data from GPS sensors can be very time consuming and expensive. A potential solution to this issue is to create artificial data using the current available data. Hence a study that considers data augmentation methods such as Generative Adversarial Networks is needed. Moreover it will be interesting to monitor how the prediction accuracy is influenced when using artificial data to train the models.

6.3.2 Traffic flow information

One of the limitations of this study is that no traffic flow related data were available. A study that repeats the experiments performed in the current study by including traffic flow information should be considered as a future work. Furthermore, collecting traffic flow data can be very hard and quite expensive, hence research for designing and developing models that can approximate traffic flows should be conducted.

6.3.3 Convolutional neural networks and hyper parameter tuning

In this project a plethora of prediction models were proposed and developed to estimate the arrival time of the trucks. However due to the time limitation of the thesis it was impossible to test the predictive capabilities of some prediction models that can potentially produce satisfactory or even better results than results of models proposed in this study. An interesting project that can also contribute to this study findings is to explore the predictive capabilities of other models such as convolution neural networks. In addition to that since these kind of models require a lot of computation power and large amount of time to train, Bayesian approaches can be utilised to tune the hyper parameters of the models.

Bibliography

[1] Talpac - truck haulage simulation software.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[3] Mehmet Altinkaya and Metin Zontul. Urban bus arrival time prediction: A review of computational models. *International Journal of Recent Technology and Engineering (IJRTE)*, 2(4):164–169, 2013.

[4] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[5] Yu Bin, Yang Zhongzhen, and Yao Baozhen. Bus arrival time prediction using support vector machines. *Journal of Intelligent Transportation Systems*, 10(4):151–158, 2006.

[6] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[7] Emmanuel K Chanda and Steven Gardiner. A comparative study of truck cycle time prediction methods in open-pit mining. *Engineering, construction and architectural management*, 17(5):446–460, 2010.

[8] Mei Chen, Xiaobo Liu, Jingxin Xia, and Steven I Chien. A dynamic bus-arrival time prediction model based on apc data. *Computer-Aided Civil and Infrastructure Engineering*, 19(5):364–376, 2004.

[9] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[11] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[12] The EU explained:. Transport. *European Commission*, 2014.

[13] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[14] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] Ranhee Jeong and Laurence R. Rilett. Bus arrival time prediction using artificial neural network model. In *Proceedings - 7th International IEEE Conference on Intelligent Transportation Systems, ITSC 2004*, pages 988–993, 2004.

[18] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[21] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

[22] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[23] B Mehlig. Artificial neural networks. *arXiv preprint arXiv:1901.05639*, 2019.

[24] Open Source Initiative OSI. The bsd license: Licensing. *The BSD*.

[25] Jayakrishna Patnaik, Steven Chien, and Athanassios Bladikas. Estimation of bus arrival times using apc data. *Journal of public transportation*, 7(1):1, 2004.

[26] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[27] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[28] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[29] Xiaoyu Sun, Hang Zhang, Fengliang Tian, and Lei Yang. The use of a machine learning method to predict the real-time link travel time of open-pit trucks. *Mathematical Problems in Engineering*, 2018, 2018.

[30] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. *ICML (3)*, 28(1139-1147):5, 2013.

[31] Mark J Van der Laan, Eric C Polley, and Alan E Hubbard. Super learner. *Statistical applications in genetics and molecular biology*, 6(1), 2007.

[32] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. When will you arrive? estimating travel time based on deep neural networks. AAAI, 2018.

[33] Maurice D Weir, Joel Hass, and George Brinton Thomas. *Thomas' Calculus: Multivariable*. Addison-Wesley, 2010.

[34] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[35] M Yang, C Chen, L Wang, X Yan, and L Zhou. Bus arrival time prediction using support vector machine with genetic algorithm. *Neural Network World*, 26(3):205–217, 2016.

A

Appendix 1

Proposition 1

$$\frac{\partial W_{kl}}{\partial W_{mn}} = \delta_{km}\delta_{ln}$$

where

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

is the Kronecker delta .

Derivation of the back propagation algorithm(N=1)

Updates for output layer

$$\frac{\partial H}{\partial W_{kl}^2} = -\sum_{\mu,i}(t_i^\mu - O_i^\mu)\frac{\partial O_i^\mu}{\partial W_{kl}^2}$$

$$\frac{\partial O_i^\mu}{\partial W_{kl}^2} = \frac{\partial}{\partial W_{kl}^2}(g(\sum_j W_{ij}^2 V_j^\mu - b_i^2)) = g'(B_i^{\mu,2})\delta_{ik}V_l^\mu$$

Combining both equations we obtain

$$\frac{\partial H}{\partial W_{kl}^2} = -\sum_\mu (t_k^\mu - O_k^\mu)g'(B_k^{\mu,2})V_n^\mu = -\sum_\mu \Delta_k^{\mu,2}V_n^\mu \qquad \text{(A.1)}$$

where

$$\Delta_k^{\mu,2} = (t_k^\mu - O_k^\mu)g'(B_k^{\mu,2}) \qquad \text{(A.2)}$$

Updates for first hidden layer

$$\frac{\partial H}{\partial W^1_{mn}} = -\sum_{\mu,i}(t^\mu_i - O^\mu_i)\frac{\partial O^\mu_i}{\partial W^1_{mn}}$$

$$\frac{\partial O^\mu_i}{\partial W^1_{mn}} = \frac{\partial}{\partial W^1_{mn}}(g(\sum_j W^2_{ij}V^\mu_j - b^2_i)) = g'(B^{\mu,2}_i)\sum_j W^2_{i,j}\frac{\partial V^\mu_j}{\partial W^1_{mn}}$$

$$\frac{\partial V^\mu_j}{\partial W^1_{mn}} = \frac{\partial}{\partial W^1_{mn}}g(\sum_k W^1_{jk}x^\mu_k - b^1_j) = g'(B^{1,\mu}_j)\delta_{jm}x^\mu_n$$

Combining those equations we obtain

$$\frac{\partial H}{\partial W^1_{mn}} = -\sum_{\mu,i}\Delta^{\mu,2}_i W^2_{im}g'(B^{1,\mu}_k)x^\mu_n = -\sum_\mu \Delta^{\mu,1}_i x^\mu_n \qquad \text{(A.3)}$$

where

$$\Delta^{\mu,1}_i = \sum_i \Delta^{\mu,2}_i W^2_{im}g'(B^{1,\mu}_m) \qquad \text{(A.4)}$$

This procedure can be generalized for any number of hidden layers. Similarly the updates for biases can be calculated.