Data-Driven Travel Time Prediction

JOAR NYKVIST
Data-driven Travel Time Prediction

JOAR NYKVIST

Master in Machine Learning
Date: July 2, 2019
Supervisor: Somayeh Aghanavesi
Examiner: Erik Fransén
School of Electrical Engineering and Computer Science
Host company: Scania CV AB
Swedish title: Datadrivna metoder för uppskattning av restid
Abstract

Being able to accurately forecast the time of arrival of a vehicle in traffic appeals both to private drivers aiming to keep up with their schedules, and to businesses that need to organize transport logistics. This thesis is assigned by the Swedish truck manufacturer Scania CV AB, and it sets out to use GPS data from Scania’s vehicle fleet to train Machine Learning models to predict the travel times of vehicles between stops. The predictive models implemented train on features engineered from quite simple information from the vehicles, yet reach high predictive accuracy in certain scenarios. Two approaches to predicting travel time are tested, one referred to as the Local Models approach, and the other as the Global Model approach. In the Local Models approach, many separate regressors are trained on geographical subsets of the data and then combined to give global predictions. In the Global Model approach, a single regressor trains on the entire data set. The Global Model approach gives better performance than that of the Local Models in the experiments, but the Local Models approach shows some promising tendencies. It is found that a regressor predicts significantly more accurately when the geographical spread of the data is limited.
Sammanfattning

Acknowledgements

First off, I want to thank my supervisor at Scania, Rikard Laxhammar, for providing great mentorship and helping me think throughout the project. Next I want to thank my supervisor at KTH, Somayeh Aghanavesi, for guiding me through the academic parts of the thesis. Thanks to Erik Franzén for examining my thesis project.

I also want to thank Zehua Chen for a great partnership when we worked on the map-matching problem. Further thanks go to the whole KSEA team at Scania who helped me so much with all kinds of things throughout my time there.

To my mother, Helen Engman Nykvist:
You are my hero and I am blessed to have been raised by you.

To my father, Björn Nykvist:
I would have gotten nowhere without your ceaseless support and wisdom.

I want to thank my brothers, Torkel, Axel, and Vidar Nykvist for being my biggest fans out there.

Finally, I thank my girlfriend, Anna Heikkilä, for being the best life partner I could ever imagine.
# Contents

1 Introduction 1
   1.1 Background ........................................ 1
   1.2 Research Question .................................. 2
   1.3 Objective ........................................... 2
   1.4 Scope ................................................ 3
   1.5 Disposition ......................................... 3

2 Literature Review 4
   2.1 Travel Time Prediction .............................. 4
   2.2 Map-matching ....................................... 5
   2.3 Hidden Markov Models for Map-matching .......... 5
   2.4 Map-Matching Software ............................. 6
   2.5 Routing ............................................. 7
   2.6 Feature selection .................................. 8
   2.7 Machine Learning ................................... 9
      2.7.1 Tree-based methods ............................ 10
      2.7.2 Support Vector Machine ....................... 11
      2.7.3 Artificial Neural Networks ................... 13

3 Method 16
   3.1 Data Preprocessing ................................. 16
   3.2 Feature Engineering ................................ 19
      3.2.1 Cyclic Features Describing Time ............... 19
      3.2.2 One-hot Encoded Vehicle IDs .................. 21
      3.2.3 The Average Median Velocity Feature .......... 22
   3.3 Data Sets ........................................... 24
   3.4 Predictive Models .................................. 27
      3.4.1 Global Model Approach ........................ 27
      3.4.2 Local Models Approach ........................ 29
3.5 Performance Metrics ........................................ 32

4 Results .................................................................. 33
  4.1 Feature Importance ........................................... 33
  4.2 Local Models and Global Model Compared ............. 35
    4.2.1 Statistical Comparison of the Local Models and the
               Global Model ............................................. 39

5 Discussion .......................................................... 41
  5.1 Data ............................................................... 41
  5.2 Method ............................................................ 42
  5.3 Evaluation ........................................................ 43
  5.4 Predictive Performance ....................................... 44
  5.5 Future Work ..................................................... 45
  5.6 Ethics and Sustainability aspects ......................... 45

6 Conclusion .......................................................... 46

Bibliography .......................................................... 47

A Feature and Target Distributions .............................. 51

B Ridge Regression ................................................ 54
Chapter 1

Introduction

1.1 Background

When analyzing a traffic network from a macroscopic perspective, the flows of vehicles occur volatile, despite the quite explicit rules for how vehicles can travel. A naive forecast of a vehicle’s journey between two specific locations would use the accumulated length of the road segments between the locations, combined with the highest legal velocity along each segment, and would so, without much difficulty, offer a prognosis of how the journey would unfold. This naive way of forecasting vehicle journeys in traffic would very seldom be accurate in comparison to the ground truth, because of the many external factors that introduce uncertainty to the traffic network.

It is challenging to mathematically model the uncertainty in traffic networks, although it is usually straightforward for a driver to explain why their time of arrival deviated from the expected. “I was late because there was a traffic jam on road X due to an accident on road Y. The holiday season is the worst for driving.”

This explicable suggests that there are coherent cause-and-effect relations that account for the anomalies in traffic flows, which could be mathematically represented by a data-driven model, given enough data with the appropriate type of information. Out of that notion springs the foundation for this thesis, how should data be used to explain the variations in travel time for vehicles traveling in traffic networks?

Scania CV AB is one of the world’s largest truck manufacturers [1], and their connected vehicle fleet makes millions of journeys everyday. This extensive connected fleet of trucks spawns a vast collection of data that can be used to analyze how vehicles travel in traffic. The data analyses in this thesis are
done using this data, and the thesis itself is assigned by Scania. The value of being able to accurately predict the travel time of vehicles lies in how it allows for journey logistics to be better planned, which in turn enables optimization in economical and environmental aspects.

1.2 Research Question

This thesis project aspires to answer the following research question:

Which is the most suitable approach for reaching accurate travel time predictions using Machine Learning models trained on historical GPS data from heavy-duty vehicles?

1.3 Objective

As indicated by the research question, the main objective of this thesis is to find a way to accurately predict the travel time of trucks, using methods of machine learning, and historical GPS data.

Reaching this objective requires solutions to two distinct but related problems. The first such problem is that of preprocessing the GPS data so that it can be analyzed profitably, and the second problem is that of designing the analysis itself. The better portion of the thesis focuses on the preprocessing because it is deemed to hold higher potential for getting favorable results, and it is also a prerequisite for doing analysis on the data.

Preprocessing GPS data demands for solutions to some manifold questions, and different approaches to providing these solutions are of interest to the research field. The first question faced will be how to clean the GPS data of noise and how to assert that the coordinates used in training the machine learning models in fact lie on some traffic road. Once this has been accomplished, the problem of grouping the GPS samples into discrete vehicle trips arises, after which one needs to decide what information to include in the patterns that the machine learning model will train on, and how to find that information. Finally, deciding on the scope of the data is an important part of the preprocessing, calling for decisions on how to confine the spatial and temporal distributions of the data, and what other limitations should be imposed in order to achieve effective modeling.
1.4 Scope

Excluded from the research scope of this thesis is crafting novel machine learning methods for analyzing the data, this is in order to aim attention at how the data itself is wielded, and in the end trained on by different recognized machine learning methods. How data is collected from the trucks is not included in the scope either, the project rather focuses on how to use the data that has already been collected from them. Finally, the scope is confined to supervised regression methods for analysis, and will not cover unsupervised methods like reinforcement learning or clustering.

1.5 Disposition

This thesis commences with a literature review that introduces the reader to relevant theory and to related studies. The literature study first introduces the reader to the problem of travel time prediction with a special emphasis on the closely linked problems of map-matching and routing, after which it continues to discuss machine learning and feature selection. The method section that follows explains the pipeline used to process data, which data sets were used and how they were selected, and which machine learning methods were used and how. In the results section, the reader is presented with results on feature metrics and distributions, along with prediction results. These results are analyzed and reflected on in the Discussion section, and finally, summarizing remarks are provided in the Conclusion section.
Chapter 2

Literature Review

This literature review serves two purposes. In part, it accounts for some theoretical concepts that will help the reader understand the proceeding parts of the thesis paper. The other purpose that this chapter serves is to cite studies that are, in one way or another, related to this thesis. The review starts with a broad outlook, discussing the problem of travel-time prediction in a general sense, and then gradually narrows down to more intricate sides of this problem.

2.1 Travel Time Prediction

The travel-time prediction problem for a vehicle can be formulated in analogy to [2], as finding an accurate predictor $\Phi$ for a stochastic variable $T$ representing the time it takes for a vehicle to travel along a route.

$$T = T(r_0, r_1, t_0)$$

(2.1)

$T$ is defined by (2.1), where $r_0$ is the point where the travel begins, $r_1$ the point where the travel ends, and $t_0$ the time when the travel begins.

$$|\Phi(T) - \Phi^*| < \delta$$

(2.2)

The predictor sought for $T$ should be such that (2.2) holds for any $T$, where $\Phi^*$ is the actual time it takes for the vehicle to travel the route between $r_0$ and $r_1$, and $\delta$ is a number as small as possible.

In essence, the travel-time prediction problem for vehicles means estimating a vehicle's arrival time at some location, given this vehicle’s initial position and the time when it begins to travel. Introducing Equation (2.2) here merely serves to give a formal expression of this definition. This equation is not used...
to assess the performance of the travel-time predictors that were implemented for this thesis project.

### 2.2 Map-matching

Solutions to the travel-time prediction problem often incorporate solutions to the map-matching problem, which is elegantly defined in [3]. To adapt this definition to the setting of this thesis, we introduce the structures $\Upsilon$, $V$, and $E$.

\[
\Upsilon = \Upsilon(V, E) \tag{2.3}
\]

\[
V = \{v_1, v_2, \ldots, v_{N_v}\} \tag{2.4}
\]

\[
E = \{e_1, e_2, \ldots, e_{N_e}\} \tag{2.5}
\]

$\Upsilon$ represents a traffic network, where $V$ is the set of vertices (road intersections), and $E$ is the set of edges (roads that connect the intersections). The map-matching problem can now be interpreted as that of establishing which road $e^* \in E$ contains the actual position of a vehicle traveling in $\Upsilon$, given an estimate of the vehicle’s position at discrete points in time. The estimate of the vehicle’s positions is commonly provided by a Global Positioning System (GPS).

The reason map-matching is closely related to the problem of travel time prediction is that GPS points reported by a vehicle can be quite far from the true position of the vehicle when the signal was sent, especially in urban areas where buildings can corrupt the signal. Thus, if one wishes to base their travel time predictor on historical GPS-data, map-matching is necessary in order for the data to be useful. Simply put, the map-matching problem entails finding the true position of a vehicle, given a reported GPS position from the vehicle.

### 2.3 Hidden Markov Models for Map-matching

A Hidden Markov Model (HMM) uses an array of observations of an observable variable to induce a sequence of states for a variable that is not observable (hidden) but assumed to be dependent on the observable variable [4]. The HMM is suitable to use for the map-matching problem because the GPS coordinates of the traveling vehicle are efficiently observable, and if one lets
the hidden states be the coordinates’ matches in a traffic network, it is possible to calculate the most likely sequence of traffic network segments given a sequence of observed GPS coordinates. As described in [5], an observation sequence that is too long becomes a problem when using the HMM for streams of data, which is why the HMM is commonly applied on smaller consecutive batches of data, or windows, in online scenarios. An online map-matching solution based on the HMM is presented in [6], where the trade-off between accuracy and output delay as a result of the window size is handled by using a variable window size with an upper bound to reach an accuracy as high as possible, but not allowing for the observation sequence to become too long. The emission probability is a measure of how likely it is to observe a GPS-coordinate, given that the match between that coordinate and the chosen segment in the traffic network is correct. The emission probability in this article is modeled with a Gaussian distribution that takes into consideration the width of the road segment, the distance between the coordinate and the mapped point in the traffic network, and an estimate of the GPS error, along with a term that penalizes surpassing the speed limit, in order to distinguish between roads that are geographically close but have different speed limits. A Support Vector Machine (SVM) is trained to find the emission probabilities using a measure of the average time derivative of the vehicle’s momentum, and the discrepancy between the distance between GPS points and an interpolated path. The model was applied on a combination of rural and urban roads, reaching an accuracy of 0.92 with the bounded variable size window, while keeping the output delay at 100 seconds. In an offline-setting, a batch-learning HMM can be applied as described in [7], in a way that is analogous to the above in theoretical aspects, but differs in some parts of the practical implementation.

2.4 Map-Matching Software

Over the years, many different pieces of software have been developed for map-matching, some of which will be discussed here. OpenStreetMap (OSM) [8] is a database with free, open source, geographical maps that users are allowed to edit either manually or by uploading data from a connected device with GPS traceability. The project was released in 2004 and now has thousands of contributors. OSM has shown to, in some cases, provide more complete information than proprietary alternatives [9]. OSM was compared to the maps provided by Ordnance Survey (OS), which is the national mapping agency of the United Kingdom, in [10], regarding positional accuracy, attribute accuracy, consistency and completeness. The study only considered the London area,
which was deemed an illustrative representation of the database’s fitness since OSM started in the United Kingdom. Results showed that there was an 80% overlap between the OSM roads and the OS roads, and that in 25.4% of the U.K. roads, OSM was more accurate, while in 61.4%, OS was more accurate, the remaining percentages were left out because the area they represent are either coastline objects, grids cells smaller than 1km$^2$, or not featured in neither OSM nor OS. It is important to note that the study comparing OSM to OS was done almost 20 years ago now, so OSM would probably do better in the comparison today.

Figure 2.1: Example of how GPS Samples are matched to corresponding points on a most likely traffic road.

Barefoot [11] is a tool that builds on OSM to provide map-matching of GPS coordinates by using the HMMs described by [6, 7] for online and off-line matching, respectively. SharedStreets [12] provides a matching tool that builds on Barefoot but differs in that it uses its own tile-system for representing map units, and in using a document-oriented format for database storage. Graphhopper [13] provides a set of map-matching tools similar to those of Barefoot, these two pieces of software are compared in [14], in regard to a manual travel journal as the ground truth. Results show high accuracy for both tools, with Barefoot scoring slightly higher. Figure 2.1 shows an example of how GPS samples are snapped to a road using an online service called TrackMatching. [15]

2.5 Routing

Given a starting point and a finishing point on a map, finding the shortest driving route between the two translates to a graph shortest-path problem, when
using a traffic network. Finding a driving route between two points will be a necessary step in the experimental stages of this study, which is why some common approaches will be discussed here.

The Dijkstra algorithm is a boilerplate method for solving the shortest-path problem. This algorithm iterates through all the nodes between the starting node and the finishing node in the graph, at each step ranking the node’s neighbors based on how much they will elongate the path, and proceeding with the one that will elongate it the least. The A* algorithm is an adjusted implementation of Dijkstra’s algorithm, where a heuristic function that rewards nodes deemed likely to be in a shortest path is used to rank the neighboring nodes. In a traffic network setting, this heuristic is typically the time it would take for a vehicle to traverse the edge if it could travel exactly at the highest allowed velocity. Dijkstra and A* typically result in the same route, but with A* examining far fewer nodes, thus relaxing the algorithm’s complexity in both time and memory. [16]

2.6 Feature selection

Feature selection is the process of singling out variables that are relevant for the application at hand, from high-dimensional data sets. Feature selection is important in Machine Learning for various reasons: it speeds up computation time, alleviates memory storage, expedites data visualization, and improves prediction accuracy. Variable ranking is a method that can be used to filter out irrelevant features by computing a score for how significant each variable is. This score can be the Pearson correlation coefficient, which is defined based on the covariance between a variable and the target, divided by the square root of the respective variances of these two multiplied. The score function can also be based on the mutual information between each feature and the target, so that it becomes a measure of how the density of the features depends on the density of the target. This is called an information theoretic ranking criterion and is well suited for discrete types of data, due to the trouble of computing the densities of continuous variables. Another way of doing feature selection is using search strategies like forward selection or backward selection, where nested subsets of features are created through elimination or incorporation, resulting in a feature relevance hierarchy. Various other established methods for feature selection exist, like dimensionality reduction or clustering, and several manual steps can be taken to reach a suitable feature set, like applying domain knowledge or data set familiarity. [17]

For the problem of travel-time prediction, some relevant features are evi-
dent, like vehicle speed or time of day, while choosing the entire feature set might be less obvious. In a study that focused on suggesting driving routes where collisions are less probable [18], the estimated hourly vehicle count turned out to be the most valuable feature. Further, this model included information about proximal roads, such as the difference in number of lanes and in speed limit between each proximal road and the road in question, along with weather conditions and sun angle. Other than the vehicle count, features like the length of the road segment and the mean slope of the road segment were shown to be important, while features like snow depth and precipitation levels ranked low in relative importance. The feature importance was computed by how often the features were used in an ensemble of decision trees. Although [18] predicts a slightly different target than that of this thesis, the applications are deemed to be sufficiently similar for the findings to be acknowledged in this literature review.

### 2.7 Machine Learning

Machine Learning as a scientific field is concerned with computer algorithms that become better at doing an assignment as they gain experience. The algorithm is not explicitly programmed for the assignment, but to adjust its parameters through learning by repeatedly evaluating itself on examples (data points) provided. These examples contain features that hold some information about the domain the algorithm is meant to learn about, and a corresponding target value, which is regarded as the correct assignment value for the specific feature values. Presenting examples to the algorithm is called training, and there are many approaches to this, which is typically what differentiates Machine Learning methods. Machine Learning methods have proved to be very successful for applications like classification, regression, and pattern recognition, especially as computer processors have become increasingly powerful. [19]
Algorithm 1 Training a Supervised Machine Learning Regression Model

1: $X_{\text{train}}$ = Set of training examples.
2: 
3: $\mathcal{M}$ = Machine Learning model.
4: $\vec{w}$ = weight vector used by the model to make predictions.
5: 
6: for all $x \in X_{\text{train}}$ do
7: \hspace{1em} $y^*$ = predict($\mathcal{M}, \vec{w}, x$)
8: \hspace{1em} $y$ = true value($x$)
9: \hspace{1em} $e = y - y^*$
10: \hspace{1em} $\vec{w}$ = update weights($\mathcal{M}, e$)
11: end for

This thesis is confined to a subdomain of the quite large field of Machine Learning, focusing on supervised methods for the regression problem. A method being supervised loosely means that the examples used for training are labeled with correct answers that the algorithm can adapt its parameters in relation to, and applying Machine Learning on the regression problem means that the output space is continuous. Algorithm 1 shows the basic idea of how a supervised Machine Learning model is trained. The remainder of this chapter will discuss how different Machine Learning models work and how they have previously been applied to the problem of travel-time prediction.

2.7.1 Tree-based methods

The decision tree is an elemental method in Machine Learning where a tree structure is used to learn patterns. A tree in this context is an undirected graph with exactly one path connecting any two vertices [20]. The decision tree is constructed based on the features chosen for the application, by starting at the root of the tree and greedily adding leaves based on how much information they contain, which is repeatedly calculated from an information gain function dependant on an entropy measure. Decision trees can advantageously be used in ensemble learning, where the results of several learners are combined as a committee to yield an output more reliable than that of the individual learners. Learners can be combined by boosting or by bootstrap aggregating (bagging), where boosting means making sure learners learn different subsets of the feature space, while bagging means repeatedly taking samples from the data set with replacement (bootstrap samples), so that the learners train on slightly different data sets. Random forest is a method using bagged decision
trees where, at each node during construction, only a random subset of the features is allowed for selection, rather than the entire feature space. [21]

A comparison of tree-based methods for travel time prediction is provided in [22], featuring bagged trees, boosted trees, random forest and a more recent method called eXtreme Gradient Boosting (XGB). The XGB method differs from traditional boosted trees in that it penalizes the number of leaves in the models, which favors simpler implementations and makes the model less vulnerable to overfitting the data. The XGB model showed to consistently outperform all the other methods significantly, especially during peak traffic hours, when the travel times vary the most.

Due to the way a decision tree apportions the feature space by trying different combinations of variable values to see which most increases the information gain at each node, it has a built-in way of calculating feature importance. This can be used on any data set to get a measure of how much each feature contributes to reaching the target. [23]

2.7.2 Support Vector Machine

The Support Vector Machine (SVM) is a method for classifying data samples into one of several classes, by finding a hyperplane that separates the classes based on labeled samples during training [24]. The same logic can be used for regression, and the method is then called Support Vector Regression (SVR). The basic idea behind SVR is finding a mapping $f(x)$ that, when applied on any pattern in the training data, does not deviate more than a number $\epsilon$ from the target value, while being the flattest possible. This gives an minimization problem to which parameters called slack variables are incorporated, and these end up controlling the trade-off between how flat $f(x)$ is and how many times the same function is allowed to deviate from a pattern by more than $\epsilon$. Typically, a kernel function is used in place of dot products in the minimization problem, thus allowing the computations to take place only in a subset of the feature space, instead of in all of it. An advantage when using support vector regression is that there are only a few parameters that dictate the model, allowing for a quick implementation without having to manually tune too many criteria. [25]
Figure 2.2: Example of SVM separating samples from two different Gaussian distributions with different levels of regularization. The higher C is, the less the slack variables are allowed to relax the margin.

The SVM finds the best possible hyperplane, defined by a vector of weights $\vec{w}$, for separating different classes in the data. If we let the data set be $\{\vec{x}_i\}$ and the label set be $\{t_i\}$, then

$$\vec{w} = \sum t_i \alpha_i \vec{x}_i$$  (2.6)

where $\alpha_i$ is called a support vector and is in actuality a point $x_i$ in the data set such that $t_i(\vec{w} \cdot \vec{x}_i + b) = 1$, with $b$ being a bias term. [26] The weight vector $\vec{w}$ can be used in a process called Recursive Feature Elimination (RFE) in order to exclude features that contribute with little information to the model. RFE is an iterative algorithm that works as described in Algorithm 2. [27]
Algorithm 2 Recursive Feature Elimination

while There are more features than wanted do
    Train the SVM model.
    Compute ranking criterion for each feature based on the weight vector.
    Remove the feature with the lowest ranking.
end while

SVR performs well when faced with the travel-time prediction problem, as shown in [28], where accurate predictions were reached with SVR using both a linear kernel and a radial basis function (RBF) kernel. The SVR was compared to a historical mean prediction method, where travel time is predicted based on historical travel times from the same time of day and day of week, and to a current travel-time prediction method, where travel-time is predicted based on the current speed and the distance of the route to be travelled. The SVR predictor generally provided two-fold lower RME and RMSE values compared to the two other methods.

2.7.3 Artificial Neural Networks

In agreement with the biological neural networks of the brain, the basic unit of artificial neural networks (ANN) is the neuron. The artificial neuron is modeled as a function $f$ taking inputs $x_1, \ldots, x_n$ with associated weights $w_1, \ldots, w_n$, such that the neuron yields an output $y = f(x_1w_1 + \ldots + x_nw_n)$. Typically, the neuron is associated with an excitation threshold $\theta$ so that the output passed on to connecting neurons is $H(y - \theta)$, where $H$ is the Heaviside function. An ANN commonly learns through the backpropagation algorithm, where each weight $w_{ij}$ is updated in proportion to the gradient of the error with respect to that weight $\frac{\partial E}{\partial w_{ij}}$, so that each weight is adjusted in the direction that makes it decrease its contribution to the error the most. The error is calculated based on the difference between neuron output and target values. [29]
In what is called Deep Neural Networks (DNN), layers of neurons are stacked in order to find more complex decision boundaries. The minimum amount of layers any ANN can have are two, one for input and one for output. As described by [30], an ANN without hidden layers can classify data that is linearly separable, while an ANN with a single hidden layer can approximate any continuous function. An ANN with two hidden layers can approximate any function to arbitrary accuracy. This author also describes some empirical rules one can follow when choosing the width of the hidden layers, such as having the width of the hidden layers lie between the width of the input layer and the width of the output layer, the width of the hidden layers being $\frac{3}{4} \cdot \text{size of input layer} + \text{size of output layer}$, or having the width of the hidden layers simply be less than two times the width of the input layer.

A survey of some architectures is provided by [nn-types], presenting, among others, the autoencoder (AE) and the convolutional neural network (CNN). The AE is an ANN with one hidden layer of neurons between input and output layers, used to reconstruct the input by encoding it to an internal format and then rebuilding the original data from this representation, this mechanism makes the AE apt for filtering out features that carry fruitless information, and
creating a minimal representation of the data. The CNN is a type of ANN that is often applied to applications concerned with data that has a lattice-like topology, such as data sets consisting of images or videos. The most important properties of the CNN are that it is inspired by the special neurons in the visual cortex of the biological brain that each have a receptive field whose stimuli they respond to, and that it uses convolutions to pass information between layers in the network.

The study [31] implements an ANN architecture called Spatiotemporal Recurrent Convolutional Neural Networks (SRCN). This type of network is explained as a crossbreed of the Recurrent Neural Network (RNN) with Long short-term memory (LSTM) units, and the CNN. RNN with LSTM units have proven to be very efficient for learning long-term dependencies in sequences of temporal data, which is capitalized on in the SRCN and coupled with the CNN’s aptitude for learning spatial patterns in mesh-like frames, in order to obtain a model that capture both spatial and temporal properties of the data set. The CNN part of the model is trained on enmeshed images that portray states of traffic, whereon two layers of LSTM cells in the RNN are trained to learn the temporal properties of the traffic states, and finally a prediction is returned. The performance of the SRCN on GPS data is compared to that of four other models, namely SVM, CNN, stacked autoencoders (SAE) and LSTM RNN, who show similar predictive power in short time ranges, however, when applied to time ranges as long as 60 minutes, the SRCN model shows a far lower RMSE, and also a lower MAPE than the other models. The SAE and LSTM RNN models also show better performance than the SVM and the CNN in this time range.
Chapter 3

Method

This chapter sets the scene for the experiments that were done in this thesis project. It starts out with explaining how the data sets were collected, proceeding to how features were engineered and how the data sets were confined, and finally illustrates how the predictive models were made.

3.1 Data Preprocessing

This part of the Method section illustrates the components used to go from a query with some conditions on vehicle data, to having predictions of the travel time between previously unseen pairs of GPS samples, and how these components were coupled in a data pipeline.

Figure 3.1: Data pipeline
### Field | Explanation
--- | ---
id | Vehicle identifier
etime | Time when the GPS point was sent from the vehicle
lat | Latitude coordinate
lon | Longitude coordinate
speed | Instantaneous speed of the vehicle recorded by the speedometer
odometer | How far the vehicle has travelled in its lifetime

Table 3.1: Fields of table pulled from Data Lake

The goal of preprocessing GPS data is to turn it into vectors that a Machine Learning model can use for training. Figure 3.1 shows how the conditions posed on the data moved through the pipeline to generate predictions.

First a query (Figure 3.1 A) is sent to Scania’s Data Lake (Figure 3.1 B). This Data Lake is an on-premises Apache Hadoop cluster where Fleet Management System (FMS) data is stored in an Apache Hive data warehouse. Through FMS, an extensive variety of data from Scania’s vehicle fleet is stored in the Data Lake, allowing insights to be drawn from how vehicles have acted historically, or how they are acting in real-time.

The query sent to the Data Lake contains limitations like which geographical area the data should be in, during what period of time, how many different vehicles should be in the data set, and how many points in total. The resulting table has one row for each GPS sample, with the fields described in Table 3.1. This table is formatted by a Python script into the type of JSON document that the Barefoot map-matcher requires. This Python script also groups together points that are considered part of the same trip (Figure 3.1 D-E).

\[
id1 == id2
\]

\[
\Delta_{\text{odometer}} > 0
\]

\[
v > 0
\]

\[
\Delta t < 10h
\]

The trips are filtered out using a for-loop over samples that are grouped by vehicle ID and ascending timestamps. If a sample passes all of the conditions (3.1)-(3.4), it is considered to belong to the same trip as the previous
sample. Here, $\Delta$odometer is the odometer difference between two consecutive samples, $v$ is the speedometer velocity of the latter of two consecutive samples, $\Delta t$ is the time difference between the samples, and $id1$ and $id2$ are the respective vehicle IDs of the samples.

![Maps](image)

Figure 3.2: OSM way 31072790 highlighted in orange in (a)-(b), and in dark purple in (c). As seen in (c), this ID only refers to the northbound part of the highway.

The Barefoot map-matcher (Figure 3.1 F) returns sequences of matched samples (Figure 3.1 G), still grouped together as trips by the conditions (3.1)-(3.4). This map-matcher is applied because GPS samples do not always coincide with the vehicles’ true positions. The map-matching problem and the Barefoot map-matcher are reviewed closer in Section 2.2.

The matched GPS samples come in pairs, where each such pair holds an array of discrete road segment IDs connecting the two samples. Figure 3.2 shows an example of how road segments are identified by OSM. Finally, the matched trips are formatted into patterns that are suitable for Machine Learning (Figure 3.1 H-I). When predicting the travel-time between points that are not from the Data Lake, it is necessary to find the shortest sequence of Barefoot road segments that connects these points, so that a data sample with the same features as the training patterns can be made. Luckily, Barefoot includes a Dijkstra shortest-path algorithm for finding a route between two points in the Barefoot traffic network. This algorithm is used for finding routes between points that are not in the Data Lake.
3.2 Feature Engineering

The data used in Machine Learning consists of a set of patterns $X = \{x_i\}$, where $x_i = (f_1, \ldots, f_D)$ and $f_1, \ldots, f_D$ are features, and $d$ is the number of features used, which becomes the dimensionality of the patterns. The target variable to predict, $y$, was chosen to be the average velocity of the trip. So for each $x_i$ in the data set there is a corresponding $y_i$ that holds the true average speed for the trip encoded by that pattern. Different Machine Learning methods were applied to train on a portion of the data $X_{train}$ using the targets in the set of corresponding ground truths $Y_{train}$, and then generate predictions $y^*$ for a smaller portion of the data $X_{test}$, that were as close as possible to the values in $Y_{test}$.

Considering that the goal of this thesis is to predict the travel time of trips, it might not be completely obvious why the target variable was chosen to be the average velocity, and not simply the travel time. This choice was motivated by how the length of the trip would outweigh any other features if the travel time were the target. This choice is still coherent with the goal of the thesis, because the travel time can be calculated trivially from the average velocity and the trip length, so accurate average velocity predictions are essentially equivalent to accurate travel time predictions.

Three different categories of information were used to engineer features for the experiments. Information about when the trip took place was incorporated in order to detect temporal patterns, the identification number of the vehicle was used for distinguishing deviations due to differences in vehicle usage, and finally, information about how fast vehicles have travelled along given roads in the past was used to catch topographical variations that influence how fast vehicles travel.

3.2.1 Cyclical Features Describing Time

Most have probably been taught by experience that traffic varies heavily with certain temporal variables, peak-hour versus the early A.M., for example, or weekdays versus weekends. Further, in a lot of countries, whether the season is summer or winter makes a big difference in how traffic flows. These variables were modeled by three temporal features, the hour of day when the trip started ($h$), the day of week ($w$) and month ($m$) when the trip happened.

What is deceptive about these features is that they appear as if they live in a subspace of $\mathbb{Z} = \{1, 2, \ldots\}$ bounded by their highest allowed value, which would be 24 for $h$, 7 for $w$, and 12 for $m$. However, in the case of $h$ for example,
the furthest two values are the first hour and the last hour of the day, which are probably quite similar in the way they affect traffic. The features of weekday and month have the same problem.

What this means is that if a Machine Learning model is presented these types of features in a natural number representation, it might fail to learn some important relationships in the data, because it does not know that these features have a periodic nature in the real world. To deal with this, each temporal feature was transformed into a two-dimensional, cyclical representation, that communicates their periodicity. This way of engineering cyclical features was inspired by [32].

\[
\begin{align*}
    m & \rightarrow \left( \sin \left( \frac{2\pi m}{12} \right), \cos \left( \frac{2\pi m}{12} \right) \right) \\
    w & \rightarrow \left( \sin \left( \frac{2\pi w}{7} \right), \cos \left( \frac{2\pi w}{7} \right) \right) \\
    h & \rightarrow \left( \sin \left( \frac{2\pi h}{24} \right), \cos \left( \frac{2\pi h}{24} \right) \right)
\end{align*}
\] (3.5) (3.6) (3.7)

![Diagram](image)

Figure 3.3: The hours 03:00 and 21:00 are initially far apart, but lie much closer when the hour is transformed into two cyclical features.

The transformations (3.5)-(3.7) convert the features from living on a line segment to living on a circle, as shown in Figure 3.3. This means that these features no longer correspond to one-dimensional points in a closed interval, but to two dimensional points on a circle, determined by the time-value treated as an angle. This cyclical representation is an improvement because it more accurately models what the hour of day, the day of week, and the month mean in the real world.
3.2.2 One-hot Encoded Vehicle IDs

One-hot encoding is a way of transforming a categorical variable to a vector in a new space of higher dimension. In this thesis, one-hot encoding is applied to the identification numbers of the vehicles making trips. As mentioned earlier, it is desired to include the vehicle IDs in the training data because this might help the models learn how the average velocity varies between vehicles that are operated in different manners or for different purposes. This information needs to be encoded because, although the IDs are numbers, they should not be treated as continuous variables. This is because categorical variables cannot be compared relative to one another, and the numbers that represent the vehicle IDs are simply labels used for reference.

One-hot encoding a categorical feature means introducing one binary variable for each possible label that the categorical feature can take. These variables are stored in a vector, and a given label is encoded by such a vector with a 1 in the place of its corresponding binary variable, and zeros everywhere else.

Let us have a categorical feature that can take exactly one of \( L \) possible labels, \([l_0, \ldots, l_L]\). Any label \( l_k \) would then be represented by a vector \( \lambda \) with elements \( \lambda_l = \delta_{lk} \), where \( \delta \) is the Kroenecker delta function and \( k, l, i \in \{1, 2, \ldots, L\} \). This means that all the elements in the \( L \) elements long vector \( \lambda \) are zero except for the element \( \lambda_k \), which is 1.

**Example 3.2.1.** Let us assume, for simplicity, that there are only four different vehicles making trips in some data set, and let these IDs be “123”, “4242”, “619c”, and “car2”. In this strange identification system, the IDs have different numbers of characters, and there are numbers and letters mixed, thankfully though, this will not be a problem for the one-hot encoding.

<table>
<thead>
<tr>
<th>ID</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>[1, 0, 0, 0]</td>
</tr>
<tr>
<td>4242</td>
<td>[0, 1, 0, 0]</td>
</tr>
<tr>
<td>619c</td>
<td>[0, 0, 1, 0]</td>
</tr>
<tr>
<td>car2</td>
<td>[0, 0, 0, 1]</td>
</tr>
</tbody>
</table>

Table 3.2: How some hypothetical IDs would be one-hot encoded.

Now, when patterns are made, once it is seen that a trip was made by a certain vehicle, its ID is looked up in Table 3.2, and the corresponding encoded vector is used to represent this vehicle. This means that if a trip made
by vehicle “123” is preprocessed, the feature vector would become \([\text{feature}_1, \text{feature}_2, \ldots, 1, 0, 0, 0]\). In summary, one-hot encoding a categorical feature means introducing a binary feature for each label, representing whether this label is present or not in the current sample.

### 3.2.3 The Average Median Velocity Feature

In order for the machine learning model to learn about how much traffic varies on different roads, an associative array was built with each key being the OSM IDs for a distinct road segment, and the values being some descriptive statistic about how fast vehicles generally travel on them, based on observations in the data.

The associative array in Example 3.2.2 holds the average velocity for each OSM road segment, however any descriptive statistic can be used instead of the average. In the experiments made for this thesis, the median was chosen as the descriptive statistic, in order for this information to be less vulnerable to outliers in the velocity observations. The information from the associative array was made into a feature for a trip by summing the median of each road segment in the trip and dividing by the number of road segments in it, resulting in an average median velocity feature.

The average median velocity provides a summary of how fast vehicles have travelled along a given route in the past, which is assumed to influence how fast vehicles will travel along the same route in the future. For previously unseen points, this feature can still be calculated this way thanks to the Dijkstra router in Barefoot, which finds the road segments that optimally connect two points in the traffic network.

![Figure 3.4: Example of Traffic Network Graph](image-url)
Example 3.2.2. Let the graph in Figure 3.4 represent a small traffic network, where each edge in $e = \{A, B, C, D, E, F, G\}$, is a road segment, and each vertex is an intersection that allows a vehicle to begin traveling on a new road segment. For simplicity, let us assume that all edge lengths are equal to 1 so that

$$|e_i| = 1 \forall e_i \in e$$

(3.8)

An associative array is created with the OSM IDs as keys, in this example represented by the letters in $e$, and the associated average velocities $\langle v \rangle_{e_i}$ as values. Each average velocity is calculated as

$$\langle v \rangle_{e_i} = \left( \sum_{u \in U_i} \frac{s_u}{t_u} \right) \frac{1}{|U_i|}$$

(3.9)

where $s_u$ is the accumulated length of the edges in trip $u$ and $t_u$ is the total duration time of trip $u$. The set of recorded trips that pass through the edge $e_i$ is represented by $U_i = \{u_j\}$, meaning $|U_i|$ is the number of trips that have traversed $e_i$. The idea is that as $|U_i| \to \infty$, $\langle v \rangle_{e_i}$ will become an increasingly accurate representation of how fast vehicles generally travel on the edge $e_i$.

<table>
<thead>
<tr>
<th>edge traversal</th>
<th>trip length $s_u$</th>
<th>trip duration $t_u$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C \to D \to G$</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$F \to E \to G$</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>$A \to B \to D \to G$</td>
<td>4</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 3.3: Example of recorded trips.

Now assume that the trips described in Table 3.3 have been recorded, where the units of time and distance have been omitted for simplicity. Let us see how the associative array will be filled if these are the only trips considered, and the only edges of concern are those of the traffic network graph in Figure 3.4.
<table>
<thead>
<tr>
<th>OSM ID</th>
<th>( \langle v \rangle )</th>
<th>OSM ID</th>
<th>( \langle v \rangle )</th>
<th>OSM ID</th>
<th>( \langle v \rangle )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td></td>
<td>( A )</td>
<td></td>
<td>( A )</td>
<td>0.44</td>
</tr>
<tr>
<td>( B )</td>
<td></td>
<td>( B )</td>
<td></td>
<td>( B )</td>
<td>0.44</td>
</tr>
<tr>
<td>( C )</td>
<td>0.75</td>
<td>( C )</td>
<td>0.75</td>
<td>( C )</td>
<td>0.75</td>
</tr>
<tr>
<td>( D )</td>
<td>0.75</td>
<td>( D )</td>
<td>0.75</td>
<td>( D )</td>
<td>0.597</td>
</tr>
<tr>
<td>( E )</td>
<td></td>
<td>( E )</td>
<td>0.5</td>
<td>( E )</td>
<td>0.5</td>
</tr>
<tr>
<td>( F )</td>
<td></td>
<td>( F )</td>
<td>0.5</td>
<td>( F )</td>
<td>0.5</td>
</tr>
<tr>
<td>( G )</td>
<td>0.75</td>
<td>( G )</td>
<td>0.625</td>
<td>( G )</td>
<td>0.565</td>
</tr>
</tbody>
</table>

Table 3.4: associative array after considering the trip \( C \rightarrow D \rightarrow G \).

Table 3.5: associative array after considering the trips \( C \rightarrow D \rightarrow G \) and \( F \rightarrow E \rightarrow G \).

Table 3.6: associative array after considering all trips in Table 3.3.

Table 3.4-3.6 show how the associative array is updated after considering each trip, using the formula (3.9). As seen by how the array is updated, each trip’s contribution to the edges’ average velocities will be combined with that of previous overlapping trips, so that information from all trips that have passed through a certain edge will be used in determining that edge’s average velocity.

### 3.3 Data Sets

Section 3.2 explained how features were engineered to represent trips. This section describes how two sets of trips were selected from the vast collections of data that were available.
Two different data sets were used in the experiments, both of which were crafted by placing geographical constraints on the GPS samples in the data. The first data set includes trips made anywhere in Sweden south of the city of Uppsala. This data set will be referred to as the SoU data set, and its geographical distribution is shown in Figure 3.5 (a). The other data set is confined to vehicles traveling in either direction on the E20 road between the Swedish city of Kungsbacka (which is located just south of Sweden’s second biggest city, Göteborg), and the city of Helsingborg. This data set is referred to as the KBK-HBG data set, and it is visualized in Figure 3.5 (b). Both data sets are limited to trips that took place in 2018.

The area covered by the SoU data set was selected because it covers most of the major cities in Sweden, and if one could make accurate travel time predictions on such a data set, then that would be very attractive for those planning transport logistics in Sweden. However, it was understood from the beginning that it might be difficult to make accurate predictions on a data set with such high geographical variation.

The KBK-HBG data set was designed with the rationale that such a data set would be much easier to make predictions on since it only covers a single highway between two major cities, so travel patterns are expected to vary less.
Table 3.7: Data set statistics. *Train/Test Size* refer the number of patterns in the training and testing sets respectively. *Outliers* refers to the number of points discarded as outliers from the data set, in total (training and testing). *Vehicles* is the number of unique vehicles making trips in the data set, in total (training and testing).

Table 3.7 shows the number of points in each of the data sets, and how many different vehicles the samples originate from in each of them. There is a big difference between the number of vehicles, and it is due to the difference in geographical area between the data sets. Since the SoU data set spans over about half of Sweden, the number of vehicles examined had to be limited in order to not have a data set so big that it would take very long to train on it. The case of the KBK-HBG set is different in this way, since this data lies along a single highway between two not too distant cities. Every single sample in the Data Lake that satisfied the geographical constraints was collected for this data set, which is why many different vehicles are included in it.

The data sets were split into separate sets for training and testing by randomly shuffling the samples and separating 75% for training and 25% for testing. These proportions were chosen because they allow for a large portion of data to be used for training, which is desirable because it allows for the model to tune its parameters better. Further, given how the data sets are quite large, there are still many points in the testing set for evaluating the models, which is necessary in order to achieve statistical significance.

Only trips that took between 30 minutes and 5 hours to complete were considered, because such trips were deemed to be the most interesting in the context of planning truck trips, which is the cause that this thesis aims to contribute to. Samples that had a target value further than three standard deviations from the target’s mean were discarded as outliers. This is a common way to deal with outliers in Machine Learning and [33] provides a tutorial on how it is done in practice.

The decision to remove outliers was made because some unfeasible trips were found when looking through the data. There can be various reasons for why outliers are found in a data set, in the context of this thesis, the outliers were most likely a result of the preprocessing sometimes failing to group GPS
samples into reasonable trips. In Table 3.7 it can be seen that relatively few points were discarded as outliers in the KBK-HBG data set, and none in the SoU data set. It would be unsettling if too many points were discarded, because that would imply that information that should in fact be taken into account is being disposed of.

### 3.4 Predictive Models

Two distinct approaches were tried for reaching accurate prediction results, resulting in two different models that can be compared on different data sets. The first approach, referred to as the Global Model approach, mainly uses features that are independent of the location of the traveling vehicle, whereas the other approach, called the Local Models approach, combines many models that are each trained on data on a single road segment in the traffic network, and this way learns how the average travel speed on each road segment varies.

The Global Model approach was chosen because it is a quite straightforward way to make predictions, and it might be the first approach that strikes one’s mind when facing a prediction problem. The Local Models approach builds on the idea that different roads have their own intricate characteristics that affect how fast vehicles generally travel on them. This would suggest that many expert models, each trained on short road segments, could profitably be combined to explain how a vehicle’s velocity varies along a trip.

#### 3.4.1 Global Model Approach

The Global Model uses regularized Linear Regression to learn how to predict the average velocity for trips. Linear Regression, as described in [34], aims to find a matrix of regression coefficients \( \hat{\beta} \), whose product with the pattern matrix \( X \), is as close to the vector of true values, \( y \), as possible. The matrix \( \hat{\beta} \) is found as the \( \beta \) that minimizes the Least Squares function.

\[
\text{Least Squares}(\beta) = (y - X\beta)^T \cdot (y - X\beta)
\]

By taking the derivative of (3.10) with respect to \( \beta \) and equating to zero, one can find the optimal coefficient matrix \( \hat{\beta} \).

\[
\hat{\beta} = (X^T X)^{-1} \cdot X^T \cdot y \tag{3.11}
\]

In traditional Linear Regression, the matrix \( \hat{\beta} \) in (3.11) is simply multiplied with \( X \) in order to infer \( y \). Regularized Linear Regression refers to finding
a similar matrix of coefficients but with the magnitudes of the values in it constrained.

Ridge Regression is a form of Regularized Linear Regression where the sum of the squares of the coefficients in $\hat{\beta}$ are bounded by a parameter $\alpha$. This increases bias in the regression, which is useful when the goal of the regression is to make predictions on previously unseen data points. With more bias, the regressor’s ability to generalize improves, since the regressor is not as closely fitted to the training data as it would be without regularization.

$$\hat{\beta}_{\text{Ridge}} = \left( X^T X + \alpha I \right)^{-1} \cdot \left( X^T \hat{y} \right) \cdot \hat{\beta} \tag{3.12}$$

In the context of this thesis, $X$ would be a $N \times d$ feature matrix where $N$ is the number of trips and $d$ is the number of features used to represent each trip, and $y$ would be a vector holding the true average velocities for the trips in $X$. So the Global Model makes predictions by multiplying the feature matrix with $\hat{\beta}_{\text{Ridge}}$ in (3.12). Appendix B shows the steps involved in deriving the Ridge Regression in closer detail.

The biasing parameter $\alpha$ used in the Ridge Regression was chosen with Leave-One-Out Cross-Validation (LOOCV). This means that the training data was split into as many folds as there are training patterns. These folds were each left out once as validation data while the other folds were used to train the model during the validation. The $\alpha$ that gave the lowest average MSE error on all folds was used to train the final model and evaluate it on the testing data. LOOCV is an extreme version of K-Fold Cross Validation, having K set to the number of patterns in the training data, minus one. The reason LOOCV was chosen over K-Fold Cross Validation was how K-Fold Cross Validation can be biased towards giving higher errors, especially if K is low, since less of the training data is used, resulting in inferior predictors to evaluate.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Explained in Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>hour&lt;sub&gt;sin&lt;/sub&gt;</td>
<td>3.2.1</td>
</tr>
<tr>
<td>hour&lt;sub&gt;cos&lt;/sub&gt;</td>
<td>3.2.1</td>
</tr>
<tr>
<td>weekday&lt;sub&gt;sin&lt;/sub&gt;</td>
<td>3.2.1</td>
</tr>
<tr>
<td>weekday&lt;sub&gt;cos&lt;/sub&gt;</td>
<td>3.2.1</td>
</tr>
<tr>
<td>month&lt;sub&gt;sin&lt;/sub&gt;</td>
<td>3.2.1</td>
</tr>
<tr>
<td>month&lt;sub&gt;cos&lt;/sub&gt;</td>
<td>3.2.1</td>
</tr>
<tr>
<td>average median velocity</td>
<td>3.2.3</td>
</tr>
<tr>
<td>vehicle id</td>
<td>3.2.2</td>
</tr>
</tbody>
</table>

Table 3.8: Features used in Global Model

As seen in Table 3.8, the Global Model uses all of the Features described in section 3.2. This means that the feature space has dimension 7 + \( n_v \) where \( n_v \) is the number of distinct vehicles in the data, because there are 7 continuous features and \( n_v \) features for the one-hot encoded vehicle ID.

### 3.4.2 Local Models Approach

In the Local Models Approach, an associative array is built by iterating through all of the trips that make up the data set. The average speed is calculated just like in Example 3.2.2 for each road segment, but instead of simply recording a statistic of the speed observations, the observations are used as patterns for a Random Forest regression model that learns to predict the average velocity of its designated road segment.
Algorithm 3 associative array with Random Forest Regressors

1: arr = initialize new hashmap
2: 
3: for all trips in the training data do
4:   for all road segments in the trip do
5:     arr.get(road segment).get("feature vector").append(features(trip))
6:     arr.get(road segment).get("target vector").append(average velocity(trip))
7:   end for
8: end for
9: 
10: for all road segments in arr.keys() do
11:   x = arr.get(road segment).get("feature vector")
12:   y = arr.get(road segment).get("target vector")
13:   rf = initialize new random forest regressor
14:   rf.fit(x, y)
15:   arr.get(road segment).put(key = "regressor", val = rf)
16: end for

Algorithm 3 shows the basic idea of how the local models are created and added to an associative array. The procedure can roughly be divided into two parts, that of adding data points to each road segment in the array, and that of training regressors for each road segment on their respective data sets.

```
{
  "1": {
    "feature vector": [
      [0, -1, -0.4, -0.9, -0.7, -0.7, 0, 0, 1],
      [-0.5, -0.9, 0.4, -0.9, 1.2, -1, 0, 1, 0]
    ],
    "target vector" : [88.4, 84.2],
    "regressor" : <Random Forest Regressor Object>
  },
  "2": {
    "feature vector": [
      [-0.6, -0.5, 0, 1, 0.5, -0.9, 0, 0, 1],
      [-1, 0, 0.6, 1.0, 0.7, -0.7, 0, 1, 0]
    ],
    "target vector" : [83.87, 86.65],
    "regressor" : <Random Forest Regressor Object>
  }
}
```

Figure 3.6: Example visualization of associative array with Local Models
To give a clearer view of what this associative array looks like, an example is shown in Figure 3.6. From this figure one can see that the road segments with IDs 1 and 7 have been travelled on. The first six values in the feature vectors correspond to month$_{\text{sin}}$, month$_{\text{cos}}$, weekday$_{\text{sin}}$, weekday$_{\text{cos}}$, hour$_{\text{sin}}$, and hour$_{\text{cos}}$, while the last three values corresponds to the one-hot encoding of the vehicle id, meaning that in the data set in this example, there are only three different vehicles.

**Algorithm 4** Predicting the Average Velocity of a previously unseen trip

**method** Predict (trip, arr)
1: \( y^* = 0 \)
2: \( n = 0 \)
3: **for all** road segments **in** trip **do**
4: \( \text{rf} = \text{arr}.\text{get}(\text{road segment}).\text{get}("\text{regressor}") \)
5: \( x = \text{features}(\text{trip}) \)
6: \( y^* = y^* + \text{rf}.\text{predict}(x) \)
7: \( n = n + 1 \)
8: **end for**
9: **return** \( y^*/n \)

When predicting the average velocity for a new trip with the Local Models, a prediction is made for each road segment the trip will pass through. These individual predictions are averaged over the entire trip, and the result is the average velocity prediction of the trip.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Explained in Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>hour$_{\text{sin}}$</td>
<td>3.2.1</td>
</tr>
<tr>
<td>hour$_{\text{cos}}$</td>
<td>3.2.1</td>
</tr>
<tr>
<td>weekday$_{\text{sin}}$</td>
<td>3.2.1</td>
</tr>
<tr>
<td>weekday$_{\text{cos}}$</td>
<td>3.2.1</td>
</tr>
<tr>
<td>month$_{\text{sin}}$</td>
<td>3.2.1</td>
</tr>
<tr>
<td>month$_{\text{cos}}$</td>
<td>3.2.1</td>
</tr>
<tr>
<td>vehicle id</td>
<td>3.2.2</td>
</tr>
</tbody>
</table>

Table 3.9: Features used in the Local Models

In the Local Models, the average median velocity feature is omitted because this approach aims to find this information by accumulating the predictions of the individual regressors found on each road segment. The feature
space for each Local Model has dimension \(6 + n_v\) with \(n_v\) being the number of vehicles in the data set.

### 3.5 Performance Metrics

In the Results section that follows, a few different metrics are used to summarize the errors of a model, the most commonly known one being the Root Mean Square Error (RMSE).

\[
\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (y_{\text{pred},i} - y_i)^2}{N}}
\]  
(3.13)

Another less known error metric used is the Mean Absolute Error (MAE), which represents the average absolute prediction error.

\[
\text{MAE} = \frac{\sum_{i=1}^{N} |y_{\text{pred},i} - y_i|}{N}
\]  
(3.14)

These two absolute metrics are complemented with the Mean Absolute Percentage Error (MAPE), in order to get a summary of the relative prediction errors.

\[
\text{MAPE} = \left( \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_{\text{pred},i} - y_i}{y_i} \right| \right) \cdot 100\%
\]  
(3.15)

In order to evaluate how well the models explain the variance in the data, the *Coefficient of determination*, also known as the \(R^2\) score, was used.

\[
\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i
\]  
(3.16)

\[
R^2 = 1 - \frac{\sum_{i=1}^{N} (y_i - y_{\text{pred},i})^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2}
\]  
(3.17)

The \(R^2\) score is defined as in 3.17, and it describes how much of the variance in the target variable that the regressor can explain, given its predictions. The closer this score is to 1, the more variance is accounted for. The denominator in 3.17 is a value proportional to the variance in the true values, and the numerator is the squared sum of residuals for the predictions.

In (3.13)-(3.17), \(y_{\text{pred},i}\) are predicted values, \(y_i\) are true values, and \(N\) is the size of the data set.
Chapter 4

Results

This part of the report describes the experiments that were conducted with the methods covered in the previous chapter. First, the relevance of the feature set was evaluated, in order to learn if the features described in section 3.2 in fact provided appropriate information that could be used to make predictions. Next, the Global Model approach was evaluated on the SoU data set, to learn how well a predictor can perform when presented a highly varied data set. Finally, the Global Model approach was compared to the Local Models approach on the smaller KBK-HBG data set, with the goal of finding which approach is better suited for future studies. The results are reflected on briefly in-place, and reviewed in a broader context in the Discussion chapter.

4.1 Feature Importance

To find which of the features were most important, some techniques for measuring feature importance were applied to the Global Model. Recursive Feature Elimination was used to rank the continuous features based on their importance for predicting the target this ranking was complemented by the feature importance calculations of a Decision Tree that evaluates each feature based on the information gain it contributes. The SVM applied in RFE used a Linear kernel and $C = 1$. 
### Table 4.1: Performance metrics for Global Model on KBK-HBG data.

<table>
<thead>
<tr>
<th>Feature</th>
<th>RFE ranking</th>
<th>Decision Tree score</th>
</tr>
</thead>
<tbody>
<tr>
<td>average median velocity</td>
<td>1</td>
<td>0.5692</td>
</tr>
<tr>
<td>month&lt;sub&gt;sin&lt;/sub&gt;</td>
<td>2</td>
<td>0.0758</td>
</tr>
<tr>
<td>weekday&lt;sub&gt;cos&lt;/sub&gt;</td>
<td>3</td>
<td>0.0398</td>
</tr>
<tr>
<td>weekday&lt;sub&gt;sin&lt;/sub&gt;</td>
<td>4</td>
<td>0.0724</td>
</tr>
<tr>
<td>month&lt;sub&gt;cos&lt;/sub&gt;</td>
<td>5</td>
<td>0.0833</td>
</tr>
<tr>
<td>hour&lt;sub&gt;sin&lt;/sub&gt;</td>
<td>6</td>
<td>0.0757</td>
</tr>
<tr>
<td>hour&lt;sub&gt;cos&lt;/sub&gt;</td>
<td>7</td>
<td>0.0837</td>
</tr>
</tbody>
</table>

In Table 4.1, it can be seen that the average median velocity is by far the most important feature, while all the other features combined make a contribution of the same magnitude. Since the target variable is the true average velocity, it makes sense that a statistic of how fast vehicles have travelled along the same route in the past (average median velocity), is a very important feature.

The idea behind using these features was that the average median velocity would set the general course for the predictions, and the predicted values would be fine-tuned by features describing the circumstances under which the trip was made. If it would have turned out that the average median velocity had a too high importance score, for example 0.97, then that would suggest that the other features are redundant. So the results in Table 4.1 show that the features describing hour, month, and weekday indeed contribute to predicting the target.

The importance of the categorical feature representing the ID of the truck was evaluated separately by comparing the performance of the model with and without this feature included in the patterns.

### Table 4.2: Performance metrics for the Global Model on SoU data without vehicle ID as a feature.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Training Data</th>
<th>Testing Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE (km/h)</td>
<td>11.3746</td>
<td>11.4111</td>
</tr>
<tr>
<td>MAPE (%)</td>
<td>36.9035</td>
<td>38.0626</td>
</tr>
<tr>
<td>MAE (km/h)</td>
<td>9.0218</td>
<td>9.0696</td>
</tr>
<tr>
<td>R²</td>
<td>0.739</td>
<td>0.739</td>
</tr>
<tr>
<td>Metric</td>
<td>Training Data</td>
<td>Testing Data</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td>RMSE (km/h)</td>
<td>10.2306</td>
<td>10.3732</td>
</tr>
<tr>
<td>MAPE (%)</td>
<td>31.4372</td>
<td>31.7647</td>
</tr>
<tr>
<td>MAE (km/h)</td>
<td>7.9174</td>
<td>8.0245</td>
</tr>
<tr>
<td>$R^2$</td>
<td>0.789</td>
<td>0.785</td>
</tr>
</tbody>
</table>

Table 4.3: Performance metrics for the Global Model on SoU data with vehicle ID as a feature.

Table 4.2 shows the performance of the model when it does not use the vehicle ID as a feature. Comparing these figures to those in Table 4.3, we see that all of the metrics improve significantly. Note that this improvement happens despite the dimensionality of the feature space growing from 7 to 173 because of the one-hot encoding.

### 4.2 Local Models and Global Model Compared

The ensemble of Local Models was implemented and tested on the KBK-HBG data, because this data set is concise enough for training a regressor for each road segment in it. Training complex regressors scales very poorly in time, and this is something that restricted the way the Local Models could be used. This is why the Local Models were not tested on the SoU data set, since training the regressors would be very tedious, and the regressors would need to be very naive. In this experiment, a Random Forest Regressor made up of 50 decision trees was trained on each road segment in the KBK-HBG data set.
Figure 4.1: Count of observations on the road segments in the KBK-HBG data set.

An overview of the distribution of observations over the road segments is useful here, in order to get a clearer idea of how the Local Models approach unfolds in practice. Figure 4.1 shows the distribution of how many true velocity values were observed on the road segments in the KBK-HBG data set. One can see that the vast majority of the road segments have over 1000 observations. In fact, there were only four road segments that had fewer than 250 observations, and these four were excluded when doing training and prediction. It can be seen that the Local Models in general had many observations to train on.

It is of interest to know how well the Local Models can compete, in terms of predictive performance, with the Global Model. To learn about this, the Global Model was also evaluated on the KBK-HBG data set. As discussed in 3.4.1, the Global Model is a Ridge Regressor. The biasing parameter $\alpha$ for this regressor was chosen to 0.8 using LOOCV. A comparison of the performance metrics for the two models on this data set follows.
<table>
<thead>
<tr>
<th>Metric</th>
<th>Training Data</th>
<th>Testing Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE (km/h)</td>
<td>3.399</td>
<td>4.023</td>
</tr>
<tr>
<td>MAPE (%)</td>
<td>2.858</td>
<td>3.072</td>
</tr>
<tr>
<td>MAE (km/h)</td>
<td>1.851</td>
<td>1.998</td>
</tr>
<tr>
<td>$R^2$</td>
<td>0.378</td>
<td>0.301</td>
</tr>
</tbody>
</table>

Table 4.4: Performance metrics for the **Global** Model.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Training Data</th>
<th>Testing Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE (km/h)</td>
<td>4.827</td>
<td>5.096</td>
</tr>
<tr>
<td>MAPE (%)</td>
<td>4.584</td>
<td>4.587</td>
</tr>
<tr>
<td>MAE (km/h)</td>
<td>2.749</td>
<td>2.870</td>
</tr>
<tr>
<td>$R^2$</td>
<td>-0.048</td>
<td>-0.055</td>
</tr>
</tbody>
</table>

Table 4.5: Performance metrics for the **Local** Models.

A summary of the errors of the predictions made by the Global Model is shown in Table 4.4, and for the Local Models in Table 4.5. It is clear that the Global Model predicts the average velocity with less error, both in absolute and relative terms. What is interesting to note however, is that the ensemble of Local Models seems to generalize better to previously unseen points, in relation to its training error. What this means is that there is a larger difference between the training errors of the Global Model, compared to those of the ensemble of Local Models, which suggests that the ensemble of Local Models is not as tightly fitted to the training data as the Global Model. The $R^2$ scores are considerably low in both cases, especially in the case of the Local Models. The meaning of these scores is reviewed closer in section 5.3.
Figure 4.2: Distribution of Predictions, True Values, and Errors for the Global Model in (a), (c), and (e), and for the ensemble of Local Models in (b), (d), and (f).

The distributions of the predictions, true values and errors for both the Global Model and the ensemble of Local Models are shown in Figure 4.2.
Subfigures (a)-(b) show traditional histograms of the counts of values in discrete buckets, while subfigures (c)-(f) show Kernel Density Estimations. A Kernel Density Estimation (KDE) is similar to a histogram, and is in this case used for the same purpose, that is visualizing a distribution. The KDE estimates a density function for the distribution it is provided, with which it can plot a smooth curve that depicts the shape of the distribution, without depending on the number of buckets, like traditional histograms do.

Examining Figure 4.2 (a)-(b), one can see that both models account for only part of the variance of the true values, especially the ensemble of Local Models where all of the points in the predictive distribution are quite close to its mode. Figure 4.2 (c)-(d) shows that the modes of the predictive distributions coincide quite well with the modes of the true distributions, for both models. Again, however, it can be seen that the predictions of the ensemble of Local Models appear to be biased to stay closed to the mode of the distribution. This suggests that the model is too simple to pick up the relations in the data that result in the higher variance that the true distribution of targets shows.

In Figure 4.2 (e)-(d), it can be seen that the absolute errors, $|y_i - \hat{y}_{\text{pred},i}|$, are for the most part between 0 km/h and 1 km/h for both models. In the Global Model’s distribution, the absolute error rarely exceeds 2 km/h, which is quite satisfactory performance. For the ensemble of Local Models, we see that there are more error measures in the interval [2, 5] km/h.

### 4.2.1 Statistical Comparison of the Local Models and the Global Model

Up until this point, the approaches of the Global Model and the Local Models have been compared mainly based on performance metrics and visualizations of their respective predictions. Here a two-sample Kolmogorov-Smirnov (KS-2) test is presented, to study how the two prediction approaches compare statistically.

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_0$</td>
<td>$\forall e \in \mathbb{R}, L(e) = G(e)$</td>
</tr>
<tr>
<td>$H_1$</td>
<td>$\exists e \in \mathbb{R}, L(e) \neq G(e)$</td>
</tr>
</tbody>
</table>

Table 4.6: Hypotheses

The KS-2 test is here designed according to the description in [35]. In the context of this thesis project’s experiments, $L$ and $G$ in Table 4.6 correspond to
the unknown distribution functions of the prediction errors of the Local Model and the Global Models, respectively. So this KS-2 test aims to determine at what significance level it is possible to reject the hypothesis \( H_0 \), which claims that the prediction errors of the Local Models and the Global Model belong to the same distribution.

\[
D_{KS} = \sup_e |L(e) - G(e)| \tag{4.1}
\]

In the KS-2 test, the D-statistic \( D_{KS} \) is calculated as the largest distance between the distribution functions \( L \) and \( G \), as shown in 4.1. This statistic can be translated to a p-value, which describes how much evidence there is in favor of the null-hypothesis \( H_0 \), and at what significance level it can be rejected.

<table>
<thead>
<tr>
<th>Data</th>
<th>( L ) Sample size</th>
<th>( G ) Sample size</th>
<th>( D_{KS} )</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>12 625</td>
<td>12 625</td>
<td>0.216</td>
<td>( 1.04 \cdot 10^{-257} )</td>
</tr>
<tr>
<td>Testing</td>
<td>3 511</td>
<td>3 511</td>
<td>0.215</td>
<td>( 3.79 \cdot 10^{-71} )</td>
</tr>
</tbody>
</table>

Table 4.7: Results of Two-Sample Kolmogorov-Smirnov Tests on training errors and testing errors. In each KS-2 test, all of the prediction errors were used. Since the Local Models approach and the Global Model approach were trained on evaluated on the same data, their sample sizes are the same in both tests.

The result of two KS-2 tests are shown in Table 4.7. It is clear that the p-value is very close to 0 in both cases, which means that the tests cannot find much evidence for \( H_0 \). This means that it is possible to reject the hypothesis that the prediction errors belong to the same distribution, with very high confidence. The p-values in Table 4.7 are calculated from \( D_{KS} \) using the statistics module of the Python library SciPy [36].
Chapter 5

Discussion

In this chapter I discuss some of the difficulties experienced through the course of this thesis. I start by describing some particulars that I found when working with the type of data that this thesis is concerned with. In continuation, I elaborate on what I think could have been done differently to get more satisfactory results, and finally, I propose some ideas for related future work.

5.1 Data

When working with GPS data to make patterns that represent trips between stops, certain phenomena emerge that complicate the analysis. A dilemma arises when making the trips where, on one hand, one would wish for the trip patterns to as accurately as possible approximate true trips made by the vehicles, and to reach this one would be quite strict when filtering out trips. On the other hand, if many points are discarded because they cannot be matched with any trip, then the needed number of GPS samples per trip pattern rises and more data is needed.

A difficulty in examining longer trips lies in the noise that is produced by trips passing through sites that constitute some form of logistics hub, especially when the data is generated by trucks that tend to drive with objectives different than those of private vehicles. What is meant by this is that it is difficult to pick out a route along which trucks mainly transit rather than start or end journeys, because there can be some minor hubs along what appears to be a transit road that add noise to the data. Traffic in cities or hubs generate noise because vehicle behavior is significantly more varied in such places than in transit, and there are many factors that are hard to account for, such as pedestrians or traffic lights. This noise problem can be dealt with by picking out the IDs of specific
roads that do not allow stopping, and limiting the data set to those roads.

Another detail about the data that is worth discussing is the sampling rate of the trucks, which lay between 10 minutes and 10 hours depending on the vehicle. This had consequences for the analysis because even the sampling rate of 10 minutes exceeds the time it takes to traverse any of the OSM identified road segments, which is why the travel-time of specific road segments could not be observed directly, which in turn was why, in the predictive models, the observed velocities at each road segment were averaged as described in Example 3.2.2.

5.2 Method

The map-matching that Barefoot provided worked well, in the sense that it for the most part succeeded in finding reasonable corresponding points in the traffic network for the GPS samples. Further, Barefoot is an open source project with well-written code, thorough documentation and attendant contributors, which provided pleasant conditions for the code to be modified to suit the needs of this thesis. The only disturbance produced by the Barefoot map-matcher lay in how long it took to match large collections of samples, which put a constraint on how many different collections of GPS-points could be examined, since the map-matching had to be run overnight. This inconvenience could have been dealt with by distributing the computations of Barefoot on a cluster of machines, which Barefoot supports. Although parallelizing Barefoot was not prioritized in this thesis, I recommend that those using Barefoot for map-matching large amounts of data in the future consider doing so.

What might have made the comparison between the different Machine Learning approaches more revealing, is the use of the same regressor in both the Global Model and the Local Models. Ridge Regression was applied in the Global Model in order to learn if such a simple algorithm could get satisfactory results in the context of travel-time prediction. The reason the Ridge Regressor was not implemented in the Local Models approach was due to the tuning needed for finding a suitable bias parameter. Since there are so many regressors combined in this approach, it was decided that a regressor that could have the same parameter setting for each Local Model would be used. The Random Forest was deemed appropriate because it only has a single, not too sensitive parameter to tune, namely the number of Decision Trees. Another reason for using the Random Forest model in the Local Models was its robustness to overfitting. This robustness, in combination with the large number of speed observations on nearly all road segments, allowed for a large number of trees
to be used, without concern for the models adapting too closely to the data. See, for example, [37] for a more detailed discussion on how to determine the number of Decision Trees to use in a Random Forest model.

As mentioned in Section 3.4.1, LOOCV was used to determine the bias parameter in the Ridge Regressor of the Global Model approach. A disadvantage to using LOOCV is how computationally heavy this method is, especially when the data set is large, since the model is evaluated once for every point in the data. If the data set is large, as is the case in this thesis project, LOOCV can be feasible if the Machine Learning model is sufficiently fast. In my experiments, applying LOOCV to the Ridge Regressor on the SoU data set took less than an hour, which suggests that the Ridge Regressor is fast enough for LOOCV on data sets of this capacity. Further, there is high variance in LOOCV, because the outputs from each fold are highly correlated. This means one should be cautious when using LOOCV as a way of measuring error, because the average error over all folds can be inaccurate. [38] It is possible that, in the case of my implementation, considering how large the data sets are, it would have sufficed to simply use a designated set of data points extracted from the training set to tune parameters. In spite of this, LOOCV was applied because it was regarded a more rigorous method, and using it proved to be feasible in the experiments.

5.3 Evaluation

A performance metric used that calls for closer discussion is that of the $R^2$ score. As is the case with any Machine Learning metric, one should keep in mind the nature of the data when interpreting it. This means that, although this score provides a useful summary of some of the models’ properties, it does not single-handedly encapsulate them. Considering the $R^2$ scores in Table 4.4 - 4.5, it is important to take into account that the variance in the true values of the target variable is very small (see Appendix A), which means that the denominator in (3.17) will make the $R^2$ score decrease. When reviewing the $R^2$ scores in Table 4.2 - 4.3, one sees that they are significantly higher than the previously mentioned ones, which at first might make one think that these models perform better. However, when examining the distribution of the true target values in this data set, it becomes apparent that this difference in the $R^2$ score is most likely a result of higher variance in the data set.

To have a model make predictions on a data set that has been excluded from the training phase is a standard way of evaluating Machine Learning mod-
els. Nevertheless, averaging testing errors over several testing sets increases the statistical significance of the results. An interesting addition to this thesis project could have been to evaluate each of the Machine Learning models on more data sets, to get a better idea of how these models compare statistically.

As a final note on the evaluations used in this thesis, something that advocates the statistical significance of the comparison of the Local Models and the Global Model are the results of the Two-Sample Kolmogorov-Smirnov test in Section 4.2.1. These results suggest that there is a statistical difference between these two approaches, and that the discrepancy in predictive performance between them, is not accidental.

5.4 Predictive Performance

Considering the high error statistics in Tables 4.2-4.3, where the Global Model was evaluated on the quite widely distributed SoU data set that spans about half of Sweden, it seems that it is quite difficult for a Machine Learning Model to accurately predict travel time when the data varies a lot geographically. The Local Models described in 3.4.2 make predictions on an almost atomic level of the traffic network, but merely averaging the local predictions along a route seems to be a too simple approach for predicting travel time, taking into account the results in 4.2.

There are several easily done changes that could be made to the ensemble of Local Models that might improve its performance. The most apparent part to change about this approach would be the way the predictions from the regressors are averaged. Instead of simply averaging over the number of predictors, one could give higher importance to the predictions of certain regressors by assigning individual weights to them. These weights could be based on how well the regressor predicts a validation data set, or how long the road segment that the regressor is trained for is.

Something that could boost the performance of any travel-time predictor is to use information from some external data sets holding climate statistics per day. What might also be promising is to incorporate data describing traffic levels into the model, for example taken from some traffic authority.

Although it might be possible to improve the ensemble of Local Models to reach better predictive power, I think that a more effective way to get accurate predictions would be to explore what level of locality is optimal when using ensembles of models. This idea is based on how the Global Model performed quite well when applied to a data set with less geographical variance, which suggests that, maybe, a good way to predict travel time would be to use
equivalents of my Global Model, trained on different geographical areas, in conjunction.

5.5 Future Work

Based on the results seen in this thesis, something that would probably yield quite accurate predictions would be to construct a specialized system for identifying roads, based on that of OSM but with several OSM road segments combined into broader entities that would represent entire routes between cities or hubs. Even better would be if these entities would represent the routes between known stop points that had been found in the data, for example through clustering. This would result in a network of identified roads that connect all major stop points in the examined area. I believe that if separate regressors were trained to predict the average velocities for each of these entities, the regressors would, when combined, give quite accurate predictions. This proposal is supported by how the Global Model in this thesis preformed quite well on the KBK-HBG data set, which connects the city of Göteborg to the south of Sweden. Maybe many similar models trained on different geographical areas could be combined to give accurate predictions for trips traversing larger areas.

5.6 Ethics and Sustainability aspects

The most apparent ethical aspects that arise in the context of this thesis are those of injury risks in traffic, and the environmental impact of vehicles powered by fossil fuels. The objective of this thesis is aligned with any goal of minimizing injuries in traffic, since better planned trips alleviate drivers’ stress, which allows them to make safer decisions in traffic.

Sustainability has been a paramount motivator for this thesis project from the start, since better travel time predictions allow for trips to be arranged in a way that is more environmentally friendly, encouraging drivers to focus more on sustainable driving and less on reaching their destination at an arranged time.

Finally, there are always ethics involved in how to wield data related to people, who in the context of this thesis are the truck drivers. When doing experiments, I had no access to any information about who was driving which vehicle at which time. All analyses performed were based on information that had been automatically reported by the trucks, regardless of their drivers.
Chapter 6

Conclusion

The objective of this thesis was to find a way to make accurate predictions of travel time using historical GPS data, and to investigate how this GPS data should be preprocessed for the problem of Travel Time prediction. It was learned that some parts of the preprocessing, like mapping GPS-points to real roads, can be done quite painlessly with robust open source tools like Barefoot. What was less apparent was how to group the GPS-points into viable real-life trips, without losing too much data. Regarding feature engineering, it was found that some quite rudimentary information about the vehicles sending GPS points could be organized into an adequate set of features for travel time prediction.

Perhaps the most valuable insight that came from this work was that it is difficult for a regressor to predict travel times of all vehicles traveling in a broad geographical area. This indicates that future endeavors of predicting travel times of vehicles should focus on combining regressors trained on smaller disjoint geographical areas, because this showed quite promising results in this thesis.
Bibliography


[27] Isabelle Guyon et al. “Gene Selection for Cancer Classification using Support Vector Machines”. In: Machine Learning, volume 46 (2002).
Appendix A

Feature and Target Distributions

Figure A.1: Distributions of variables in the SoU Data Set. Before splitting into separate sets for training and testing.
Figure A.2: Distributions of variables in the KBK-HBG Data Set. Before splitting into separate sets for training and testing.
Appendix B

Ridge Regression

This appendix merely highlights some of the steps in deriving the Ridge Regression as described in [34]. Please refer to that reference for a more comprehensive walkthrough.

\[ y = \beta_0 + \sum_{i=1}^{d} \beta_i u_i + \epsilon \]

(B.1)

The standard multiple regression problem means finding the regression coefficients \( \beta_0, \ldots, \beta_d \) in (B.1) that make the error \( \epsilon \) as small as possible, for an output variable \( y \) and an input variable \( x = (u_0, u_1, \ldots, u_d) \).

\[ \text{LS}(\beta_0, \ldots, \beta_d) = \sum_{i=1}^{N} \left( y_i - \beta_0 - \sum_{j=1}^{d} \beta_j u_{ij} \right)^2 = \sum_{i=1}^{N} \epsilon_i^2 \]  

(B.2)

The regression coefficients are found by minimizing the Least-Squares function in (B.2) with respect to the regression coefficients, for each input-output pair in the data set \( \mathcal{D} \).

\[ \mathcal{D} = \{(u_{11}, \ldots, u_{1d}), y_1\}, \ldots,\{(u_{N1}, \ldots, u_{Nd}), y_N\} \]

(B.3)

The equation (B.2) can be written with matrix notation as

\[ \text{LS}(\beta) = (y - X\beta)^T \cdot (y - X\beta) \]  

(B.4)

After taking the derivative of (B.4) with respect to the least-squares estimator of \( \beta \), denoted \( \hat{\beta} \), and equating the derivative to zero, the estimator can be expressed with basic matrix operations on the data.
\[ \hat{\beta} = (X^T X)^{-1} \cdot X^T \cdot y \]  \hspace{1cm} \text{(B.5)}

Ridge Regression introduces bias to the estimator by transforming it with a parameter \( \alpha \), so that we arrive at a new estimator \( \hat{\beta}_{Ridge} \) that has less variance.

\[ \hat{\beta}_{Ridge} = \left( X^T X + \alpha I \right)^{-1} \cdot \left( X^T X \right) \cdot \hat{\beta} \]  \hspace{1cm} \text{(B.6)}