Vrije Universiteit Amsterdam

Master Thesis

Predicting the ETA of cargo trains using AI models

Author:
M.G. Vaessen

Academic supervisors:
Dr. E.J. Dugundji
Prof. Dr. S. Bhulai

August 30, 2021

Je bent op tijd of je bent te laat. Als je te laat bent, moet je zorgen dat je op tijd vertrekt.

Johan Cruijff

VRIJE UNIVERSITEIT AMSTERDAM

Abstract

MSc Business Analytics

Predicting the ETA of cargo trains using AI models

by M.G. Vaessen

This thesis describes an approach for predicting the Expected Time of Arrival (ETA) of cargo trains in real time using machine learning algorithms. Moreover, the thesis presents a concept for deploying a machine learning model in production. The analysis in this thesis is based on spatial data derived from a GPS tracker inside a locomotive. The spatial data is matched to timetable data from the railway operator that operates the locomotive. A wide variety of machine learning algorithms is compared, including Linear Regression (with and without regularisation), a Multi-Layer Perceptron, Support Vector Regression, Random Forest Regression and Gradient Boosting Regression. Results show that the best performing models are the Multi-Layer Perceptron (MLP) and the Gradient Boosting Regression (GBR) model. The MLP model achieved the lowest RSME (1553.36), however, the behaviour of the model causes it not to be suitable for deployment in production. The lowest MAE (988.31) is achieved by the GBR model. Despite having a higher RSME (1775.40) than the MLP model, the GBR model's low average error and desirable behaviour make the model the best-suited model for implementation. The GBR model is deployed in an application where a customer of a railway operator can track the arrival time in real time. As the MAE of the GBR model is still considerable, the ETA is presented as an interval that shrinks as a train comes closer to its destination.

Acknowledgements

I would like to thank the following people who helped this thesis and internship become the success it was.

I want to thank my university supervisors. Dr Elenna Dugundji whose expertise was extremely helpful for pointing me in the right direction on many occasions. Moreover, I would like to express my gratitude towards Prof. Dr Sandjai Bhulai for his feedback and input.

Contents

1 Introduction **1**
- 1.1 Problem background . . . 1
- 1.2 Problem statement and context . . . 1
- 1.3 Anticipated added value . . . 2
- 1.4 Research Question . . . 3
- 1.5 Thesis outline . . . 3

2 Literature review **4**
- 2.1 Data-driven models in railway transportation systems . . . 4
- 2.2 Travel time and ETA prediction in other transportation methods 5
- 2.3 Railway travel time and ETA prediction . . . 6

3 Data Description **8**
- 3.1 Datasets . . . 8
  - 3.1.1 AVL data . . . 8
  - 3.1.2 Timetable data . . . 8
- 3.2 Data processing . . . 8
- 3.3 Data cleaning . . . 9
- 3.4 Extracting timeline for GPS data . . . 10
- 3.5 Data Matching . . . 11
- 3.6 Features Engineering . . . 12

4 Methods **14**
- 4.1 Hyperparameter tuning . . . 14
- 4.2 Performance Measure . . . 14
- 4.3 Baseline Model . . . 15
- 4.4 Linear Regression . . . 16
  - 4.4.1 Lasso Regression . . . 17
  - 4.4.2 Ridge Regression . . . 17
  - 4.4.3 Elastic Net Regression . . . 17
- 4.5 Multi-Layer Perceptron . . . 18
- 4.6 Support Vector Regression . . . 19
- 4.7 Random Forest Regression . . . 21
- 4.8 Gradient Boosting Regression . . . 22

5 Results **24**
- 5.1 Baseline Model . . . 24
- 5.2 Linear Regression . . . 24
  - 5.2.1 Lasso Regression . . . 25
  - 5.2.2 Ridge Regression . . . 25
  - 5.2.3 Elastic Net Regression . . . 26

5.3 Multi-Layer Perceptron 26
5.4 Support Vector Machine 27
5.5 Random Forest Regression 28
5.6 Gradient Boosting Regression 29

6 Implementation **30**
6.1 Implementation in UbiOps 30
6.2 Implementation in Mendix 31
6.2.1 Prediction Overview 31
6.2.2 Train Movement Overview 33

7 Discussion **34**

8 Conclusion **37**
8.1 Limitations and further research 38

List of Figures

3.1 Comparison of origin data (left) and the cleaned and interpolated data (right) projected on a map . . . 10
3.2 Example of timeline extracted from GPS data . . . 11
3.3 One-hot encoding of the feature *Interval* . . . 13

4.1 Representation of the data partitioning of three-fold Cross-Validation (Adapted from Pedregose et al. [1]) . . . 14
4.2 Architecture of a Multi-layer Perceptron (Adapted from Hassan et al. [2]) . . . 18
4.3 Two-dimensional Support Vector Machine classification (Adapted from: Yadavendra and Chand [3]) . . . 20
4.4 Soft margin approach for linear Support Vector Regression (Adapted from Schölkopf and Smola [4]) . . . 20
4.5 Architecture of a Random Forest (Adapted from Chakure [5]) . . . 21
4.6 Representation of the iterations of the Gradient Boosting Regression algorithm (Adapted from Baturynska and Martinsen [6]) . . . 22

5.1 Distribution of errors (left) and Cumulative MAE over distance (right) of the baseline model . . . 24
5.2 Distribution of errors (left) and Cumulative MAE over distance (right) of Linear Regression . . . 25
5.3 Distribution of errors (left) and Cumulative MAE over distance (right) of Lasso Regression . . . 25
5.4 Distribution of errors (left) and Cumulative MAE over distance (right) of Ridge Regression . . . 26
5.5 Distribution of errors (left) and Cumulative MAE over distance (right) of Elastic Net Regression . . . 26
5.6 Distribution of errors (left) and Cumulative MAE over distance (right) of Multi-Layer Perceptron . . . 27
5.7 Distribution of errors (left) and Cumulative MAE over distance (right) of Support Vector Regression . . . 28
5.8 Distribution of errors (left) and Cumulative MAE over distance (right) of Random Forest Regression . . . 28
5.9 Distribution of errors (left) and Cumulative MAE over distance (right) of Gradient Boosting Regression . . . 29

6.1 Representation of how a request in UbiOps works . . . 30
6.2 Start screen of the application (left) and start screen for a specific locomotive (right) . . . 31
6.3 Prediction overview of a train that is ahead of schedule (left), a train that is on time (middle) and a train that is delayed (right) 32
6.4 Screen when the train has already arrived . . . 32

6.5 Overview of train movements (left) and information of a specific movement (right) 33

List of Tables

3.1 Description of AVL data fields 8
3.2 Description of movement specific data fields 9
3.3 Description of train dimension specific data fields 9
3.4 Overview of all features used in the machine learning algorithms 12

5.1 Hyperparameters of the MLP model 27
5.2 Hyperparameters of the SVR model 27
5.3 Hyperparameters of the RFR model 28
5.4 Hyperparameters of the GBR model 29

7.1 Summary of model results 34

1 Introduction

1.1 Problem background

Trains are often used to transport cargo over great distances. In 2019, railway transportation accounted for over 17% of all in-land transportation in Europe [7]. There are a great number of factors why rail transit is a preferable option for transporting cargo. Railway transportation enables lots of goods to be transported in one shipment, which would otherwise be transported with multiple trucks. This reduces the number of human and logistic resources involved in the transportation process. Ultimately, this makes railway transportation a cost-effective transportation mode [8].

Moreover, rail freight transport is one of the most environment-friendly modes of transport. Transportation accounts for 25% of Europe's total CO_2 emissions. Of this 25%, railway transportation only constitutes 0,4% of the greenhouse gas emissions [9]. In this day and age where the impact of the high CO_2 emissions becomes more and more apparent, rail freight transportation forms a great alternative for road, marine, and short-distance aviation transportation.

One of the great pitfalls of rail freight transportation is the lack of flexibility. As trains are bound to their tracks, defective trains or poor weather conditions can lead to long delays. Contrarily, trucks or aeroplanes can swerve from their trajectory, choose different routes, and avoid potential delays. Delays cost money. Resources are allocated based on the scheduling of the railway operators. An unexpected delay can lead to resources being allocated, while the train is still hours away from its destination. This results in unnecessary costs and resources not being utilised optimally.

1.2 Problem statement and context

Crossing borders within Europe by train is paired with multiple compatibility issues. Each country used to have its own dimensions for the rail and voltage systems. Over the years the dimensions have become more and more uniform. However, there are still some discrepancies within Europe that complicate the smooth transitions between borders.

New locomotives have been developed that tackles this problem. The locomotive considered in this research is a locomotive that has a variable system configuration, i.e. components can be changed to fit country-specific railroad dimensions. These locomotives are leased to railway operators and are used for hauling cargo throughout Europe.

These locomotives are equipped with a GPS tracker. The GPS data from these trackers provides the opportunity to perform analysis on these locomotives. The trajectories the locomotive travels can be mapped accurately and the

travel time can be determined accordingly. In this research, a framework is provided that can convert GPS data into useful features, such as the distance to the next station, the speed of the train, etc. Furthermore, the GPS data is matched to timetable data from the railway operator. This way additional features can be extracted, e.g. the delay at departure and the weight and dimension of the train.

An accurate prediction of the Expected Time of Arrival (ETA) is an important asset in railway transportation. This research aims to improve the accuracy of the ETA prediction for railway operators. Different machine learning algorithms were examined to find the model that achieves the most accurate ETA prediction. The models can predict the ETA in real time, so if unexpected delays occur, the ETA will automatically be adjusted. The final model is deployed in an application that can be provided to the clients of a railway operator. In this application, a client can track the predicted ETA of the train that carries their goods.

1.3 Anticipated added value

There exist plenty of comparable models for business-to-customer deliveries. For example, when ordering a product online the ETA of the package is often displayed and most of the times very accurate. For business-to-business deliveries, these models are more scarce. This lack of accurate prediction models forms a gap in the market. Hence, an accurate indication of the ETA of cargo is highly valuable in the planning of processes.

This research can positively affect the planning of freight transportation. The railway operator that is considered in this research operates from the port of Rotterdam. When shipping freight from Rotterdam to another location in the Netherlands or across the border, a so-called *path* must be reserved. This path is reserved at ProRail and costs a certain amount of money. When this timeslot is missed, the cost for the reservation still has to be paid partially. When a train happens to run into an error, causing it to exceed the allocated time of the path, the company will be fined. Hence, inaccurate planning can lead to a lot of avoidable costs.

A freight train must be unloaded upon arrival. For this, employees must be present along with all the equipment and machinery that is needed for the unloading. An unexpected delay can result in employees being unnecessarily idle while waiting on the arrival, which comes with avoidable costs. This does not only hold for the unloading of the train, but also all the further steps in the supply chain (e.g. further transportation from the train station to the final destination).

Currently, when a train is delayed, it is communicated to the customers over a series of phone calls. The railway operator is informed by the railway manager or machinist that the train is delayed. Thereafter, the customer will be informed by the railway operator. If the customer wants an update, again a phone call needs to be made to the railway operator to enquire about the whereabouts of the train. Evidently, this approach is outdated and unnecessarily time-consuming. A simple application can greatly increase the transparency

towards the customer and redundant phone calls between the different parties involved can be avoided.

1.4 Research Question

The main research question in this thesis is:

How can machine learning algorithms be used to accurately predict the ETA of cargo trains in real time?

From a business perspective, this question inherently raises a sub-question:

How can a machine learning model that predicts the ETA of a cargo train be deployed in production?

1.5 Thesis outline

Section 2 presents a review of relevant literature on the topic of ETA prediction in railway transportation. Furthermore, in this section, research on ETA prediction for other modes of transportation is summarised, as well as other implementations of data-driven models in railway transportation systems.

In section 3, the data that was used for this research is described. The section explains the process of transforming data into useful features for the machine learning models. Section 4 focuses on the different machine learning models that were implemented.

The results of the models are presented in section 5. In section 6, a description of the application, where the machine learning model is deployed, is given. In section 7, the results of the machine learning models are discussed. Finally, section 8 offers a summary of the research and provides some limitations and recommendations for further research.

2 Literature review

2.1 Data-driven models in railway transportation systems

Railway transport, among many other industries, is undergoing a digital transformation. With this digital transformation comes a wide variety of opportunities to innovate, possibilities to improve the business processes, and chances to create new insights using data analytics. This phase of digitalisation in the business world is referred to as "Industry 4.0". The main goal of Industry 4.0 is to fundamentally change the traditional processes in industries into smart solutions [10]. The three major fields of railway transportation can be classified as maintenance, safety, and operations [11]. Each of these fields can benefit greatly from the smart solutions resulting from the fourth industrial revolution.

Predictive models are used in decision support systems (DSS) to provide timely, preventive maintenance. Preventive maintenance can increase the sustainability of railway transportation systems [12]. Roberto Nappi has found that condition-based maintenance models, when used correctly, represent an element of absolute efficiency improvement compared to the use of the only time-based and failure-driven strategies [13]. Yang et al. provide a decision support framework for the maintenance of the signalling systems of trains [14]. Furthermore, a DSS approach can be used to monitor railway track maintenance and renewal [15], scheduling the maintenance of rolling stock [16], and conducting risk assessments based on the condition of the infrastructure [17].

The safety and maintenance of railway systems go hand in hand. Recognising defects early on, performing preventive maintenance, and renewing parts that are beyond repair will increase the safety of railway transportation significantly. However, railway safety is much broader than avoiding and repairing defective parts. Most research on railway safety concerns statistical analysis on accidents to gain an understanding of the cause, frequency, severity, and the contributing safety factors related to infrastructure, operations, or environment [11]. As the dataset on railway accidents is relatively small, predictive modelling in this field is difficult. However, there exist some research on predictive models for railway safety. Yilboga et al. use an Artificial Neural Network (ANN) to predict the failure of a railway turnout system [18]. Hu and Liu propose a technique for modelling track geometry degradation using a Support Vector Machine (SVM) [19]. An automated diagnosis network using a deep belief network (DBN) is described by Yin and Zhao, which can be used to predict the reason for a failure [20].

Railway operations include scheduling, signalling, rail traffic management, and much more. In this field, smart solutions can be used to reduce costs, reduce the number of delays, predict the travel time and estimate the time of arrival. Kamburjan et al. present a comprehensive model of railway operations.

This model uses a uniform approach to modelling that enables the global effects of local changes to be analysed [21]. Delay prediction is an important factor in train timetabling and dispatching. Wang and Zhang propose a gradient-boosted regression trees model based on historical and weather data to predict the duration of delays [22]. Mou et al. compared a random forest, an ANN model, and a Long Short-Term Memory (LSTM) model for predicting delay duration and found that the LSTM model outperformed the rest [23]. Comparable models were used for predicting the ETA and travel time of trains. Section 2.3 gives an in-depth overview of research on this subject.

2.2 Travel time and ETA prediction in other transportation methods

Accurate travel time and ETA prediction are valuable assets in timetabling and planning processes in all forms of transportation. Each transportation method raises its own issues when predicting travel time. Busses travel on public roads. Although busses often use a separate bus lane, they are still affected by high traffic density, accidents, traffic jams, and other obstructions on the road. Historical average models, as opposed by Jeong and Rilett [24] and Vanajakshi and Rilett [25], tend to only perform well when traffic flow is small and stable. More complex traffic flows ask for more sophisticated models. Artificial Neural Networks (ANN) are widely used in literature. Gurmu and Fan show that an ANN outperforms historical average models in 70% of all cases [26]. Amita et al. compared an ANN to a linear regression model and found that the ANN performed better [27]. Despite that, Maiti et al. found that an ANN only outperforms a historical data-based model by a neglectable percentage, while the historical model is approximately two-and-a-half times faster than the ANN, in terms of computation time [28].

In aviation, most models used by airlines to predict the travel time and ETA are a combination of parametric models, physical models, and aeroplane performance models [29]. These models are unable to capture external factors such as the weather and high traffic density. Similar to railway cargo transportation, although state-of-the-art models do exist, airlines still rely on simple models due to insufficient data and real-time data integration[30]. Literature shows that tree-based regression models tend to perform well on this matter. Kern et al. showed that a Random Forest Regression (RFR) model outperforms the traditional models used by airlines [31]. Glina et al. compared multiple regression tree ensemble models and found that the Quantile Regression Forest was the most promising model [32].

When transporting cargo from A to B, freight is often shipped in intermodal containers, i.e. it is shipped using multiple modes of transportation. This raises a challenge because research has shown that each form of transportation has its difficulties. As a result, no single model performs best for all modes of transportation. Balster et al. argue that the best approach for ETA prediction for intermodal transportation is to develop an individual machine learning model

for each leg of the supply chain [33]. Servos et al. did attempt to use a single model to predict the ETA of intermodal transportation and found that a Support Vector Regression (SVR) model performs the best [34].

2.3 Railway travel time and ETA prediction

Timetables for trains are developed using mathematical optimisation. The basis of the Dutch railway timetables is the Basis Uur Patroon (basic hourly pattern), i.e. the timetable repeats every 60 minutes. This basis is adjusted with extra trains in peak hours and fewer trains at night. This hourly timetable has to fulfil certain constraints. For each section of the network (from station A to station B) there is a minimum and a maximum time the train can take. Conflicting schedules have to be prevented, only one train can travel along a certain track at a certain time. Taking all the constraints into account, the timetable is optimised using CPLEX [35; 36].

In railway transportation, we can distinguish between passenger and freight trains. Most research in the field of rail transit is done on passenger trains. Throughout literature, the predominant approach to predict the ETA of freight trains is to predict delays. As the scheduled departure and arrival time are fixed, it can be argued that the actual travel time is equal to the travel time corresponding to the timetable plus a contingent delay [37; 38].

The models that were used in past research are similar to those we have seen in other modes of transportation. The most promising models for ETA (and delay) prediction for railway transportation are ANNs, tree-based models, and SVMs. Artificial Neural Networks tend to work as a "black box". ANNs have great predictive power, however, they are very hard to understand. As a result, the choice of which model to use is not purely based on the performance, but also on how interpretable the results are. Yaghini et al. found that a neural network performs best for delay prediction compared to decision tree and multinomial logistic regression models [39]. Prokhorchenko et al. also concluded an ANN to be the best performing model, compared to a linear regression model, a ridge regression model, and a Bayesian ridge regression model [40]. Hu and Noche found that although an ANN performs very well, it can be approved by using a combination of Genetic Algorithms (GA) and a Back Propagation Neural Network (BPNN), resulting in a GA-BPNN model [41].

On the other hand, Babour et al. found that a random forest model outperformed five other models. The five models the random forest were compared to were three different versions of Support Vector Regression models, a deep neural network model, and a statistical model. The random forest improved the predictive accuracy by 60% and was found to be helpful for freight rail operational decision making.[42] Li et al. came to a similar conclusion when comparing a random forest model to an ANN, an Extreme Gradient Boosting model, a Gradient Boosting Decision Tree model and a statistical average model. It emerged that the random forest model achieves high accuracy while the training is low, making it well suited for delay and travel time prediction [43].

Babour et al. compared different linear and non-linear Support Vector Regression algorithms to create an individual model for predicting the ETA of

each origin-destination pair freight railway network. They achieved an average improvement of 14% over a historical average baseline model [44]. Markovic et al. showed that statistical comparison of the generalisation power of a Support Vector Regression (SVR) model and an ANN model indicated that the SVR model performed better [45].

All things considered, there is not a single model that can be considered the best for predicting the ETA or delays for railway transport. ANN models perform satisfactorily throughout the literature. However, there is a good deal of research that shows that other models, such as SVR and Random Forest models, outperform ANN models in some instances. The only way to find out which model performs best for a particular problem is to deploy multiple suitable models and compare the results. Also, the computation time and the interpretability should be considered when choosing a suitable model.

3 Data Description

3.1 Datasets

There are two different types of datasets used in this research: Automated Vehicle Location (AVL) data and timetable data. Both datasets consist of four months of data from the months of January to April 2021.

3.1.1 AVL data

The AVL dataset contains the GPS data of one locomotive that travels between the Port of Rotterdam and Germany. A GPS tracker inside of the locomotive sends the GPS location to a database every second. Also, the velocity of the locomotive is captured every minute. However, these data points are omitted as they are in a different interval from the GPS data. The data fields contained in the AVL data are described in Table 3.1.

Data field	Description
Value name	Type of the data point (Longitude, Latitude or Speed)
Value	Decimal value of the data point
Timestamp	Unix epoch timestamp of data point

TABLE 3.1: Description of AVL data fields

3.1.2 Timetable data

The timetable data is exported from RailCube, a tool that is used by a large portion of the railway freight forwarders. RailCube can be used for visualising data, making rosters, and monitoring processes. The data contains information of each planned locomotive movement corresponding to the locomotive consistent with the source of the AVL data. Tables 3.2 and 3.3 describe the data fields that were used for analysis corresponding to the movement of the train and the dimensions of the train, respectively.

3.2 Data processing

The AVL data is processed into time-series data, in which the GPS locations and timestamps are contained in a one-second interval. The distance between two points is calculated using the haversine formula[1]. This formula computes

[1]Haversine formula: https://www.en.wikipedia.org/wiki/Haversine_formula

Data field	Description
OriginLocationName	Name of the origin station
OriginLocationLatitude	Latitude of origin station
OriginLocationLongitude	Longitude of origin station
DestinationLocationName	Name of the destination station
DestinationLocationLatitude	Latitude of destination station
DestinationLocationLongitude	Longitude of destination station
RevisedDeparture	Definitive planned departure time
RevisedArrival	Definitive planned arrival time
ActualDeparture	Time the train actually departed
ActualArrival	Time the train actually arrived

TABLE 3.2: Description of movement specific data fields

Data field	Description
WagonCount	Number of wagons
WagonLength	Length of all wagon combined
TrainWeightGross	Weight of the train, including locomotive and freight

TABLE 3.3: Description of train dimension specific data fields

the distance between two GPS locations, taking into account the curvature of the earth. The haversine formula is defined as followed:

$$haversine\left(\frac{d}{r}\right) = haversine(\phi_2 - \phi_1) + \cos(\phi_1)\cos(\phi_2)haversine(\lambda_2 - \lambda_1),$$

where d is the distance between two points, r is the radius of the earth (6.371 km), ϕ_1,ϕ_2 are the latitudes of two points, and λ_1,λ_2 are the longitudes of the two points. We can solve for d as all other variables are known, which results in the distance between two GPS points in kilometres. This value can be multiplied by 1000 to compute the distance in metres.

Subsequently, the average velocity between two data points can be calculated by dividing the distance between two adjacent points in the data, by the time in seconds between the points. The velocity can be used to determine whether a train is idle or moving. There is some noise in the data (minor inaccuracies in the GPS location), which is visible when a train is idle. The velocity fluctuates from zero metres per second to a few decimetres per second. Therefore, a train is considered to be idle when the velocity is less than $0.5m/s$.

3.3 Data cleaning

As previously mentioned, the GPS data contains some irregularities. Apart from the minor fluctuations when the train is idle, there are some undoubtedly incorrect GPS locations within the AVL data. This becomes apparent in two particular cases. The first case is when the train is idle for a while and suddenly the velocity increases drastically and drops back to zero. A train accelerates

gradually, so a big increase like that is physically impossible. The second case is when the train is driving and suddenly "stops" for a few seconds and continues driving at the same velocity afterwards. To battle this data issue, short stops and short drives are deleted from the data. Short stops and rides are considered to be those that are less than five seconds and are contained in the opposite state. To be more precise, a short stop is a stop of fewer than five seconds while the train is driving. On the other hand, a short ride is a ride of fewer than five seconds while the train is stopped.

Apart from gaps in the data that appear after deleting short stops and rides, there are additional gaps as a result of the GPS tracker losing signal (in tunnels or areas with bad reception). Both types of gaps are filled using linear interpolation. For each second between the last point before a gap and the first after a gap the latitude and longitude can be estimated using the following function for both:

$$y(t) = x_0 + \frac{(x_1 - x_0)}{n}t,$$

where $y(t)$ is the computed latitude/longitude t after the last timestamp before the gap, x_0 is the latitude/longitude of the last data point before the gap and x_1 is the latitude/longitude of the first data point after the gap. Figure 3.1 shows a comparison of the original data and the cleaned and interpolated data.

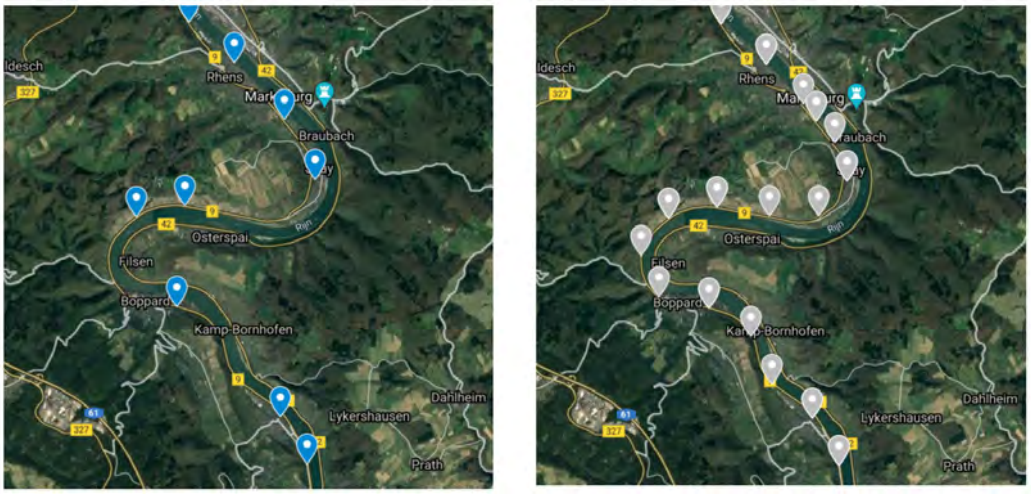

FIGURE 3.1: Comparison of origin data (left) and the cleaned and interpolated data (right) projected on a map

3.4 Extracting timeline for GPS data

After having cleaned the data, a timeline can be computed from the GPS data. For each station the locomotive can pass or stop at, the GPS locations are available. With these GPS locations and the current GPS location of the locomotive, the distance of the locomotive to each station in the system can be determined using the haversine formula. By taking the closest station to the locomotive at

each timestamp, an initial timeline can be established. However, a station should only be considered to be part of the timeline, when the train either passes the station or stops at the station.

To determine whether a train has stopped at a station, an imaginary circle with a radius of 2,000 metres is drawn around the centre of each station. If a locomotive stands still in one of those circles, the train is considered to have stopped at the corresponding station. The radius of the circles is rather large, because, especially for cargo trains, trains can stop at different terminals of a station. To cover all the possible terminals the train can stop at, 2,000 metres was found to be a reasonable radius.

FIGURE 3.2: Example of timeline extracted from GPS data

A train does not stop at each station. Some stations the train just passes through. Nonetheless, these stations should still be considered as part of the route. The approach for determining the stations the train passes through is similar to the approach for determining whether the train is stopped. Again, an imaginary circle is drawn, this time with a radius of 200 metres. If a train comes within this circle (and does not stop) the train is considered to have passed the station.

Finally, the exact timeline of the train can be extracted as shown in Figure 3.2. For each station where the train has stopped, the arrival and departure time can be determined. Furthermore, the exact time a train passes a station can be computed by taking the timestamp for which the distance between the GPS location of the locomotive and the station was minimal. The timeline is used in the next step, where the train movements in the timetable data are matched to the timeline found in the GPS data.

3.5 Data Matching

As the timetable data contains important information, only the sequences in the timeline that can be matched to the timetable data will be considered in further research. The two datasets are matched using a matching algorithm. This algorithm loops through all train movements in the timetable data and looks for the movement in the timeline. The search space is based on the actual departure and arrival times provided in the timetable data. However, these times turned out to be inaccurate in many instances. Therefore, the search space was extended by six hours on both sides. The interval was chosen so that the highest number of movements could be matched, without matching the wrong movements to the timetable data. If the recorded actual arrival times were more accurate, a smaller interval should be used. For each movement in the timetable data, a subset of the timeline (from the actual departure time - six hours to the actual arrival time + six hours) is searched. If the movement is found, the movement is used for further analysis. If not, the movement is discarded from the analysis.

3.6 Features Engineering

After having matched the movements, additional information for each movement can be computed. The actual arrival and departure times can be corrected, using the timestamps found in the timeline. Using these corrected times, the total travel time from origin to destination for each movement can be determined. The total distance covered during a movement can be calculated by taking the sum of all distances between adjacent points in the AVL data within one movement.

Feature	**Description**
Distance_Travelled	Distance the train has travelled since the departure
Distance_Remaining	Distance remaining to the destination
Distance_Percentage	Proportion of the total trip that has elapsed
Time_travelled	Total time elapsed since the departure
Remaining_Traveltime_Planned	Remaining time to the destination corresponding to the schedule
Current_Stop_Length	Amount of seconds the train has been standing still
Interval	Time interval the current timestamp falls in (Night, Morning, Afternoon, Evening)
Weekday	The weekday the current timestamp is on (Monday - Sunday)
Movement	Unique combination of origin and destination of movement
Speed	Current speed of the train
Status	Status of the train (1 is driving, 0 is idle)
Weight	Total gross weight of the locomotive, freight and wagons
Length	Total length of the train
WagonCount	Number of wagons the locomotive pulls
Target	**Description**
Remaining_Traveltime	Remaining time until the train reaches the destination

Table 3.4: Overview of all features used in the machine learning algorithms

The distances are used to compute the three distance-related features: *Distance_Travelled*, *Distance_Remaining* and *Distance_Percentage*. *Distance_Travelled* is the distance the train has travelled since departure from the origin. *Distance_Remaining* is the distance the train has yet to travel to the destination. *Distance_Percentage* is the proportion of the total distance of the train movement that the train has already covered (= distance travelled / total distance).

The timestamps are not used as a feature as they are, however, they are used to compute the following four features: *Remaining_ Traveltime*, *Time_ Travelled*, *Remaining_ Traveltime_ Planned*, and *Current_ Stop_ Length*. *Remaining_ Traveltime* is the remaining time to the destination of the current movement. This feature is the target feature, the value we want to predict with the machine learning models. *Time_ Travelled* is the total time elapsed since the departure from the origin station. *Remaining_ Traveltime_ Planned* is the remaining time to the destination corresponding to the schedule (= total scheduled travel time - time travelled). *Current_ Stop_ Length* is the number of seconds the train has been standing still for (= 0 if the train is driving).

Additionally, there are three categorical features: *Interval*, *Weekday*, and *Movement*. The feature *Interval* defines in which time interval the current timestamp falls. Four intervals are defined: Night [00:00 - 06:00], Morning [06:00 - 12:00], Afternoon [12:00 - 18:00], and Evening [18:00 - 00:00]. Each timestamp is mapped to the corresponding time interval. Similarly, the feature *Weekday* is computed by mapping the timestamps to the corresponding weekday. *Movement* defines the origin and destination station of the movement. As the machine learning models that are used are only able to process numeric features, the categorical features are transformed using one-hot encoding. One-hot encoding transforms a categorical vector into multiple binary columns. Each column represents one of the categories. The value in the columns is 1 if the feature is equal to the corresponding category, and 0 if not. Figure 3.3 shows how the one-hot encoding looks for the feature *Interval*. Finally, the previously explained features *Speed*, *Status*, *Weight*, *Length* and *WagonCount* are used.

Interval
Night
Morning
Afternoon
Evening

Night	Morning	Afternoon	Evening
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

FIGURE 3.3: One-hot encoding of the feature *Interval*

An overview of all features is displayed in Table 3.4. The final dataset contains the features in a one-second interval between the departure and arrival times of all matched movements. This dataset is rather large with over 1,5 million data points. Training a machine learning model on a dataset of this size takes a lot of computation time. It turned out that reducing the size of the dataset to a one-minute interval had almost no impact on the performance of the models. However, increasing the interval size decreases the computation time of the model significantly. Therefore, the choice was made to proceed with the data in a one-minute interval.

4 Methods

The models in this section were implemented in Python using the Scikit-learn library [1]. This library features multiple well-known machine learning algorithms and provides an easy-to-use framework to train and deploy models.

4.1 Hyperparameter tuning

The hyperparameters of the models are optimised using a three-fold Cross-Validation Grid-Search[1] approach. This approach performs an exhaustive search over a specified parameter search space. The three-fold Cross-Validation splits the training data into three equally sized subsets (folds) as illustrated in Figure 4.1. For each parameter combination in the search space, the model is trained on three splits. In each split, one of the folds is used as test data and the other two folds are used to train the model. Each split outputs a score, the MAE (section 4.2) of the model based on the training fold. The parameter combination that results in the lowest average score over all three splits, is found to be the optimal hyperparameters for the model.

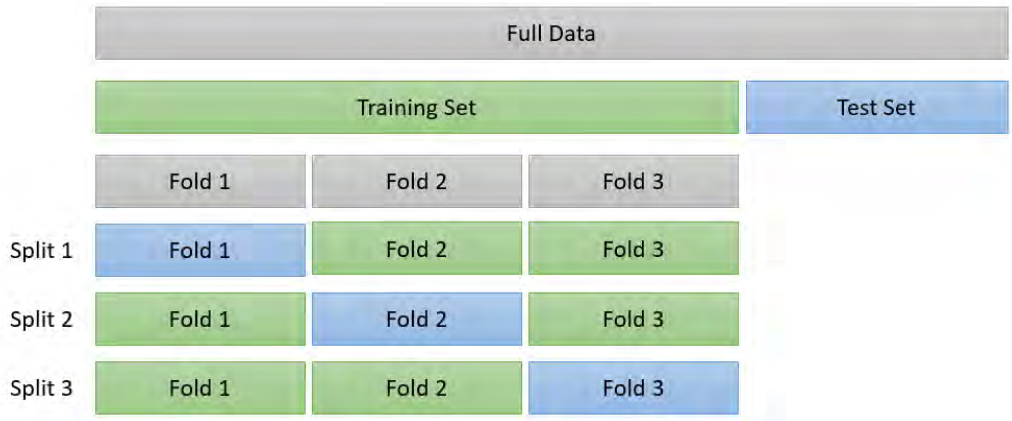

FIGURE 4.1: Representation of the data partitioning of three-fold Cross-Validation (Adapted from Pedregosa et al. [1])

4.2 Performance Measure

All the models considered are evaluated and compared on two metrics: the Root Means Squared Error (RMSE) and the Mean Absolute Error (MAE). The

[1] GridsearchCV: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV

metrics are computed using the following formulas:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

$$MAE = \frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|}{n}$$

Where n is the number of sample in the evaluated dataset, y_i is the actual target value, i.e. the remaining travel time and $\hat{y}_i$ is the remaining travel time predicted by the machine learning models.

Both metrics are commonly used to measure the accuracy of a model that predict continuous variables. The MAE measures the average magnitude of the prediction error, neglecting whether the error is positive or negative. The MAE can be interpreted as the average deviation between the prediction of the model and the target value. In this particular case, the MAE defines how many seconds the predicted arrival time of the train differs from the actual arrival time. For example, an MAE of 1.200 means the difference between the predicted and actual arrival time is 20 minutes on average.

The RMSE also measures the average magnitude of the prediction error. This metric is the square root of the average squared prediction error. As the error is squared, high errors have a bigger impact on the outcome of the metric. In other words, the RMSE metric penalises higher errors. The RMSE is always higher than the MAE. When comparing two models, the MAE of model 1 may be higher than model 2 while the RMSE of model 1 is lower than model 2. This means that on average model 1 has a lower error, but that there are bigger high outliers in the errors of the first model.

4.3 Baseline Model

The original schedule functions as a simple baseline model to compare the machine learning models with. The total scheduled travel time can be deducted from the timetable data (difference in seconds between planned departure and planned arrival). The predicted remaining travel time ($\hat{y}_i$) is equal to the total scheduled travel time minus the time that has elapsed since the train departed. For example, a train that was scheduled to depart at 1 pm and arrive at 3 pm has a total scheduled travel time of two hours. If the train were to depart at 1:30 pm, the prediction at 2 pm of the baseline model would be:

$$\begin{aligned}\hat{y}_i &= \text{total scheduled travel time} - \text{elapsed time since departure} \\ &= 2 \text{ hours} - 0.5 \text{ hour} \\ &= 1.5 \text{ hours.}\end{aligned}$$

4.4 Linear Regression

Linear regression is used to estimate the relation between one or more independent variables and one dependent variable. As multiple independent variables are considered in this research, multiple linear regression (MLR) is used. When applying MLR a linear function of the form

$$y = \beta_0 + \beta_1 X_1 + ... + \beta_n X_n + \epsilon$$

is fitted to the data, where

- y is the predicted value of the dependent variable.
- β_0 is the intercept of the function, i.e. the value of y if all values in X are equal to 0.
- β_i is the weight in the function of variable i.
- X_i is the value of independent variable i.
- n is the number of independent variables.
- ϵ is the error of the estimation, i.e. the difference between the predicted value and the actual value of y.

The goal of MLR is to minimise the error ϵ of the estimate. This is achieved by the ordinary least squares (OLS) approach. This approach minimises the squared sum of the errors to find the best set of values for β. In other words, the loss function that is minimised is

$$L(\beta) = \sum_{i=1}^{n} \epsilon_i^2 = \epsilon'\epsilon = (y - X\beta)'(y - X\beta),$$

which can be rewritten to

$$L(\beta) = y'y + \beta' X' X\beta - 2\beta' X' y.$$

To find the minimum of this function we have to find the point for which the derivative of $S(\beta)$ is equal to zero. We have,

$$\frac{\delta L(\beta)}{\delta \beta} = 2X'X\beta - 2X'y$$

$$\frac{\delta L(\beta)}{\delta \beta} = 0 \Rightarrow X'X\hat{\beta} = X'y \Rightarrow \hat{\beta} = (X'X)^{-1}X'y.$$

Hence, $\hat{\beta}$ minimises the sum of squared errors in the MLR approach. MLR can be prone to overfitting. This means that the model performs very good on the training set, but performs poor on the test set or new data. This is a result of the model giving too much weight to noise (random outliers) in the training data. Regularisation can be used to avoid overfitting. Regularisation forces the weights in the function towards zero, in other words, it reduces the complexity

of the model. In the following three sections three forms of regularisation for linear regression are introduced: lasso, ridge, and elastic net regularisation.

4.4.1 Lasso Regression

Least Absolute Shrinkage and Selection Operator (Lasso) adds a penalty for non-zero weight. Lasso penalises the sum of the absolute values of the weights. This is also known as an L1 penalty. With Lasso regularisation, the loss function is defined as

$$L(\beta) = \sum_{i=1}^{n}(y_i - x_i'\beta)^2 + \lambda \sum_{j=1}^{m} |\beta_j|.$$

The variable λ regulates the amount of shrinkage of the weights in β. When λ is equal to zero lasso regression is the same as OLS linear regression as explained in the previous section. A high value for β forces more weight to be zero. This eliminates variables from the model and results in a sparse model with few weights. Therefore, Lasso can also be used for feature selection.

4.4.2 Ridge Regression

Similar to lasso regression, ridge regression adds a penalty for non-zero weights. However, Ridge regression penalises the squared sum of the weights, called the L2 penalty. The loss function for Ridge regression is

$$L(\beta) = \sum_{i=1}^{n}(y_i - x_i'\beta)^2 + \lambda \sum_{j=1}^{m} \beta_j^2.$$

Again the penalty variable defines the amount of regularisation in the model. The difference between Lasso and Ridge is that Lasso can force some weights to be equal to zero, while Ridge forces weights towards zero but not equal to zero. The L2 penalty increases exponentially, so weights far away from zero result in a very high penalty. As a result, when minimising the loss function the weights are chosen so that they are close to zero. This means that some variables have a very low impact on the model, but they are still all used.

4.4.3 Elastic Net Regression

Elastic Net Regression combines both the L1 and L2 penalty. This form of regularisation combines the best of both worlds of Lasso and Ridge regression. Elastic Net regression minimises the following loss function:

$$L(\beta) = \sum_{i=1}^{n} \frac{(y_i - x_i'\beta)^2}{2n} + \lambda(\frac{1-\alpha}{2} \sum_{j=1}^{m} \beta_j^2 + \alpha \sum_{j=1}^{m} |\beta_j|),$$

where $\alpha \in [0, 1]$ is the trade-of variable between the L1 and L2 penalty. When $\alpha = 0$ Elastic Net is similar to Ridge regression, while for $\alpha = 1$ it is similar to Lasso regression. The λ for all three regularisation methods and the α of Elastic Net need to be optimised. There is no "smart" way of doing this, it is a

matter of testing different values and finding the values that work best for the given problem.

4.5 Multi-Layer Perceptron

Artificial Neural Networks are inspired by the workings of the human brain. These networks emulate how electrical activity moves through the brain and nervous system [46]. The most commonly used ANN is a feed-forward multi-layer perceptron (MLP). An MLP consists of three types of layers: an input layer, one or multiple hidden layers, and an output layer. Each layer embodies several nodes, which are connected to nodes in adjacent layers. Figure 4.2 illustrates the architecture of an MLP with a single hidden layer.

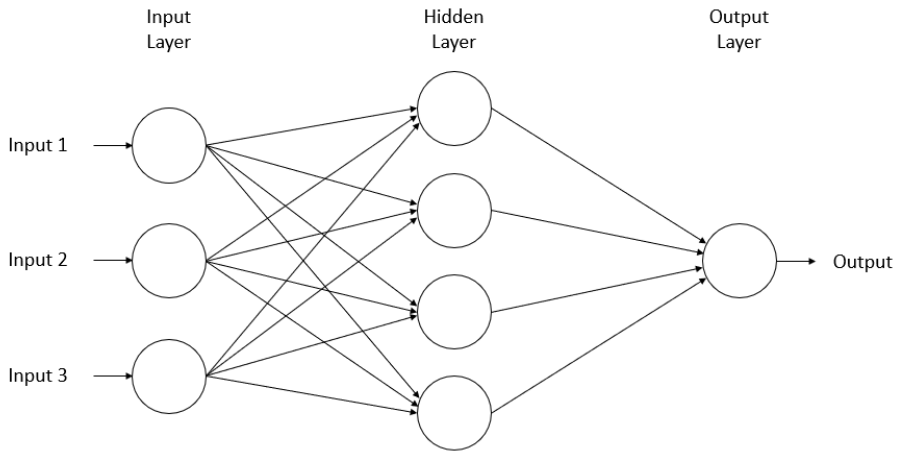

FIGURE 4.2: Architecture of a Multi-layer Perceptron (Adapted from Hassan et al. [2])

An MLP is a feed-forward network, i.e. information moves forward through the network from the input nodes to the output nodes. Each node in the network has an activation function. This is mostly a non-linear function that is applied to the sum of all inputs to the particular node. This great mixture of non-linear functions enables an MLP to approximate extremely non-linear relations in data [47]. The nodes in the network are connected by numerical weights. The output of the activation function of a node is scaled by the corresponding weight when it is fed forward to the next node.

An MLP is trained using the back-propagation algorithm. This algorithm iteratively adjusts the weights in the network to find the optimal set of weights. For a set of weights, the network can predict the target variable, given a set of input values. The difference between the output of the MLP and the actual target value is the error of the model, which needs to be minimised. This error is minimised using gradient descent. Gradient descent is an algorithm that is used for finding a local minimum of a differentiable function. The idea behind the algorithm is to move in the opposite direction of the gradient of a function, i.e. follow the direction of the steepest descent. Repeating this process will result in finding the local minimum of a function. In the case of back-propagation,

this function is the loss function (the error). The back-propagation algorithm consists of the following steps [47]:

1. Initialise the weights
2. Randomly choose input and the corresponding target value
3. Propagate the input through the network to obtain the output value
4. Calculate the error by comparing the actual output to the target output
5. Propagate the error back through the network
6. Adjust the weights to minimise the error (gradient descent)
7. Repeat steps 2-7 until the error is satisfactorily small

There are multiple hyperparameters an MLP is subject to when implemented in scikit-learn, that need to be tuned to achieve the best performance. The *activation function*, the function that maps the weighted inputs to the output of a neuron is, as mentioned before, a non-linear function most of the times. The most commonly used activation functions are the logistic function, the sigmoid function, and the rectifier linear unit (ReLU) function. The *learning rate* is a numerical variable that regulates how fast the gradient descent algorithm converges. Furthermore, the *number of hidden layers* and the *number of nodes* in the hidden layers can be adjusted. Finally, an L2 regularisation penalty can be added to the model. The amount of L2 penalisation is regulated by the parameter *alpha*.

4.6 Support Vector Regression

Support Vector Machines are frequently used for classification problems. The idea behind an SVM is to create a line or hyperplane between two separable classes within data. This separation is chosen so that the margin between the separation, and the point in each class that is closest to the separation, is maximised. The closest points in each class are called the support vectors. For a two-dimensional feature space, this easily visualised as seen in Figure 4.3, where the separation is a line. For high-dimension feature spaces, the problem becomes more complex.

Support Vector Regression follows the same idea as its classification counterpart. Instead of fitting a hyperplane, a function is fitted to the data. This approach is also known as ε - SV regression. Smola and Schölkopf provided a good explanation of how a Support Vector Regression model works [48]. The goal is to find a function $f(x)$, with x being a vector with input features, so that $f(x)$ deviates at most ε from the actual target values, and that the function is as flat as possible. This means that the error of function is ignored as long as it does not exceed the given ε. This enables the possibility to choose a maximum error that is allowed for the model.

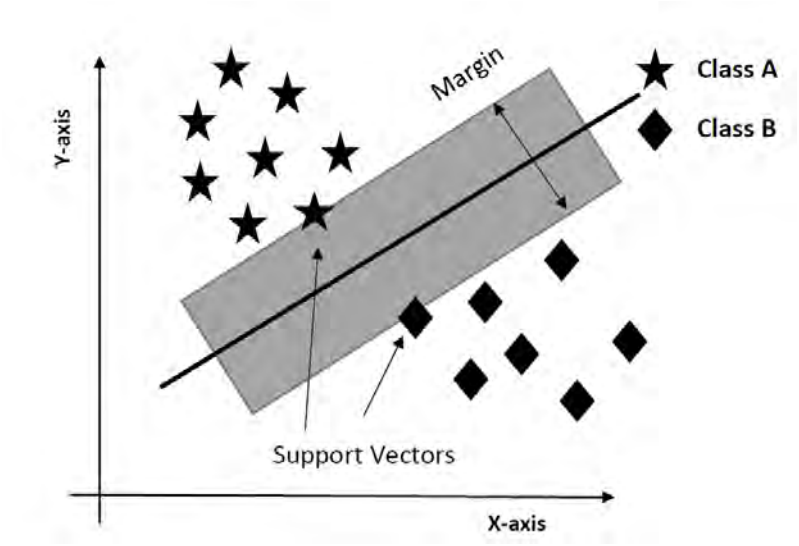

FIGURE 4.3: Two-dimensional Support Vector Machine classification (Adapted from: Yadavendra and Chand [3])

If the function $f(x)$ is linear, it has the form $f(x) = \langle w, x \rangle + b$, where w has the same dimension as the input vector x and $b \in R$. The smaller the values of w, the more *flat* the function is. Therefore, we want to minimise the norm of w. This leads to the following optimisation problem:

$$\text{minimise} \qquad \tfrac{1}{2}||w||_2$$

$$\text{subject to} \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon \end{cases}$$

If there exists a function f that approximates the target value y_i of all input vectors x_i with precision ε, the problem is feasible and the solution is found by solving the problem above. However, in some cases, a feasible solution can not be found. This problem can be rectified by introducing some slack to the model. This approach is known as the "soft margin" approach and is visualised in Figure 4.4. The variables ξ and ξ^* define the amount of slack in the model, i.e. the amount that the actual error exceeds the maximum error ε.

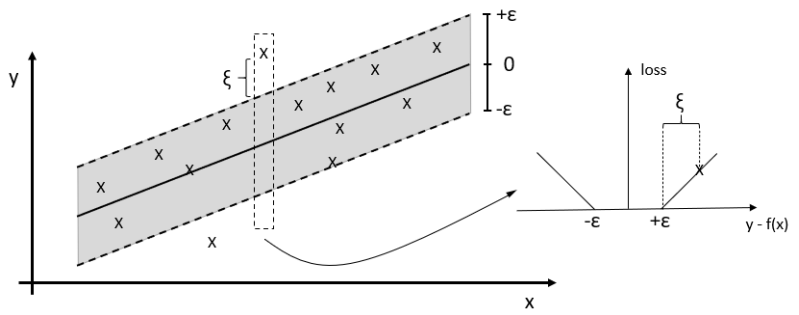

FIGURE 4.4: Soft margin approach for linear Support Vector Regression (Adapted from Schölkopf and Smola [4])

This leads to a new formulation of the optimisation problem:

$$\text{minimise} \qquad \tfrac{1}{2}||w||_2 + C\sum_{i=1}^{l}(\xi + \xi^*)$$

$$\text{subject to} \qquad \begin{cases} \mathrm{y}_i - \langle w, x_i\rangle - b \leq \varepsilon + \xi \\ \langle w, x_i\rangle + b - y_i \leq \varepsilon + \xi^* \\ \xi, \xi^* \qquad\qquad\quad \geq 0 \end{cases}$$

The hyperparameters of the SVR model are the kernel type, and the variables C and ε. The kernel type defines the form of the function f. In this example f is linear, but f can also be polynomial, a radial basis function, a sigmoid function, or any other viable function. The variable ε defines the amount of prediction error the model allows. The variable C regulates the trade-off between the flatness of the function f and the extend of the slack in the model. A higher C would mean a more flat function with less slack.

4.7 Random Forest Regression

Random Forest models can be used for both classification and regression problems. In this case, we consider a Random Forest Regression model as the aim is to predict a continuous variable, i.e. the travel time of a cargo train. A random forest is an assembly technique that combines the prediction of multiple machine learning algorithms, i.e. multiple decision trees. The prediction results of all the trees in the model are averaged to produce one single output value. The structure of a random forest is shown in Figure 4.5. Note that in this example the forest consists of 100 trees, yet in practice, the number of trees can be any reasonable number.

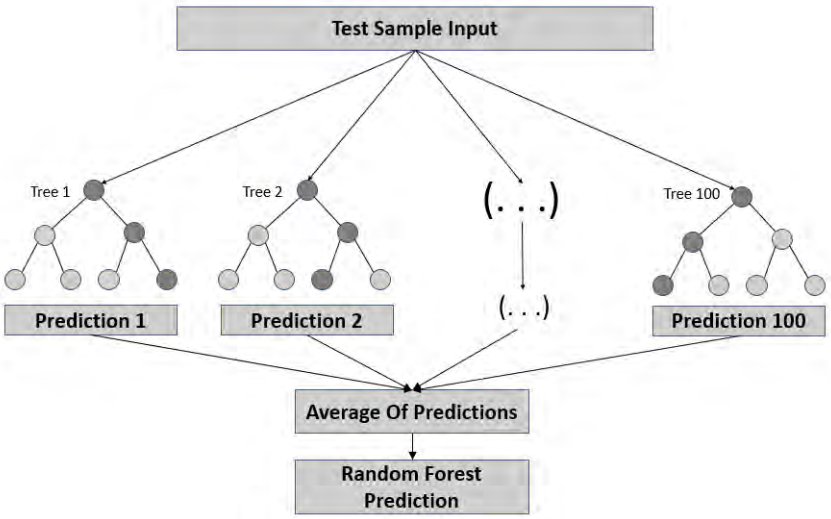

FIGURE 4.5: Architecture of a Random Forest (Adapted from Chakure [5])

A random forest model makes use of a technique called bootstrap aggregating, also known as bagging. This technique fits a decision tree on a bootstrap

sample rather than on the original data sample. The trees in the forest run in parallel, so there is no interaction between the trees. Decision trees are commonly prone to overfitting. As each tree draws a random sample from the original data, this added randomness prevents overfitting. In fact, the Strong Law of Large Numbers shows that random forests always converge, so overfitting is not a problem [49].

The most important hyperparameters of a random forest are *n_estimators, max_features, max_depth, min_samples_split*, and *min_samples_leaf*. The parameter *n_estimators* defines how many trees are built in the random forest. A random forest model resamples the features before deciding the best split. The parameter *max_feature* defines how many features to resample. The maximum depth of a tree in the forest is defined by *max_depth*. *Min_sample_split* is the minimum number of samples needed to split a node. Lastly, the parameter *min_samples_leaf* defines the minimum number of samples required for each leaf in a tree.

4.8 Gradient Boosting Regression

Gradient Boosting Regression (GBR) in an ensemble model, which means that various simple individual models are combined to generate one powerful model. GBR has recently gained more and more attention for being a fast and accurate machine learning model for large and complex data sets. As the name of the model shows, GBR uses boosting to create the ensemble. Boosting is a technique where first an initial model is fitted to the data. Thereafter, a second model will focus on improving the accuracy of the model for the instances where the initial model performs poorly. The process of boosting transforms a weak predictor into a strong predictor. The boosting procedure of GBR is illustrated in Figure 4.6.

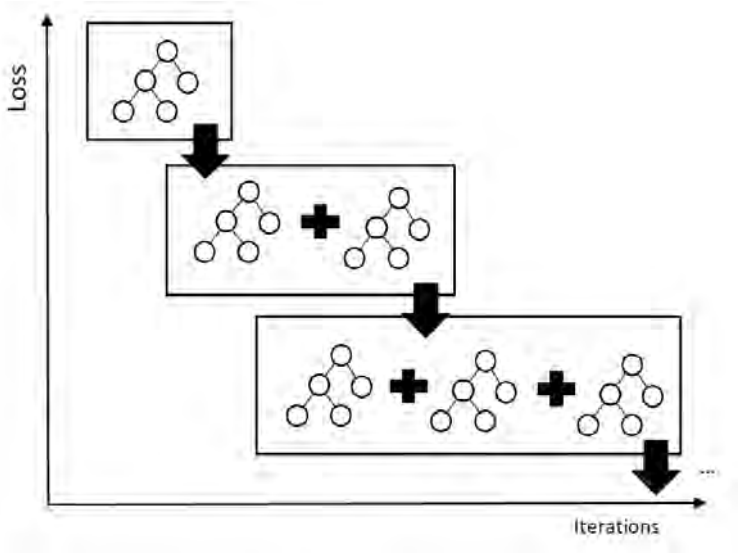

FIGURE 4.6: Representation of the iterations of the Gradient Boosting Regression algorithm (Adapted from Baturynska and Martinsen [6])

Gradient Boosting is used to optimise a given loss function, which can be any differentiable function. Similar as in Random Forest Regression, the initial model of GBR is a decision tree. After the initial tree is fitted, trees are added to the model iteratively. In each iteration, a new tree is fitted that reduces the loss of the model. This is achieved by using a variation of the gradient descent procedure as explained in section 4.5. At each iteration, the gradient of the loss function is computed and the tree is fitted on the negative gradient of the given loss function [50].

The hyperparameters of GBR are similar to those of Random Forest Regression as both are built from decision trees. Hence, GBR is subject to the parameters *n_estimators, max_features, max_depth, min_samples_split*, and *min_samples_leaf*. Furthermore, GBR has two additional parameters: the *loss function* and *learning rate*. The four possible loss functions are the least square function, the least absolute deviation function, the Huber function (a combination of the previous two), and the quantile loss function. The learning rate is similar to the learning rate of an MLP, i.e. it regulates how fast the gradient descent algorithm converges.

5 Results

The models are trained on the data from January to March 2021. The data from April 2021 is used as test data to compare the performance of the different models. The models are trained to predict the remaining travel time of a train. In this chapter, the results of each model are displayed in terms of the performance metrics and two error plots. The first plot shows how the errors of the model are distributed. The second plot, the cumulative MAE over the distance, illustrates how the error behaves depending on the distance of the train to its destination. Furthermore, the optimal values of the hyperparameters of each model are portrayed.

5.1 Baseline Model

The baseline model offers much room for improvement. The MAE of the baseline error is 2730.26 and the RSME is 4328.63. The high errors of the baseline model are visualised in Figure 5.1. There are very high positive outliers in the errors. The error is especially high when the train is close to its destination.

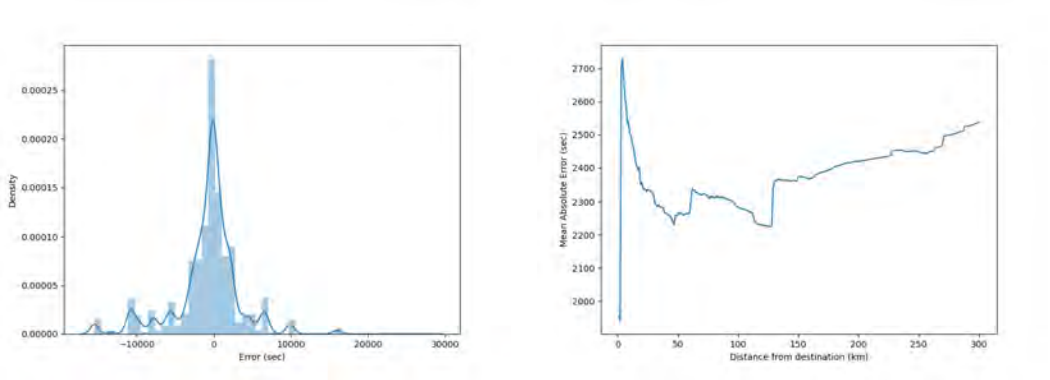

FIGURE 5.1: Distribution of errors (left) and Cumulative MAE over distance (right) of the baseline model

5.2 Linear Regression

MLR resulted in an MAE of 1867.49 and an RSME of 2561.47. Figure 5.2 shows that the errors are not evenly distributed. The majority of the errors are positive, although there are some very high negative outliers. The prediction error increases the closer the train comes to its destination.

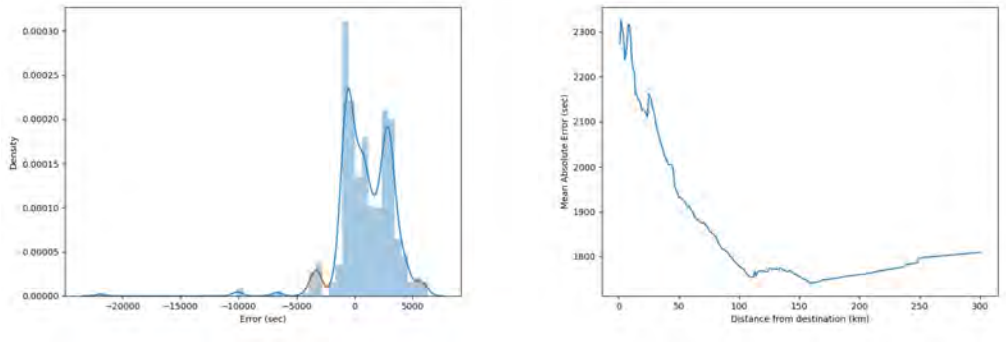

FIGURE 5.2: Distribution of errors (left) and Cumulative MAE over distance (right) of Linear Regression

5.2.1 Lasso Regression

The Lasso Regression model performed the best out of the three regularisation approaches. The optimal value for variable λ, the amount of shrinkage or regularisation, was found to be 30. Lasso Regression achieved an MAE of 1284.01 and an RSME of 1753.48. The error distribution is shown in Figure 5.3. The curve of the error distribution is not smooth, however, it shows that are only a few (negative) outliers. The figure also shows that the prediction error is particularly high when the train is 30-50 kilometres away from the destination. There is a peak around a distance of 40 km, followed by a decrease in the error, which more or less stagnates when the train is more than 150 km away from the destination.

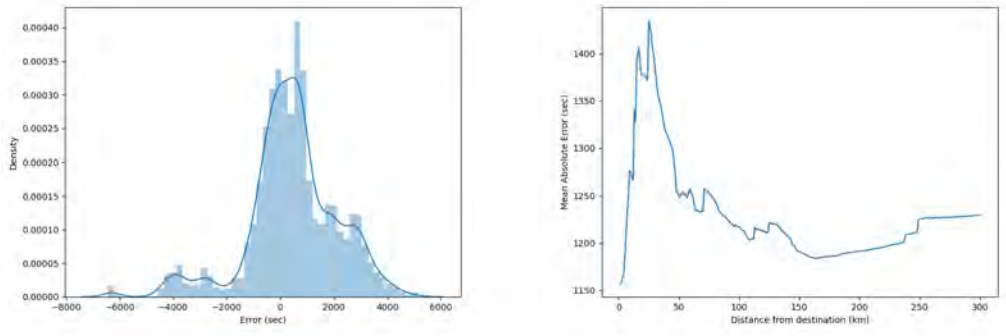

FIGURE 5.3: Distribution of errors (left) and Cumulative MAE over distance (right) of Lasso Regression

5.2.2 Ridge Regression

Ridge regression performed slightly worse than Lasso regression, although the difference between the two models is neglectable. The Ridge regression model resulted in an MAE of 1293.15 and an RSME of 1757.24. The optimal value for λ turned out to be 530. The error distribution and the cumulative average distribution over the distance of Ridge is nearly identical to those of Lasso Regression.

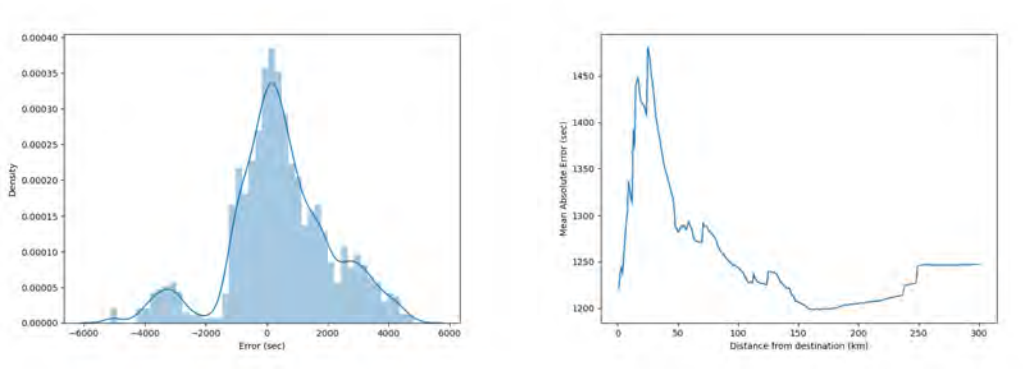

FIGURE 5.4: Distribution of errors (left) and Cumulative MAE over distance (right) of Ridge Regression

5.2.3 Elastic Net Regression

Elastic Net Regression performed the worst out of the three regularisation approaches, with an MAE of 1293.93 and an RSME of 1761.31. The optimal value trade-off parameter α between the L1 and L2 penalty was found to be 1. As explained in section 4.4.3 if $\alpha = 1$ Elastic Net is equal to Lasso regression. Still, the grid search for optimal hyperparameters revealed that $\lambda = 50$ is the optimal value instead of $\lambda = 30$ for Lasso Regression. Again the error plots in Figure 5.5 indicate that there is a minimal difference between the results of the three regularisation approaches.

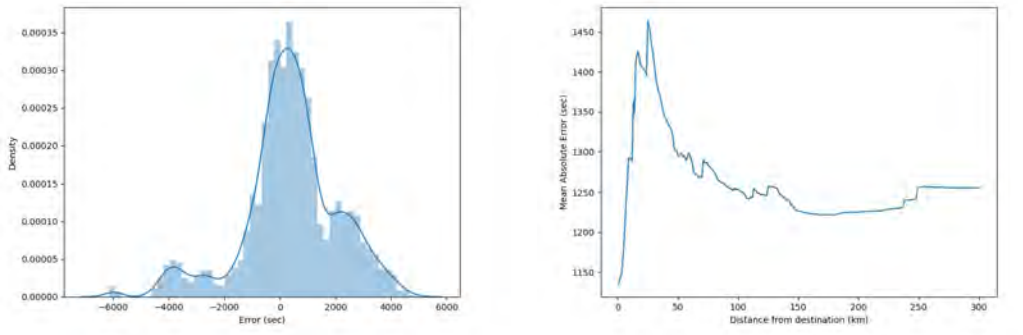

FIGURE 5.5: Distribution of errors (left) and Cumulative MAE over distance (right) of Elastic Net Regression

MAE = 1293.93 RSME = 1761.31

5.3 Multi-Layer Perceptron

The MLP model is one of the best performing models with an MAE of 1073.76 and an RSME of 1553.36. The optimal values for the hyperparameters of the MLP model are displayed in Table 5.1.

The errors are distributed smoothly as shown in Figure 5.6. The error of the model is more or less stable when the train is far away from the destination.

Parameter	Value
Activation function	ReLu
# of hidden layers	1
# of hidden nodes	5
Learning rate	0.0001
α	1

TABLE 5.1: Hyperparameters of the MLP model

There is a sudden error decrease when the train is 10 to 20 kilometres away from the destination. When the train is even closer to the station the error increases again.

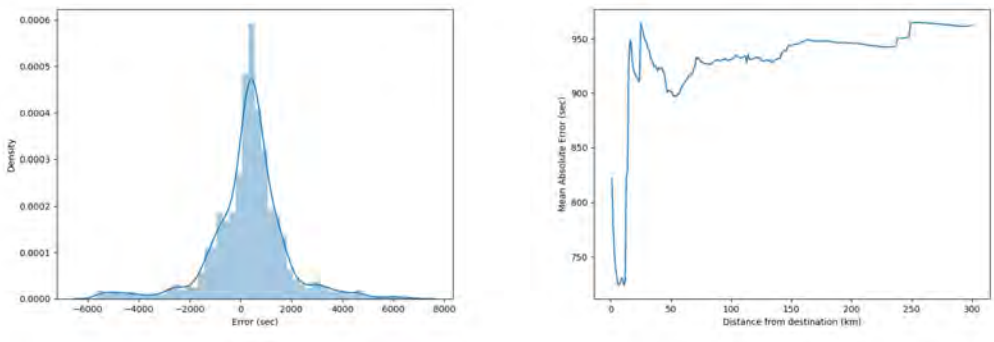

FIGURE 5.6: Distribution of errors (left) and Cumulative MAE over distance (right) of Multi-Layer Perceptron

5.4 Support Vector Machine

The SVR performs well in terms of MAE (1165.04) but achieved one of the worst RSME scores (2173.09). Table 5.2 shows the hyperparameters that were found using the Grid Search approach.

Parameter	Value
Kernel	RBF[1]
C	500
ε	100

TABLE 5.2: Hyperparameters of the SVR model

Figure 5.7 shows that there are high negative outliers in the errors. The second figure shows that the error decreases gradually as the train approaches the destination.

[1]Radial Basis Function: https://en.wikipedia.org/wiki/Radial_basis_function

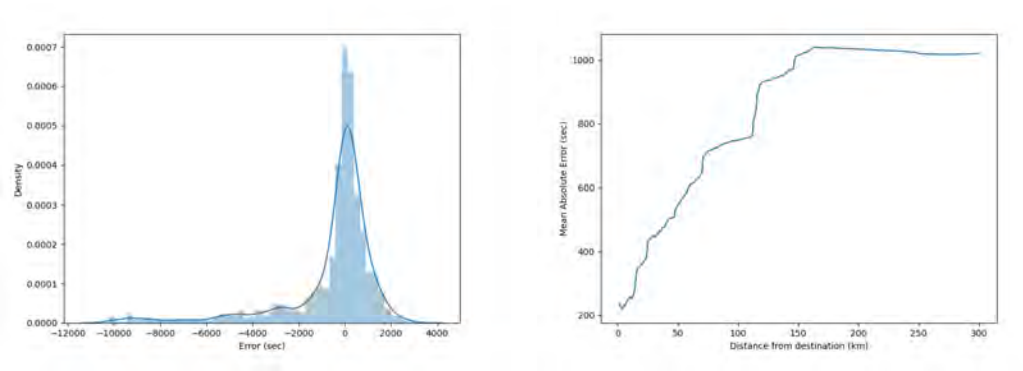

FIGURE 5.7: Distribution of errors (left) and Cumulative MAE over distance (right) of Support Vector Regression

5.5 Random Forest Regression

Overall, the Random Forest Regression performs poorly with an MAE of 1368.57 and an RSME of 2326.94. The optimised hyperparameters are displayed in Table 5.3.

Parameter	Value
# of estimators	300
Max. features	Auto
Max. depth	40
Min. samples split	3
Min. samples leaf	4

TABLE 5.3: Hyperparameters of the RFR model

The error plots in Figure 5.8 show that there are both high negative and positive errors. Nonetheless, the behaviour of the error over the distance does look as expected, i.e. the error increases gradually.

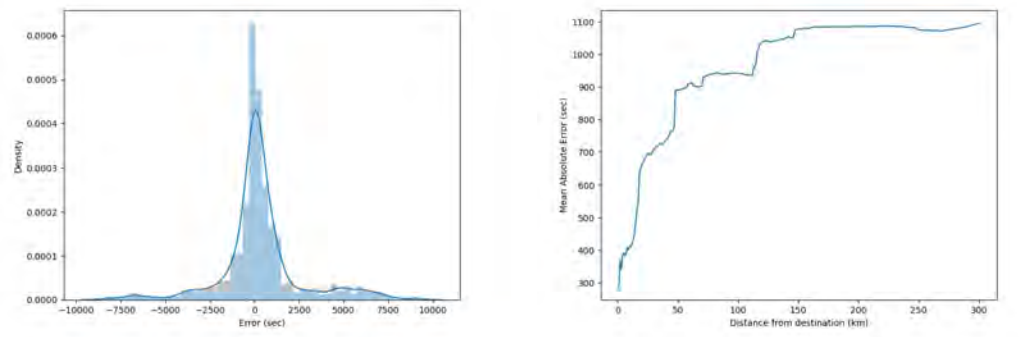

FIGURE 5.8: Distribution of errors (left) and Cumulative MAE over distance (right) of Random Forest Regression

5.6 Gradient Boosting Regression

GBR is one of the better models with an MAE of 988.31, the lowest out of all the models, and an RSME of 1775.40. The many hyperparameters of the model were optimised and are displayed in Table 5.4.

Parameter	Value
Loss function	LAD[2]
Learning rate	0.1
# of estimators	90
Max. features	Auto
Max. depth	53
Min. samples split	2
Min. samples leaf	1

TABLE 5.4: Hyperparameters of the GBR model

Figure 5.9 shows that the errors of the model are distributed smoothly, with some high negative outliers. The error gradually decreases as the train approaches the destination.

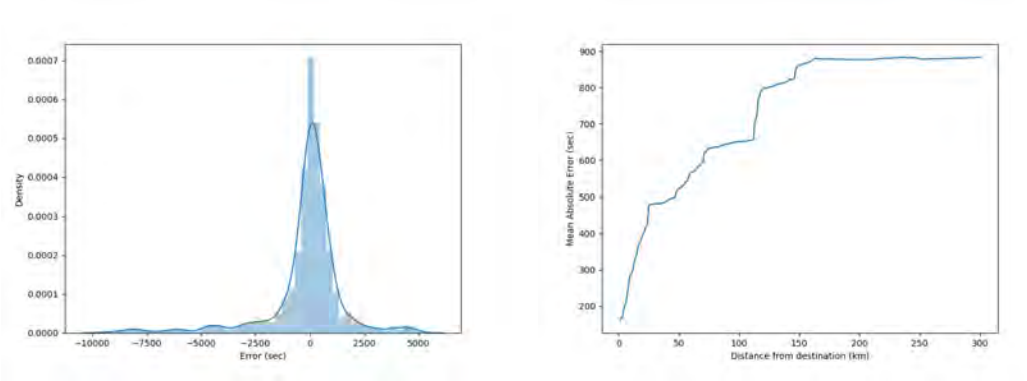

FIGURE 5.9: Distribution of errors (left) and Cumulative MAE over distance (right) of Gradient Boosting Regression

[2]Least Absolute Deviations: https://en.wikipedia.org/wiki/Least_absolute_deviations

6 Implementation

The data and the machine learning model were deployed in an application, that can be both used on a phone and in a web browser. The application was build using Mendix[1] and UbiOps[2].The application can be seen as a proof of concept, as it does not work yet in actual production. The app is based on the data that was used to train and test the models and is not able to process data in real-time yet.

6.1 Implementation in UbiOps

A so-called deployment was created in UbiOps. This deployment contains the full dataset (train and test data), the machine learning model, and a python script. The deployment acts as an API endpoint, i.e. requests can be sent to the deployment to extract the prediction and additional data. Figure 6.1 illustrates the process of extracting data from the deployment using a request.

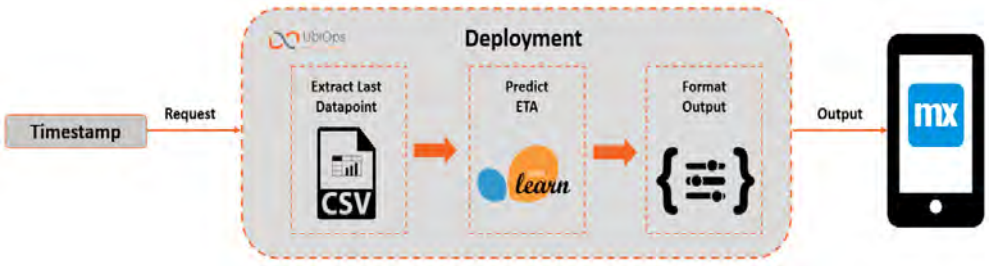

FIGURE 6.1: Representation of how a request in UbiOps works

When making a request to the deployment, a timestamp has to be passed along. The python script is run using the timestamp as an input variable. In the python script, the database will be searched for the data point corresponding to the timestamp defined in the request. As we have a dataset containing data points in a one-minute interval, the last available data point before the defined timestamp will be used to predict the ETA.

The data point is used as input for the prediction (Gradient Boosting Regression) model. The model predicts the remaining travel time, which can be easily converted into a timestamp, the ETA. The ETA prediction, along with other features that are interesting for the user, is outputted as a JSON format. Thereafter, the output can be displayed in an application using Mendix.

[1]Mendix: https://www.mendix.com/
[2]UbiOps: https://www.ubiops.com/

6.2 Implementation in Mendix

Upon opening the application the user is presented with an overview of all locomotives that are used by the railway operator, which is shown on the left of Figure 6.2. In this case, only the data of the first locomotive is available and the other locomotives are added to demonstrate how the application would look in production. The user can select the locomotive that he is interested in from the start screen.

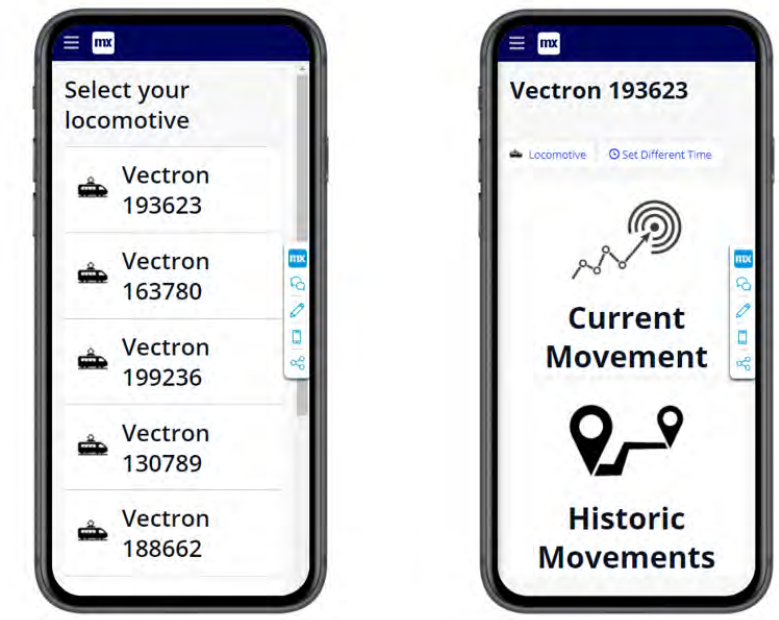

FIGURE 6.2: Start screen of the application (left) and start screen for a specific locomotive (right)

After selecting the locomotive, a request is sent to the UbiOps deployment. In this demo version of the app, the user is asked to specify the current time. As mentioned earlier, the app does not work in real-time yet, so the application will act as if the selected time is the current time of the device. A request is sent to the UbiOps deployment, which outputs the prediction and other information to the application.

When the request is processed (after a few seconds) the home screen that is displayed on the right in Figure 6.2 is shown. On this page, the user has two options: the user see an overview of the prediction (output of deployment), or the user can view an overview of the rides the locomotive has made in the past.

6.2.1 Prediction Overview

After clicking on the button "Current Movement", the user is given an overview of the current state of the locomotive, including the predicted ETA. However, the ETA is displayed as an interval, rather than a single timestamp. This approach was inspired by postal service and grocery delivery applications. Portraying the ETA as an interval allows some room for errors. When the train is more than 50 kilometres away from its destination, the interval is half an hour

to each side of the predicted ETA. As the train comes closer to the destination, the interval shrinks. When the train is closer than 50 kilometres to its destination, the interval is reduced to 15 minutes to each side. Finally, when the train is within ten kilometres of its destination the interval is five minutes to each side. The different interval sizes are portrayed in Figure 6.3.

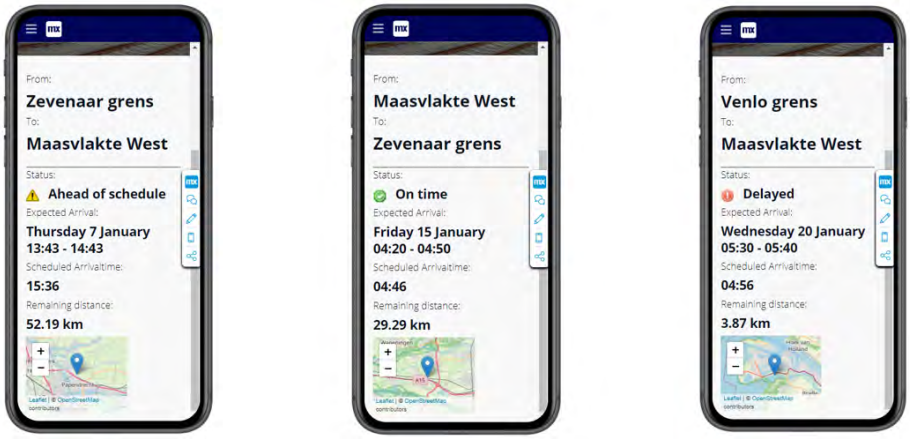

FIGURE 6.3: Prediction overview of a train that is ahead of schedule (left), a train that is on time (middle) and a train that is delayed (right)

Figure 6.3 shows the prediction overview screen of the application. The page shows the origin and destination of the current train movement. Along with the prediction interval of the ETA, the scheduled arrival time is also shown. This is used to compute whether the train is ahead of schedule (scheduled time is after the interval), on time (scheduled time is contained in the interval) or delayed (scheduled time is before the interval). Furthermore, the remaining distance to the station and a map with the current location of the locomotive are displayed. Of course, this screen is only relevant when a train is on an active journey. If the train has already arrived, the user will be informed of the arrival time on the screen displayed in Figure 6.4.

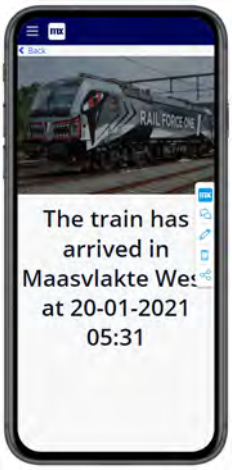

FIGURE 6.4: Screen when the train has already arrived

6.2.2 Train Movement Overview

The application can also be used to view the rides a locomotive has made in the past. To access this overview the user can click on "Historic Movements" on the home screen. The user will be presented with a list of all the movements the locomotive has made, as seen on the left in Figure 6.5. The user can scroll through the list of movements and click on the movement he/she is interested in. A second screen will load where the specifics of the selected movement are displayed.

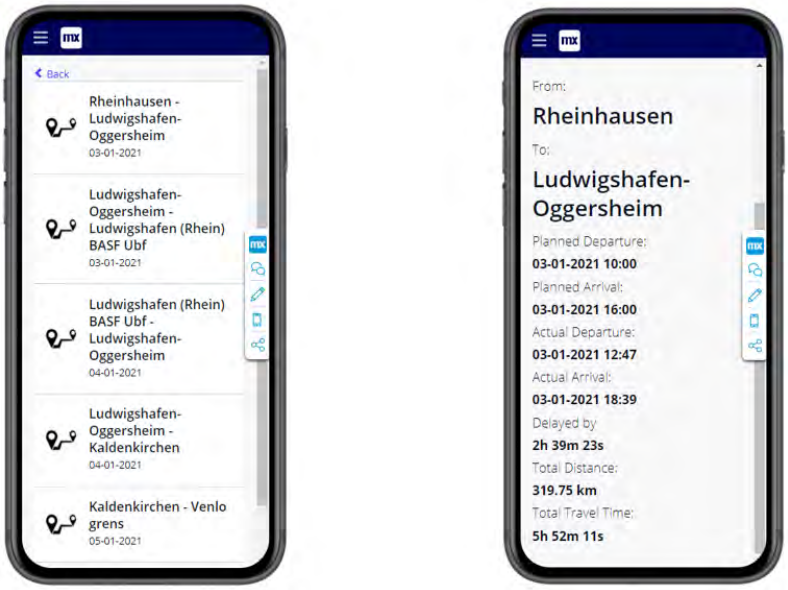

FIGURE 6.5: Overview of train movements (left) and information of a specific movement (right)

The information of a specific movement that is portrayed is the planned departure and arrival times (from the timetable data), and the actual departure and arrival times (from the AVL data). The difference between the planned arrival time and the actual arrival time is the delay of the train. If this value is negative, i.e. the train arrived earlier than the planned arrival time, the train is ahead of schedule and the caption "Delayed by" is changed to "Ahead by". Additionally, the total distance of the movement and the total amount of time the journey took are displayed at the bottom of the page.

7 Discussion

The results presented in section 5 are summarised in Table 7.1. The table shows the MAE and RSME of each model, along with a comparison between the models and the baseline model. The percentages shown in the table depict how much the metric of a model improved compared to the baseline model.

Model	**MAE**	**MAE (%)**	**RSME**	**RSME (%)**
Baseline	2730.26	-	4328.63	-
MLR	1867.49	31.60	2561.47	40.82
Lasso	1284.01	52.97	1753.48	59.49
Ridge	1293.15	52.64	1757.24	59.40
Elastic Net	1293.93	52.61	1761.31	59.31
MLP	1073.76	60.67	**1553.36**	**64.11**
SVR	1165.04	57.33	2173.09	49.80
RFR	1368.57	49.87	2326.94	46.24
GBR	**988.31**	**63.80**	1775.40	58.98

Table 7.1: Summary of model results

All the models were able to outperform the baseline model significantly. Multiple Linear Regression is the worst performing model of all models considered, however, the model still offered a great improvement of the baseline model. The model could be further improved upon by adding regularisation. The models with regularisation performed surprisingly well, especially in terms of their RSME scores.

The three regularisation approaches achieved nearly equivalent results. There is a 0.36% difference in MAE improvement between the best performing approach (Lasso) and the worst performing approach (Elastic Net). In the same manner, the difference in RSME improvement is 0.18%. This difference in the performance of the three approaches is so insignificant that no binding conclusion can be made on which approach is best in this case. Having said that, it has become evident that adding any form of regularisation is a major step forward.

Nonetheless, the linear models with and without regularisation are not suitable for actual implementation. Linear regression fits a linear function to the data. This linear function contains coefficients that can be both positive and negative. Hence, the function can also output negative values. In practice, a negative prediction makes no sense. Predictions are only made when the train is not yet at its destination. A negative prediction would mean the train has already arrived, hence the ETA is in the past. The effect of negative predictions is visible in the error graphs of the linear regression models (fig 5.2 - 5.5). The

errors of the linear models are the highest when the train is close to its destination. This is a result of negative predictions when the remaining distance to the destination is close to zero.

The lowest RSME was achieved by the MLP model. Furthermore, the model achieved a competitive MAE score. A low RSME score shows that the model has few extreme errors. This can also be concluded from the error distribution plot in figure 5.6. The error plot for the MLP model is the smoothest plot of all the models considered, i.e. MLP has the densest error distribution. However, the second plot in figure 5.6 shows some unpreferable behaviour. There is a sudden drop in the error when the train gets closer to its destination, yet the error increases when the train is in immediate proximity. Again, this can be explained by the MLP model outputting negative predictions. Although this occurs less often with the MLP often than for the Linear Regression models, this still has an adverse impact on the performance of the model.

Both the Support Vector Regression and Random Forest Regression models failed to live up to the expectations. Even though the MAE score of the SVR model is acceptable, the RSME can not hold up with the other models. This means on average the SVR model performs reasonably well, yet the model is subject to a lot of high errors. This can be explained by the hyperparameters that were used for the model. The optimal value for ε is relatively low (100), while the optimal value for C is high (500). This means the model initially allows the prediction error to be 100 seconds. However, the high value for C means the model has a high tolerance for errors that exceed the margin of 100 seconds. The resulting model has a relatively low error on average but has a high risk for big outliers.

The RFR performed poorly on both metrics. The model does behave as wanted, i.e. the error increases as the distance increases as seen in figure 5.8. Despite that, the errors are too high for the model to be implemented in production. The disappointing performance of the RFR model could be the result of the sparsity of the data. Data is considered to be spare when most elements are equal to zero. As stated in section 3.6, there are three categorical features included in the data. However, these three features have 79 categories combined (4 intervals, 7 weekdays, 68 movements), of which only three are non-zero. Hence, as the majority of the data is filled with zero values, the data is sparse. Random Forest models tend to not perform well on sparse data [51], which is most likely also the case for this model.

The Gradient Boosting Regression achieved the lowest MAE score of all the models. This indicates that the model has the lowest error on average. The RSME score is in the midfield of all the models. Still, the GBR model was found to be the best-suited model for implementation. Figure 5.9 shows that although the error distribution is reasonably smooth and dense, there are some big negative errors. The reason the GBR model is prefered over the MLP model is the behaviour of the errors over the distance. The cumulative error plot in figure 5.9 shows that the error of the model gradually decreases as the train approaches its destination. While there are some high outliers in the predictions of the model, the low average error, and the desirable behaviour of the model outweigh the limitations of the model.

The best model, the GBR model, still has a considerable MAE of 988.31 seconds (over 16 minutes). When the model is deployed in an application, the error could mislead a potential user. If the exact ETA is presented to the user, this raises the expectation of the user that the train will arrive at that exact time. However, as research has shown it is almost impossible to predict the ETA exactly, especially for long-distance train rides. Inspiration was drawn from applications of postal services (DHL, PostNL, UPS, etc.) to present the user with an ETA interval, rather than an exact ETA. The interval has to be big enough that the actual arrival time is contained in the interval on most occasions. However, the interval should be kept small enough that the prediction of the ETA has an added value. The decision was made to take twice the MAE as an interval to each side of the ETA. The bounds were rounded to a presentable number (from 32 minutes to 30 minutes). As the error of the model decreases as the train approaches its destination, the interval size also decreases as the train comes closer. When the train is less than 50 kilometres away from the destination the interval shrinks to 15 minutes to each side. When the train is within 10 kilometres from the destination the interval shrinks again to five minutes to each side.

8 Conclusion

To conclude, an accurate prediction of the ETA of a cargo train is an important asset in railway transportation. Knowing when a train arrives and recognising deviations from the schedule on time can help reduce cost. Furthermore, resources can be allocated to exactly where they are needed, which can save time, resources and money.

Two datasets were used for this project, both entailing four months of data. One dataset contains the GPS data from a locomotive and the second dataset consists of the timetable data corresponding to the locomotive. The GPS data was transformed into temporal features, such as the distance travelled and the speed of the train. Inconsistencies in the data were removed and the gaps were filled using linear interpolation. Thereafter, the timeline of the train could be extracted, i.e. a timeline of when the train has passed or has stopped at a station. The timeline was matched to the timetable data, using a matching algorithm. Multiple time, distance and train dimensions related features were engineered. This resulted in a dataset of data points in a one-minute interval.

The data was split into a test and a train set. A wide variety of models was trained and examined. The models were evaluated on their MAE and RSME scores. A Multiple Linear Regression model was found to perform the worst. The model could be improved by adding regularisation. Lasso, Ridge and Elastic Net regression all formed a significant improvement compared to the MLR model. The difference in the performance of the three regularisation methods was so small that no conclusion could be made on which approach works the best.

Support Vector Regression and Random Forest Regression both performed considerably worse than expected. The Multi-Layer Perceptron and Gradient Boosting Regression model were the two best performing models. The MLP model resulting in the lowest RSME score, while the GBR model achieved the lowest MAE. The choice for the best model could therefore not solely be based on which model achieved the lowest metric scores. The behaviour of the errors of the model over the distance was the decisive factor in which model would be considered "the best". The MLP model showed some unintended behaviour when the train comes close to its destination due to the model returning negative predictions, whereas the GBR model behaves as expected. Hence, the GBR is considered to be the best-suited model for predicting the ETA of cargo trains.

The GBR model was deployed in an application that acts as a proof of concept. The application shows how the machine learning model could be implemented by a railway operator. The application can be provided to customers of a railway operator. The application can be used to monitor the ETA of a cargo train. The ETA is presented as an interval, rather than as an exact timestamp. This approach was chosen so that despite the model being subject to a considerable error, the ETA prediction is still accurate and valuable.

8.1 Limitations and further research

A limitation of the research is the data, or rather the lack thereof. The available data was limited, as only the data of one locomotive over the span of four months was considered. This results in certain rides being overrepresented in the data, while other rides are only contained once or twice. An extensive dataset, spanning over a year or even multiple years would enable the possibility to include each ride the same amount of times as train data. An evenly distributed training dataset reduces the bias of the models and would increase the overall accuracy of the models.

The arrival times of trains are highly influenced by external features. Delays often occur when there are bad weather circumstances, such as heavy rainfall, snow or storms. Therefore, including features on the current weather conditions (temperature, wind and precipitation) would further enhance the predictive power of a machine learning model. However, in this research including the weather conditions would have an adverse result. As the training data is from January until March, the weather conditions are more or less the same throughout the data. The weather in this particular winter was extremely bad, with a lot of snow, rain and heavy wind. If the weather conditions were to be included, the models would only be trained on bad weather. As the weather in the month the test data stems from and the months that followed was a lot better, the model would have to make a prediction on weather conditions the model has not been trained on. If the model would be trained on a year of data or more, including the weather conditions would make more sense.

Another important external feature that impacts the arrival time of trains is other trains in the network. A train can be held up by other trains in front of them. Delays are more likely to occur when there is a high traffic density. On the other hand, trains can get ahead of schedule when there are no afflicting trains in their way. For public transportation in the Netherlands, there are open databases, called NDOV data[1], where data on traffic density and the network can be found. However, for cargo trains, this data is more restricted as there are a lot of parties involved (multiple railway operators and clients). The data can only be accessed with the permission of the railway manager ProRail. It would be interesting to look into a cooperation with ProRail, so that data on the entire network could also be included in the model.

A lot of models were examined to find which model performs best for predicting the ETA. The conformity between all the models is that they are deterministic. The models are used to predict the remaining travel time of a train to its destination. For any data point between the departure and arrival of the train a prediction can be made how long the train will need from the current location to the destination. As we have data of a fixed interval size, time-series models such as a Long Short-Term Memory (LSTM) models or a Seasonal Autoregressive Integrated Moving Average (SARIMAX) model would also be very fitting for this problem.

[1] NDOV loket: https://www.ndovloket.nl/

Traditionally, time-series models are used on continuous data of fixed interval size, i.e. for each data point there are previous data points are available. Time-series models can find patterns in data based on the prior data points. For example, a time-series model can be used to predict the demand of a product based on the demand of the product in the previous week. The difficulty of implementing a time-series model in this particular case is that we do not have continuous data. The data points when a train is stationary (at a station, a depot or for maintenance) are not used for training the models. The problem of using time-series models on data with gaps is known as the cold-start problem. Due to time limitations, the decision was made to focus on deterministic models. However, there are approaches to bypass the cold-start problem [52], which would be interesting to explore in the future.

The application described in section 6 works on historical data. To be able to use the application in actual production, the data needs to be processed in real-time. The GPS tracker sends the data to a database. The raw GPS data needs to be transformed into features as explained in section 3. Thereafter, a prediction can be made on the last available data point. For the data to be up-to-date at all times, there should be a script running in a cloud that constantly transforms the latest GPS location of the train into features. The UbiOps deployment should be slightly altered so that the last data point in the database is used as input for the machine learning model, rather than the last point before the given timestamp. The deployment would no longer take a timestamp as input. Instead, the deployment should have access to the database containing the transformed data.

Bibliography

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

[2] H. Hassan, A. Negm, M. Zahran, and O. Saavedra, "Assessment of artificial neural network for bathymetry estimation using high resolution satellite imagery in shallow lakes: Case study el burullus lake.," *International Water Technology Journal*, vol. 5, 12 2015.

[3] Yadavendra and S. Chand, "A comparative study of breast cancer tumor classification by classical machine learning methods and deep learning method," *Machine Vision and Applications*, vol. 31, 07 2020.

[4] B. Schölkopf and A. Smola, "Support vector machines and kernel algorithms," *Encyclopedia of Biostatistics, 5328-5335 (2005)*, 04 2002.

[5] A. Chakure, "Random forest regression along with its implementation in python." https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f, 2020. Accessed: 10.06.2021.

[6] I. Baturynska and K. Martinsen, "Prediction of geometry deviations in additive manufactured parts: comparison of linear regression with machine learning algorithms," *Journal of Intelligent Manufacturing*, vol. 32, 01 2021.

[7] Eurostat, "Freight transport statistics - modal split." [Online]: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Freight_transport_statistics_-_modal_split, 2019. Accessed: 11.06.2021.

[8] J. Shefer, "Rail freight transportation: How to save thousands of dollars the environment." https://web.archive.org/web/20151208051933/http://moveitwithjon.com/blog/rail-freight-transportation-save-thousands-of-dollars/. Archived on 08.12.2015.

[9] E. E. Agency, "Motorised transport: train, plane, road or boat — which is greenest?." https://www.eea.europa.eu/highlights/motorised-transport-train-plane-road. Accessed on 11.06.2021.

[10] W. Paprocki, "How transport and logistics operators can implement the solutions of "industry 4.0"," in *Sustainable Transport Development, Innovation and Technology* (M. Suchanek, ed.), (Cham), pp. 185–196, Springer International Publishing, 2017.

[11] F. Ghofrani, Q. He, R. M. Goverde, and X. Liu, "Recent applications of big data analytics in railway transportation systems: A survey," *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 226–246, 2018.

[12] A. Consilvio, J. Solís-Hernández, N. Jiménez-Redondo, P. Sanetti, F. Papa, and I. Mingolarra-Garaizar, "On Applying Machine Learning and Simulative Approaches to Railway Asset Management: The Earthworks and Track Circuits Case Studies," *Sustainability*, vol. 12, pp. 1–24, March 2020.

[13] R. Nappi, "Integrated maintenance: analysis and perspective of innovation in railway sector," 04 2014.

[14] L. Yang, T. Xu, and Z. Wang, "Agent based heterogeneous data integration and maintenance decision support for high-speed railway signal system," pp. 1976–1981, 10 2014.

[15] H. Guler, "Decision support system for railway track maintenance and renewal management," *Journal of Computing in Civil Engineering*, vol. 27, no. 3, pp. 292–306, 2013.

[16] O. Asekun and C. Fourie, "Selection of a decision model for rolling stock maintenance scheduling," *The South African Journal of Industrial Engineering*, vol. 26, no. 1, pp. 135–149, 2015.

[17] M. Kovacevic, K. Gavin, I. S. Oslakovic, and M. Bacic, "A new methodology for assessment of railway infrastructure condition," *Transportation Research Procedia*, vol. 14, pp. 1930–1939, 2016. Transport Research Arena TRA2016.

[18] H. Yilboga, F. Eker, A. Güçlü, and F. Camci, "Failure prediction on railway turnouts using time delay neural networks," in *2010 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pp. 134–137, 2010.

[19] C. Hu and X. Liu, "Modeling track geometry degradation using support vector machine technique," p. V001T01A011, 04 2016.

[20] J. Yin and W. Zhao, "Fault diagnosis network design for vehicle on-board equipments of high-speed railway: A deep learning approach," *Engineering Applications of Artificial Intelligence*, vol. 56, pp. 250–259, 2016.

[21] E. Kamburjan, R. Hähnle, and S. Schön, "Formal modeling and analysis of railway operations with active objects," *Science of Computer Programming*, vol. 166, pp. 167–193, 2018.

[22] P. Wang and Q.-p. Zhang, "Train delay analysis and prediction based on big data fusion," *Transportation Safety and Environment*, vol. 1, pp. 79–88, 02 2019.

[23] C. Wen, W. Mou, P. Huang, and Z. Li, "A predictive model of train delays on a railway line," *Journal of Forecasting*, vol. 39, no. 3, pp. 470–488, 2020.

[24] R. Jeong and R. Rilett, "Bus arrival time prediction using artificial neural network model," in *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*, pp. 988–993, 2004.

[25] L. Vanajakshi and L. R. Rilett, "Support vector machine technique for the short term prediction of travel time," in *2007 IEEE Intelligent Vehicles Symposium*, pp. 600–605, 2007.

[26] Z. Gurmu and W. Fan, "Artificial neural network travel time prediction model for buses using only gps data," *Journal of Public Transportation*, vol. 17, pp. 45–65, 06 2014.

[27] J. Amita, S. Jain, and P. Garg, "Prediction of bus travel time using ann: A case study in delhi," *Transportation Research Procedia*, vol. 17, pp. 263–272, 2016. International Conference on Transportation Planning and Implementation Methodologies for Developing Countries (12th TPMDC) Selected Proceedings, IIT Bombay, Mumbai, India, 10-12 December 2014.

[28] S. Maiti, A. Pal, A. Pal, T. Chattopadhyay, and A. Mukherjee, "Historical data based real time prediction of vehicle arrival time," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1837–1842, 2014.

[29] J. Murphy, R. Reisman, J. Clayton, and R. Wright, *Physics-Based and Parametric Trajectory Prediction Performance Comparison for Traffic Flow Management.*

[30] A. Achenbach and S. Spinler, "Prescriptive analytics in airline operations: Arrival time prediction and cost index optimization for short-haul flights," *Operations Research Perspectives*, vol. 5, pp. 265–279, 2018.

[31] C. Strottmann Kern, I. P. de Medeiros, and T. Yoneyama, "Data-driven aircraft estimated time of arrival prediction," in *2015 Annual IEEE Systems Conference (SysCon) Proceedings*, pp. 727–733, 2015.

[32] Y. Glina, R. Jordan, and M. Ishutkina, "A tree-based ensemble method for the prediction and uncertainty quantification of aircraft landing times," in *American Meteorological Society–10th Conference on Aritificial Intelligence Applications to Environmental Science, New Orleans, LA*, 2012.

[33] A. Balster, O. Hansen, H. Friedrich, and A. Ludwig, "An eta prediction model for intermodal transport networks based on machine learning," *Business Information Systems Engineering*, vol. 62, 10 2020.

[34] N. Servos, X. Liu, M. Teucke, and M. Freitag, "Travel time prediction in a multimodal freight transport relation using machine learning algorithms," *Logistics*, vol. 4, no. 1, 2020.

[35] A. Schrijver, "Wiskunde achter het spoorboekje," *Centrum voor Wiskunde en Informatica*, 2007.

[36] A. Schrijver and A. Steenbeek, "Spoorwegdienstregelingontwikkeling," *Centrum voor Wiskunde en Informatica*, 1993.

[37] W.-H. Lee, S.-S. Tseng, and S.-H. Tsai, "A knowledge based real-time travel time prediction system for urban network," *Expert Systems with Applications*, vol. 36, no. 3, Part 1, pp. 4239–4247, 2009.

[38] T. Huisman and R. J. Boucherie, "Running times on railway sections with heterogeneous train traffic," *Transportation Research Part B: Methodological*, vol. 35, no. 3, pp. 271–292, 2001.

[39] M. Yaghini, M. M. Khoshraftar, and M. Seyedabadi, "Railway passenger train delay prediction via neural network model," *Journal of Advanced Transportation*, vol. 47, no. 3, pp. 355–368, 2013.

[40] A. Prokhorchenko, A. Panchenko, L. Parkhomenko, G. Nesterenko, M. Muzykin, G. Prokhorchenko, and A. Kolisnyk, "Forecasting the estimated time of arrival for a cargo dispatch delivered by a freight train along a railway section," *Eastern-European Journal of Enterprise Technologies*, vol. 3, pp. 30–38, 06 2019.

[41] J. Hu and B. Noche, "Application of artificial neuron network in analysis of railway delays," *Open Journal of Social Sciences*, vol. 04, pp. 59–68, 01 2016.

[42] W. Barbour, C. Samal, S. Kuppa, A. Dubey, and D. Work, "On the data-driven prediction of arrival times for freight trains on u.s. railroads," pp. 2289–2296, 11 2018.

[43] Z. Li, C. Wen, R. Hu, C. Xu, P. Huang, and X. Jiang, "Near-term train delay prediction in the dutch railways network," *International Journal of Rail Transportation*, vol. 0, no. 0, pp. 1–20, 2020.

[44] W. Barbour, J. C. Martinez Mori, S. Kuppa, and D. B. Work, "Prediction of arrival times of freight traffic on us railroads using support vector regression," *Transportation Research Part C: Emerging Technologies*, vol. 93, pp. 211–227, 2018.

[45] N. Marković, S. Milinković, K. S. Tikhonov, and P. Schonfeld, "Analyzing passenger train arrival delays with support vector regression," *Transportation Research Part C: Emerging Technologies*, vol. 56, pp. 251–262, 2015.

[46] S. Walczak and N. Cerpa, "Artificial neural networks," in *Encyclopedia of Physical Science and Technology (Third Edition)* (R. A. Meyers, ed.), pp. 631–645, New York: Academic Press, third edition ed., 2003.

[47] M. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, no. 14, pp. 2627–2636, 1998.

[48] A. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 08 2004.

[49] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[50] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent," in *Advances in Neural Information Processing Systems* (S. Solla, T. Leen, and K. Müller, eds.), vol. 12, MIT Press, 2000.

[51] M. Shammeer, "Random forest fails." https://www.medium.com/swlh/random-forest-fails-a8ca2d46c312. Accessed on 15.06.2021.

[52] C. Fitzpatrick, "How to use an lstm for timeseries and the cold-start problem." https://www.drivendata.co/blog/benchmark-cold-start-lstm-deep-learning/. Accessed on 16.06.2021.