Resume

Titel: En Intelligent Model til Forudsigelse af Energiforbrug for Elbiler ved brug af Big-Data

Elbiler er en større og større del af de biler, der kører rundt på vejene. Et problem med elbiler er, at de indeholder en begrænset mængde strøm samt at det tager tid at lade dem op. I dette projekt arbejder vi med en problemstilling, der omhandler at komme med pålidelige estimater på energiforbrug for en fremtidig tur for elbiler.

Vi anvender et stort datasæt fra et eksisterende data warehouse, der indeholder historiske observationer, som er map matched til et kort over Danmark. Desuden indeholder datasættet informationer omkring vejr og oplysninger om de enkelte veje.

Som tilføjelse til data warehouset har vi konstrueret en generel metode til at kombinere et kort med højdedata. Vi har desuden analyseret i hvilket omfang højdeforskellen og andre faktorer påvirker energiforbruget. Udover højdeinformation har vi tilføjet og analyseret information omkring rundkørsler, zoner, trafiklys og retningsskift på en tur. Baseret på vores analyse har vi fundet frem til hvilke faktorer, der har en påvirkning, og anvendt disse i vores videre arbejde. Vi anvender ikke den sande tid og hastighed i vores model, fordi disse ikke kendes for fremtidige ture. I stedet anvender vi hastighedsgrænsen og et estimat af tiden på baggrund af disse.

Vi har udarbejdet tre forskellige modeller: Lineær regression (LR), Neural Network (NN) og et neural network, der er kombineret med historiske observationer (NN-observationer). Kvaliteten af disse modeller er evalueret i forhold til tre baselines, der bygger på at anvende det gennemsnitlige energiforbrug eller at anvende det energiforbrug, der opgives af bilproducenterne. Desuden har vi vurderet hvilke features, som har størst betydning ved brug af LR med en L1 regularization og NN, hvor vi har lavet tests med forskellige kombinationer af features. Vi har evalueret vores modeller på et testsæt, hvor de faktiske energiforbrug er kendt. De to modeller LR og NN er betydeligt bedre end baseline metoderne. Dog bliver vores estimater mere pålidelige ved at anvende de historiske observationer i modellen NN-observations.

Vores eksperimenter viser at både trafiklys, rundkørsler og det implementerede højdekort forbedrer kvaliteten af vores energiforbrugsestimater. Desuden viser det sig, at den sande hastighed og tid forbedrer vores estimater. Så ved at lave en hastighedsmodel, der er mere præcis end hastighedsgrænsen, kan vi formentligt øge kvaliteten af vores resultater.

Da vi har valgt at komme med estimater på segmenter og derefter regne det samlede energiforbrug for ture bagefter, kan vores arbejde udvides, så det kan bruges i forbindelse med at foreslå de mest energieffektive router.

An Intelligent Model for Prediction of Energy Consumption for Electric Vehicles using Big-Data

Master Thesis

SOFTWARE MASTER THESIS
GROUP MI101F17
AALBORG UNIVERSITY

SPRING 2017

AUTHORS
MIKKEL GIEDSING NIELSEN

JACOB ELEFSEN

AALBORG UNIVERSITY
STUDENT REPORT

Department of Computer Science
Aalborg University
Software
Selma Lagerlöfs Vej 300
9220 Aalborg Øst
`www.cs.aau.dk`

Title:

An Intelligent Model for Prediction of Energy Consumption for Electric Vehicle using Big-Data

Project:

Master Thesis

Project period:

01/02-2017 - 06-06/2017

Project group:

MI101F17

Participants:

Jacob Elefsen
Mikkel Giedsing Nielsen

Supervisors:

Thomas Dyhre Nielsen
Kristian Torp

Report pages: 75
Appendix pages: 10
Completed: 06/06-2017

Abstract:

This report considers the problem of predicting a trustworthy energy consumption for a future trip. To this end, we have access to a data warehouse that contains historical observations about trips that are map-matched to a road network. We construct a general method of combining a road network with elevation data. The elevation information and other factors which impact the energy consumption are analyzed. From the analysis, a feature set that describes energy consumption is constructed. This feature set does not include information which is not available for future trips, such as the exact speed and time. We construct three models, Linear Regression (LR), Neural Network (NN), and a Neural Network that is combined with historical observations (NN-observations). We evaluate the quality of the feature set using the LR and NN. In order to determine the performance of these three models, we compare them to three baselines. The LR and NN models outperform the baselines by a slight margin. However, we find that the including the historical observations in the NN-observation model increases the performance significantly.

Contents

Introduction 1

Climate change is a topic for a vast number of discussions; this has resulted in many advances in sustainable development. One of these developments are Electric Vehicles (EVs), which has the potential of zero emission transportation. Every year the number of EVs that are sold increases, in Fig. 1.1 we show the trend. From this graph, it is evident that EVs are gaining tracking in the car market. An Electric Vehicle (EV) has both benefits and drawbacks when compared to a Conventional Vehicle (CV). The drawbacks are a limited range, and the time it takes to recharge an EV. As a result, people experience "range anxiety", i.e. a fear of depleting the battery on a trip. [Franke et al., 2012] finds that a key factor in easing range anxiety is the trustworthiness of a range estimation system.

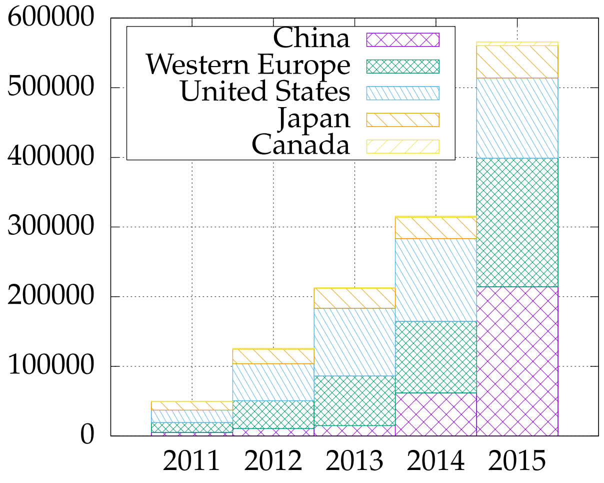


Figure 1.1: The number of EVs sold per year, for different regions of the world [U.S. Department of Energy, 2016].

The objective of this report is to present a method that can accurately predict an energy consumption for any route from A to B. To this end; we have access to a data warehouse which contains information from EVs. We use this dataset to train three different models, that predicts energy consumptions.

1.1 Problem Description

We want to predict the energy consumption for a route from A to B. One crude way to predict the energy consumption is using the length of the route combined with energy consumption which is specified by the car manufacturer.

A more elaborate method of predicting energy consumption is by using historical observations on a route. An average energy consumption for a route can be found using these observations. However, this approach requires observations for all roads. There are, however, many factors that affect energy consumption. In [Krogh et al., 2015] some of these factors are studied, they find weather conditions, and seasons affect the energy consumption.

Based on our understanding of the subject and the results from [Krogh et al., 2015] we have established the following hypotheses about the factors that have an impact on the energy consumption.

1. A higher speed will result in a higher energy consumption.
2. A higher acceleration will result in a higher positive energy consumption, and deceleration will result in a negative energy consumption.
3. A lower temperature will cause the driver to use the heating system causing a higher energy consumption.
4. Different drivers have a different pattern, causing different energy consumption.
5. The time of day has an impact on the energy consumption due to various congestion.
6. The slope of a road has an effect on energy consumption.

We will confirm or invalidate, each of these hypotheses to determine whether or not to use them in our energy consumption model.

1.2 Initiating Problem

The initiating problem for the project is the following:

Initiating Problem

> *Based on a real-world dataset collected from EVs containing GPS and CAN bus data, which provides information about speed, location, and energy consumption, is it possible to accurately predict an energy consumption for any arbitrary trip from point A to B given criteria such as weather information and time of day, using machine intelligence techniques*

In Chapter 2 related work in the research field is explored. Chapter 3 gives an overview of the available data. In Chapter 4 we show how additional information is included to the data warehouse Chapter 5 provides an in-depth data analysis, in which hypothesis about energy consumptions are confirmed and invalidated. Based on the data analysis we form a problem statement. We explain the machine learning methods we use in Chapter 6 and show the construction of features. Chapter 7 describes our experiments, the setup, and compare three baselines with our methods. In Chapter 8 we conclude our work.

Related Work 2

Predicting energy consumptions for EVs is a novel area. There has been some research within the area, a common factor for the majority of articles is the goal to alleviate the sense of range anxiety. In [Zheng et al., 2016] a hybrid machine learning model is explored. The hybrid model composes of two elements, Self-Organizing Maps and Regression Trees (SOM-RT). They construct a feature vector that describes an entire trip, including variables such as duration, length, and information about the distribution of accelerations, speeds, and other sensory data. The dataset they use in their works consists of only 421 trips. The 421 trips are split into 5000 sub-trips for training and 500 for testing. Splitting the trips into sub-trips could create uniform trips which in turns means their training and test sets and synthetic. They assume that sensory data is available for future trips. This assumption means that the feature vector is inherently flawed in a real world setting. These sensory data give an indication the driving style for a given trip. However, the information is not available for the prediction of new trips.

In [Yu et al., 2012] a method for identifying the driving pattern is developed. This method uses static information, the speed-limit and the elevation for segments. This is useful because it can be generalized for an entire road network. In this approach, they cluster a trip into both multiple speed zones and hilly zones. The speed zones they use are based on the speed limits, and the hilly zones are identified by the change in elevation throughout a trip. These clusters are synthesized into a pattern that described the speed and elevation changes for a given trip. Using the driving pattern the energy consumption is predicted using kinetic equations. The prediction is accurate; however, it is only evaluated on a single trip.

In [De Cauwer et al., 2017] they work with 2 sets of data. The first data set consists of 3 EVs that are used as taxis, which are driven in Brussels Capital Region. These taxis are constantly driven by a set of different drivers. The other set of data has been collected from 30 EVs during a period of 1 year, in Belgium. They link only a subset of this data to the road network, altitude information. The subset consists of 2 EV taxis and 3 EVs of the 30 from the other data set. In their work, they consider how to predict an energy consumption for segments. Using a neural network, they predict two outputs, which describes the speed profile of a segment at a given time. They use speed profile, and altitude information, and the temperature in a linear regression model. In the final step, they aggregate the energy consumption for each segment of a trip into a total energy consumption for the trip. In the paper, they compare the method to an average consumption for all trips. For the data set with taxis, their method performs best, while average energy consumption is best for the other set. Lastly, instead of estimating the speed profile, they give the observed values to their linear regression model. With the observed speed profile, the linear model outperforms the average energy consumption.

Preliminaries 3

In this chapter we describe the road network, and the terminology. Additionally, we provide a description of our data-warehouse including how the trips are stored. We also list and give concise information of the data source which we have included. How the additional data sources are included and used is detailed in Chapter 4.

3.1 Road Network

In this section we describe the road network notation, and terminology used in this report. In our road network we have a set of segments $\mathbb{S}$. Each $s \in \mathbb{S}$ is equivalent to an *edge* from graph notation. We say that a sequence of segments is a *route*, the definition route is given in Definition 3.1.1.

Definition 3.1.1. A route must always contain at least one segment, and s_i, s_{i+1} must be adjacent. A *route* is given as sequence of segments:

$$route_i = (s_1, s_2, \ldots, s_n)$$

In our data-warehouse we have a notion of *trips*, a trip follows a route. Information has been recorded for several trips and is stored in our data-warehouse. We introduce the notation of $t, (i)$ pairs, where t specifies a specific trip, and (i) denotes the i*th* segment of the trip.

3.2 Data-Warehouse

A large Extract Transform Load process foregoes our data-warehouse, in this process Global Positioning System (GPS) data is map matched to a map from Open Street Map (OSM). In addition to map matching, data fusion of GPS data and Controller Area Network bus (CAN bus) data is performed. In Fig. 3.1 we show an overview of the data fusion. The product of the data fusion of GPS and CAN bus will be referred to as *observations*. These observations are then map matched and formed into trips, as shown in Fig. 3.1. These trips are stored as $t(i)$ pairs which have a reference to a set of observations. From these observations an energy consumption is calculated which is specified in *kWh*.

In addition to the observations being map matched and formed into trips and the trips have also been enriched with information from other sources. In the paragraphs below, we explain which information has been added to observations, and trips in the data-warehouse.

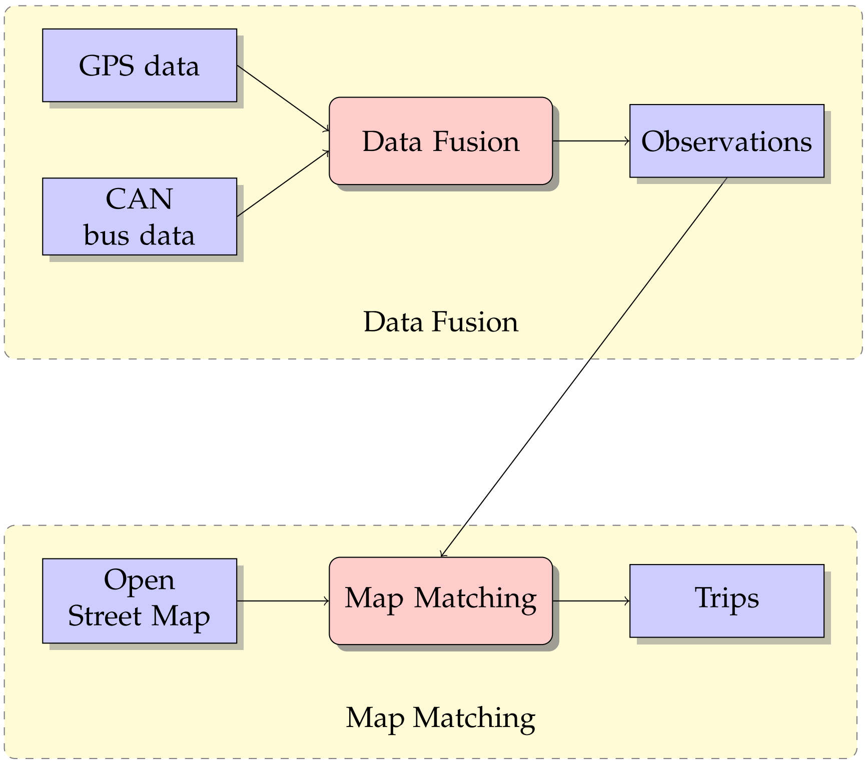


Figure 3.1: Important data components, and their relationship.

Weather Data The weather data is recorded hourly from 71 different weather stations across Denmark. Each $t(i)$ pair is linked to a weather information from the closest weather station. This data includes information about wind speed and direction, air temperature, weather classes (e.g. dry, wet, etc.), the data is collected from [Oceanic and Administration, 2017].

Open Street Map OSM is used for the road network in the data-warehouse, the map is from 2014-01-01. Each segment has information about the category (motorway, primary, etc.), speed-limit (this information is sparse), length, and direction (either forward or both). OSM contains more information which has not been included in the data-warehouse.

Regions The data-warehouse has information about the regions of Denmark, there are five regions, 99 municipalities, and 592 zip codes. While this information is in the data warehouse, it is not directly linked to each $t(i)$ pair.

Electric Vehicle Data The foundation of our dataset is the EV data which will be the basis for our predictions. All data from this dataset was collected by [Clever, 2017], and the data was collected in the period from 2012-01 to 2014-06. We omit data from the first three months as it has calibration issues as shown in [Andersen et al., 2014a]. This calibration issue means we consider data from the period of 2012-04 to 2014-06.

In the aforementioned time period 247 835 trips were recorded consisting of 199 399 109 observations. The observations were recorded at a $1Hz$ frequency. An observation contains several attributes such as location, direction, speed, time-stamp, State of Charge (SoC), watt usage since last observation, a list of attributes is found in Chapter 4.

The observations are from three types of cars; Citroën C-Zero, Mitsubishi iMiEV, and Peugeot iOn. The three cars are near identical; therefore, the data from all the cars are combined in the dataset [Andersen et al., 2014b]. According to [Peugeot, 2017] the cars have an average energy consumption of $106\frac{Wh}{km}$. This energy consumption is found based on a test based on the New European Driving Cycle (NEDC).

3.3 Additional Data Source

In addition to the data from the Data Warehouse described in Section 3.2, we have chosen to include data from the following data sources, because we assume it will improve our prediction model:

Supplementary OSM Data In addition to the 2014-01-01 version of OSM in the data-warehouse we include a newer version 2015-01-01. From this version, we include information about roundabout and traffic signals. We chose the newer 2015-01-01 version as it is readily available from previous work.

Zone Data [Styrelsen for Dataforsyning og Effektivisering, 2017] provides information about city zones, rural zones, and vocational zones for Denmark.

Elevation There exists different elevation models for Denmark, we have included a model into our data warehouse from [Styrelsen for Dataforsyning og Effektivisering, 2017].

The implementation of the three additional data sources is described in Chapter 4.

3.4 Data Distribution

In this section, we show how the data is distributed in the road network. We show all covered segments, i.e. segments with at least one observation, in Fig. 3.2 the black lines are motorways, the green lines are primary roads, and the rest are mixed. In this figure, we see that the majority of the western section of Denmark is missing.

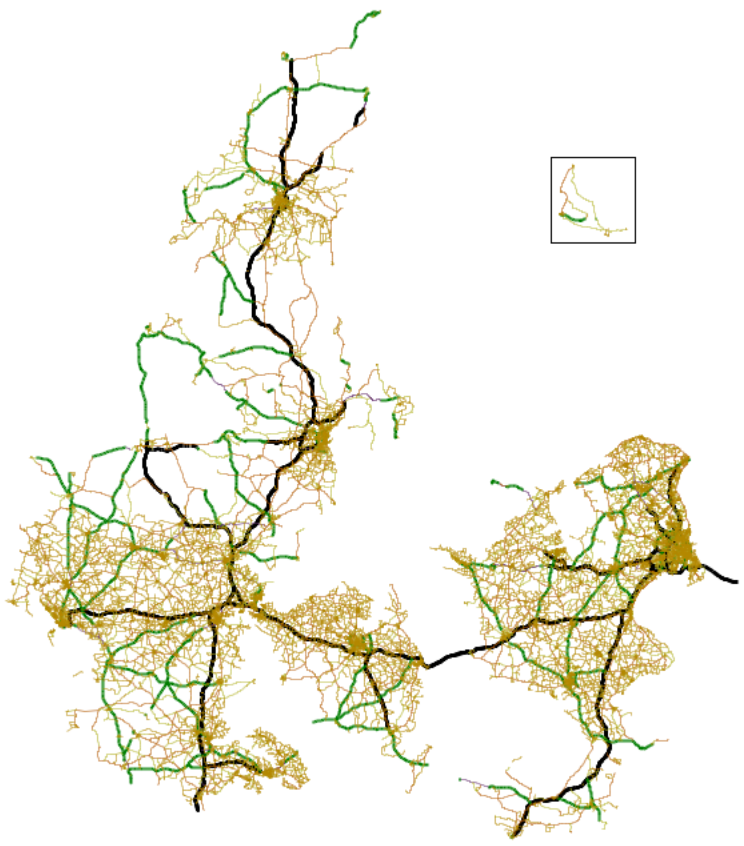


Figure 3.2: Spatial Data Coverage.

We show more concise information in Table 3.1. Where *Total segments* is the number of segments for each category, while *Segments w. Trips* is the number of segments which at least one trip has passed. Lastly, we show how many *t*(*i*)pairs there are for each category. It is noteworthy that the majority of *t*(*i*) pairs are recorded on *tertiary*, *secondary*, *residential* categories. This is due to these categories being shorter in general, i.e. one *t*(*i*)pair on a *motorway* could traverse 1km while a pair on a tertiary road would be less such as 100m. In Table A.1 the average length, the standard deviation of length for each category is shown. The average length of motorways is 1080m while *tertiary*, *secondary*, and, *residential* averages 155m, 212m, and, 104m respectively.

OSM Category	Total Segments	Segments w. Trips	$t(i)$
residential	288 081	65 652 (23%)	2 193 847 (16.7%)
service	140 834	12 055 (9%)	205 751 (1.6%)
unclassified	130 005	39 985 (31%)	1 381 253 (10.5%)
tertiary	56 240	39 976 (71%)	4 297 580 (32.6%)
track	45 400	578 (1%)	4742 (0.03%)
unpaved	38 734	2041 (5%)	15 622 (0.1%)
secondary	28 947	22 364 (77%)	3 104 621 (23.6%)
primary	13 039	9702 (74%)	1 309 449 (10%)
motorway	2213	2121 (96%)	387 162 (2.9%)
living_street	1751	316 (18%)	4668 (0.04%)
motorway_link	1727	1423 (83%)	87 179 (0.7%)
trunk	878	732 (83%)	148 879 (1.1%)
road	770	21 (3%)	78 (0.0%)
trunk_link	234	156 (66%)	12 055 (0.1%)
primary_link	228	144 (63%)	6786 (0.05%)
secondary_link	169	123 (73%)	4869 (0.05%)
tertiary_link	67	48 (72%)	1532 (0.01%)
ferry	54	0 (0%)	0 (0.0%)
Total	749 371	197 437 (26%)	13 166 073

Table 3.1: Distribution of Data on the different Road Network categories.

Database 4

In this chapter, we will show the information which is available and show our contributions to the data-warehouse. OSM is described such that we can discuss how new attributes can be derived, and how the elevation model can be included. For the inclusion of the elevation model, we consider four approaches, two of which we include. By using information from OSM we derive new attributes, namely direction change, traffic signal information, and roundabout information. Throughout this chapter, we show diagrams which show what we have contributed. The green diagrams are our contributions, while the yellow is the information present in the data warehouse.

4.1 Open Street Map

In this section we will describe the format of OSM, and which information from OSM is already included in our data-warehouse, and which information we include.

4.1.1 Vector Representation

All the segments in our data warehouse are stored in a data type called *geography*, which can represent different spatial objects such as, Points, Polygons, and Linestrings. These objects have a vector representation as they are connections between points. A Point is given as a coordinate; this is the smallest instance we use. Segments are represented as Linestrings, which consists of two or more points, 380 576 (50.8%) of segments consists of two points.

4.1.2 Data-Warehouse Open Street Map Information

In the data-warehouse a map from OSM has been included, in Fig. 4.1 we show the information that is available, the attributes are described below:

Category Describes the type of segment can be one of the following ferry, living_street, motorway, motorway_link, primary, primary_link, residential, road, secondary, secondary_link, service, tertiary, tertiary_link, track, trunk, trunk_link, unclassified, unpaved.
Direction Signifies which direction a segment can be traversed, can be either *Forward* or *Both*.
Segangle The angle of the segment, given by the start and end point of each segment.
Speedlimit Forward The speed limit going forward on a segment, 0 if the limit is unknown.
Speedlimit Backward The speed limit going backward on a segment, 0 if the limit is unknown.
Segmentgeo The spatial representation of a segment.
Streetname The street name of a segment, the street is not always known.

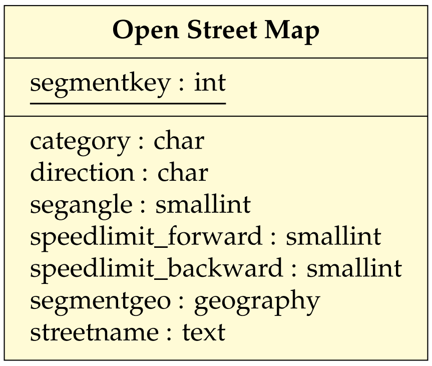


Figure 4.1: Data-Warehouse OSM information

4.1.3 Additional Open Street Map Information

As mentioned in Section 3.3 OSM contains more information than has been included in the data-warehouse. In Fig. 4.2 we show that we include information about roundabouts and traffic signals. This component has a geography, the geography represent either a roundabout or traffic signal. We use this component to derive new attributes in Section 4.4

4.2 Elevation Data

In this section, we will contribute to the data-warehouse by showing a general method for combining a given vector map (e.g. OSM), with a raster map. Moreover, we will discuss approaches for this implementation. We have access to different elevation models from [Styrelsen for Dataforsyning og Effektivisering, 2017] the different types of models are also described in the section.

4.2.1 Data-Set

The elevation information for Denmark is from 2015 it is available in different formats. There are two different models and Laser Scanning (Lidar) data. The two models are Digital Surface Model (DSM) and Digital Terrain Model (DTM), these models are derived from Lidar data. In Fig. 4.3 we show the difference between the two models. Using DSM we will get the elevation of house roof, while DTM has the elevation at ground level. Our goal is to find the elevation of any given segment. As such we are not interested in the elevation of obstructions such as houses and trees. Using this criterion we chose between the DTM and Lidar. The DTM has

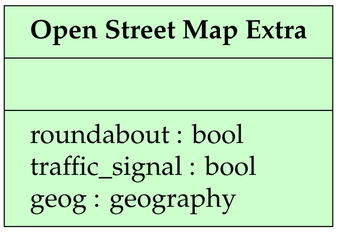


Figure 4.2: Data-Warehouse OSM information

been derived from the Lidar data by professionals, as such we choose to use the DTM instead of deriving it from the Lidar data ourselves.

4.2.2 Raster Format

The DTM comes in raster format, specifically GeoTIFF. In Fig. 4.4 we show a raster map of Denmark the squares are called pixels, the values of the pixels are arbitrary altitudes above sea-level. Pixels, in the figure, without a value are at sea-level. Given a coordinate in Denmark, we can find the altitude of that coordinate, by finding which pixel that contains the coordinate. Each pixel represents an area of 40x40cm, and are mutually exclusive, i.e. there is no overlap.

4.2.3 Slope Inclusion

The purpose of the including elevation data is to enhance the map from OSM by finding a slope for every segment. In this section, we describe four approaches for including the slope.

Approach 1 For all segments in OSM the start and end points will be enriched with an elevation. Using the elevation, and the distance between the start and end point a slope is calculated.

Approach 2 Subdivide all segments into sub-segments s.t. no sub-segment is longer than a given maximum length. We can then enrich start and end points of each sub-segment with elevation information to calculate a slope.

Approach 3 Using the size of the pixels in the raster, we can construct interpolated points, using these points we can subdivide segments. In this approach, we utilize all possible elevation information for each segment.

Approach 4 A combination of Approach 2 & 3 such that we have a constraint on sub-segments, where they must be at least some given length. After subdividing the segment, we can check the slope on the new segments and smooth away nodes if the difference of slope is within some threshold. An example is shown in Fig. 4.5.

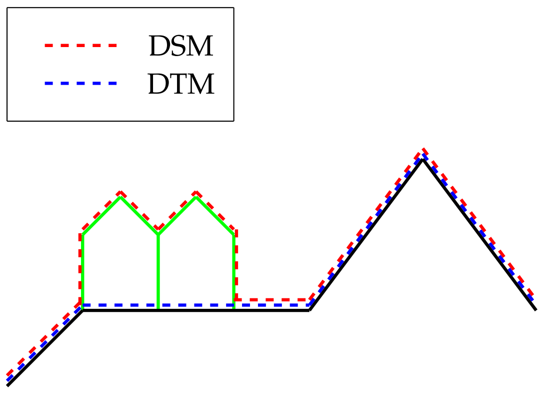


Figure 4.3: The dashed lines signifies the given elevation from a model. In this example the black line is the ground, while the green lines are houses.

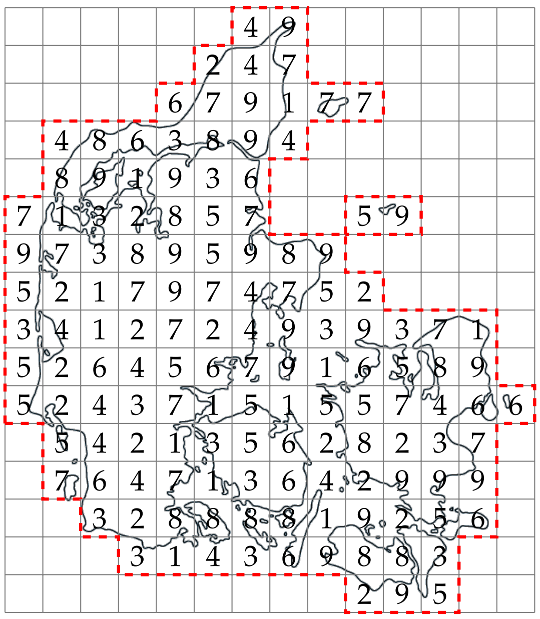
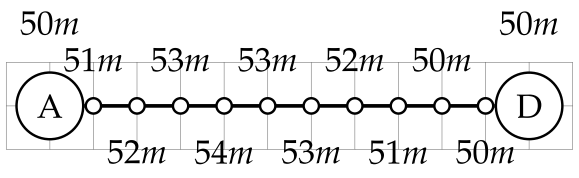
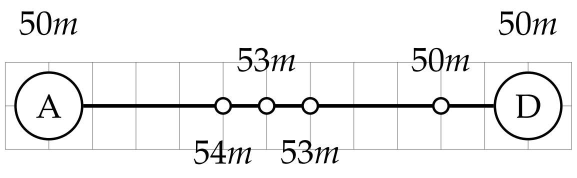


Figure 4.4: An abstract example of a raster map for Denmark



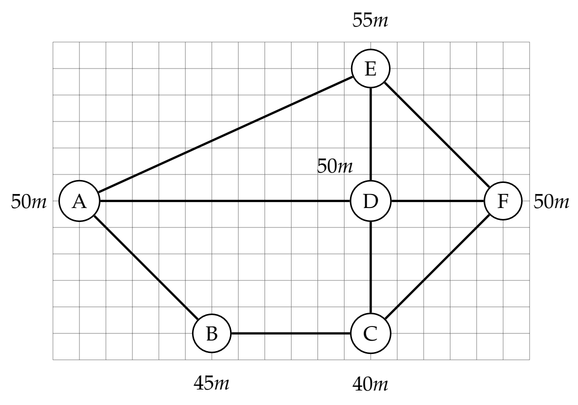
(a) Approach 4 - Subdivide



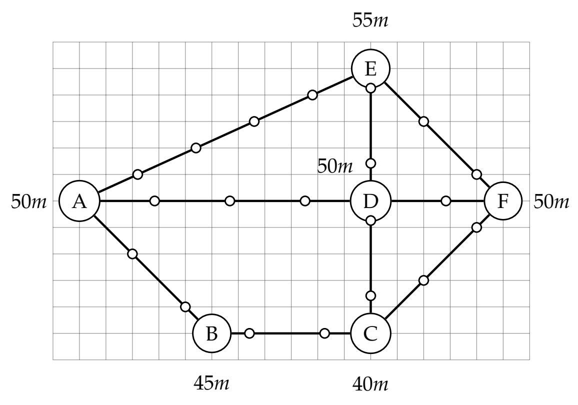
(b) Approach 4 - Smoothing

Figure 4.5: Figure 4.5a show the elevation of all nodes after subdivision. Figure 4.5b shows the smoothing of nodes, in this example we remove nodes when they do not impact the slope.

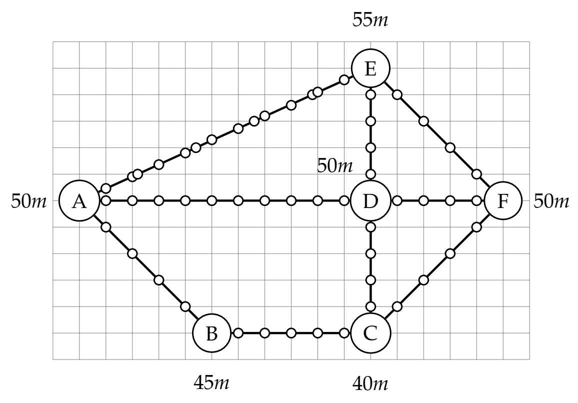
We have depicted these four approaches in Fig. 4.6. The smaller nodes signify subdivided segments, these nodes also include elevation information. There are compromises between the different approaches, computation time will naturally increase with the number of nodes, while an increase of nodes will provide a richer notation of inclination between nodes. In addition to computation time and accuracy, there is also an issue of matching observations to segments. We will implement and evaluate approach 1 and approach 2 to see if a larger granularity in the slope results in better prediction.



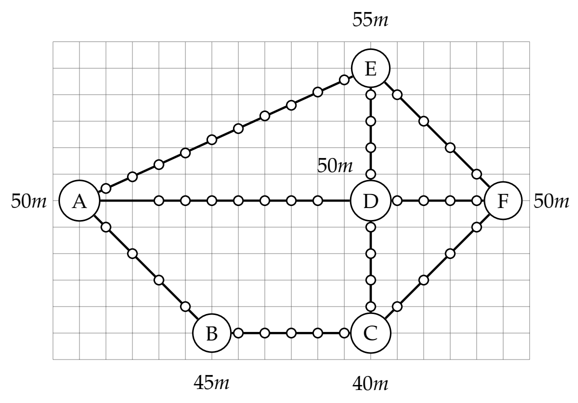
(a) Approach 1 - Basic



(b) Approach 2 - Fixed Length



(c) Approach 3 - Interpolation



(d) Approach 4 - Combination

Figure 4.6: Four different approaches for including elevation information in nodes.

4.2.4 Approach 1 Implementation

Approach 1 is the minimal solution for combining raster with OSM, as including less information cause the calculation of slope to be problematic.

With the raster data included the data-warehouse, we construct an SQL query which allows us to find the elevation at the start point and end point of each segment. In Listing 4.1, a snippet of our SQL query is shown, in this snippet we show how the elevation of start points is found. There are some important aspects of this query. Specifically, it makes use of three functions. The sub-query in Lines 3-4 uses `ST_StartPoint` which provides the start point, or coordinate of a given segment. In other words, the subquery contains the start point for all segments. With this information, the next step is to find which pixel each start point belongs too, and this is achieved with `ST_Intersects` in Line 5. Lastly, `ST_Value` is used to find the elevation value in the raster, `FALSE` signifies that pixels with no data are included. There are ten start points and end points which have no elevation information. These points belong to segments categorized as *ferry*, i.e. it does not present as a problem because we omit ferry segments as there are no observations on these segments as shown in Table 3.1. The same method is used to find the elevation of end points, using `ST_EndPoint` as opposed to `ST_StartPoint`. The full SQL query is shown in Listing B.1.

Listing 4.1: Approach 1 SQL Snippet

```sql
SELECT segment.segmentkey, ST_Value(rast, segment.startpoint, FALSE)
FROM experiments.elevation ele,
    (SELECT segmentkey, ST_Startpoint(segmentgeo) as startpoint
    FROM maps.osm_dk_20140101 LIMIT 800000) segment
WHERE ST_Intersects(ele.rast, segment.startpoint);
```

Having found the start and end height of each segment we are interested in finding the slope of each segment, the length of each segment is given in our data-warehouse. However segments can consist of multiple points, i.e. a segment can be curved. 49.2% (368 795) of segments are contains multiple points. Because of this large number we chose to find the distance between the start point and end point of each segment, s.t. we can calculate the slope from this distance. We show the equation for slope in Eq. (4.1).

$$slope = \frac{end\ height - start\ height}{distance} \tag{4.1}$$

4.2.5 Approach 2 Implementation

We will begin by subdividing all the segments in our road network. Knowing that GPS locations have some error we chose a sub-segment length of $50m$, such that we may be able to map match observations to a sub-segment. If a segment is $130m$ it will divided into 3 segments of length $50m$, $50m$, and $30m$. We construct the sub-segments by implementing a function in the data-warehouse. The function is called `split_linestring`, it takes two parameters, the *length* which segments will be split into, and the *segment* which is going be subdivided. The function is shown in Listing B.3.

We run this function on all our segments, s.t. all segments are subdivided. Once the sub-segments are created, finding the slope is similar to Approach 1, the different being we find the elevation of start and end points of the sub-segments and then use Eq. (4.1) to find the slope.

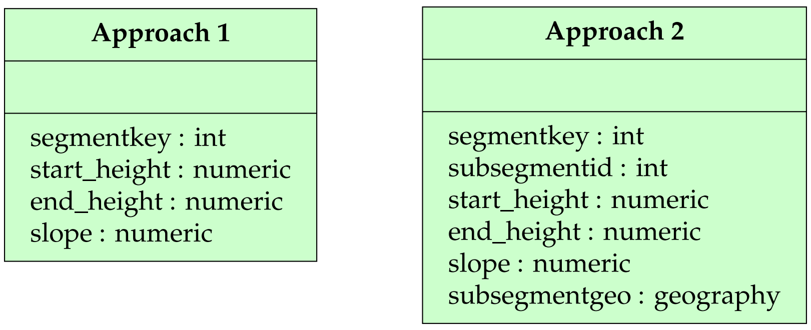


Figure 4.7: Approach 1 & Approach 2 contribution

4.3 Zone Data

Using information from [Erhvervsstyrelsen, 2017] we can include information about City, Rural, and Cottage Zones. We include this information into our data warehouse, such that we can use any difference of energy consumption in the zones for our model. Our assumption is that you may drive faster in rural zones compared to city zones.

The data consist of one shapefile which includes 4258 polygons where each polygon represents a zone. In addition to the zone type, the data include municipality of the each zone. We show a map of these zones in Fig. 4.8 the zones are reported by the municipalities which are why there are such differences, especially in the Rural Zones.

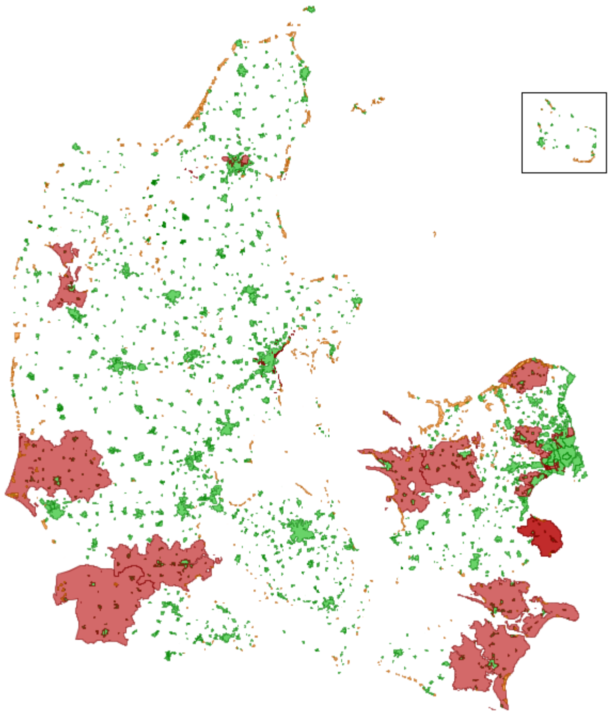


Figure 4.8: Green: City Zones, Red: Rural Zones, Orange: Cottage Zones

Since the zones do not fully cover Denmark, we consider how we handle segments that are not in a zone. The segments that are not in a zone will we annotate as an Unknown zone.

Each of the 749 371 segments from OSM are mapped to the zones. 8798 segments are part of multiple zones, for these cases we chose the zone where the segment overlaps the most. An example of a segment in multiple zones is seen in Fig. 4.9. The black line is the segment, and the red area is Rural Zone, and the green area is City Zone, in this case, the segment is mostly in the Rural Zone, and is annotated as such.

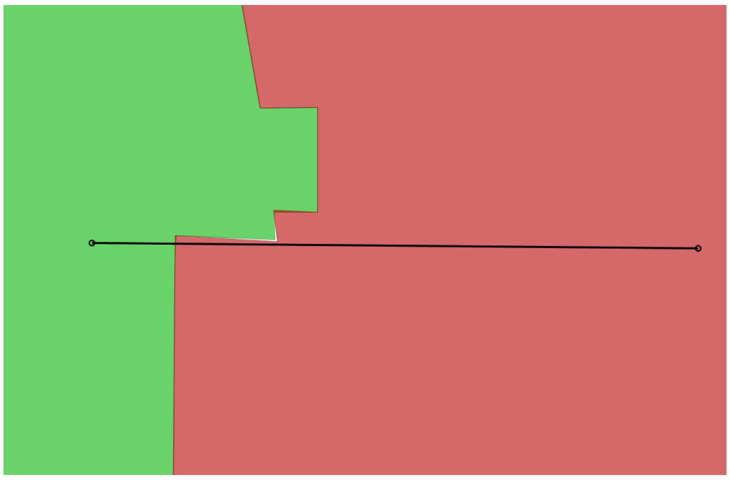


Figure 4.9: Segment in Multiple Zones

Zone
segmentkey
zone : text

Figure 4.10: Zone Contribution

4.4 Derived Attributes

We derive 4 new attributes from the OSM information and the Zone Data. These attributes are *Direction Change*, *Traffic Signal*, *Roundabout*, and *Speed-limit*

4.4.1 Direction Change

The purpose of this attribute is to find how much a driver turns when he goes onto a new segment. Segments are annotated with a *segangle*, however this attribute has a shortcoming as it is the angle between the start and end point. Instead we find the angle as shown in Fig. 4.11, here we have two segments *Segment 1* and *Segment 2*. In our example, a driver is moving from Segment 1 to Segment 2, since segments are directed the driver is traversing Segment 1 backward thus the angle we need is between point 2 and 1 on Segment 1 which is 334°. The angle we need on Segment 2 is between 1 and 2 which is 222°. This gives us a direction change of $332° - 222° = 112°$. The idea behind this feature is to differentiate between right, and left turns, as well as going straight across an intersection. We store this attribute as a natural number, opposed to labeling, with the right, left, and straight s.t. we can capture different possibilities. Intersections are not always limited to only one right, one straight, and one left i.e. there can be any number of possibilities.

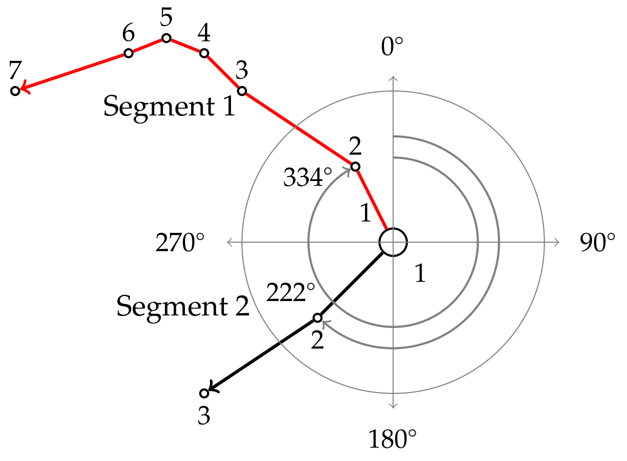


Figure 4.11: Direction of two segments sharing a connection

The order of which the segments are traversed is important, as we need to find the correct angle as shown in Fig. 4.11. There are four different cases which are considered.

1. Forward to Forward.
2. Forward to Backward.
3. Backward to Forward.
4. Backward to Backward.

Our example in Fig. 4.11, where a driver moves from Segment 1 to Segment 2 is case 3 Backward to Forward. With this knowledge, we can construct a function which calculates the direction change We make use of the following functions, in our data warehouse:

- `ST_Azimuth(geometry p1, geometry p2): returns the north based azimuth in radians`
- `ST_PointN(geometry linestring1, integer n): returns the nth point of a geometry`
- `ST_NPoints(geometry g1): returns number of points in the geometry`

Using these function, and given two segments and the direction they have been traversed, i.e. *forward* or *backward* we can find the direction change between the segments as shown in Algorithm 1.

In Listing B.5 we show the our SQL Statement that constructs the feature. An end result is a number between 0-359, where 0 correlates to forward, 90 to the right, 180 to backward, 270 to the left.

For this attribute we must also consider that it is sparse, i.e. at the end of a trip we do not travel to a new segment thus there cannot be a change of direction, this case we say that the direction change is 0.

4.4.2 Traffic Signal

From OSM we have information about the location of traffic signals. This information is not included in the data warehouse, but we include it. The task here is to find the locations of traffic signals and finding which segments they affect. Since the traffic signals are represented as points, i.e. they are not directly annotated on segments, we must find which traffic signals

Algorithm 1 $(s1, s2)$

```
input:
    $s1$: From Segment, $d1$: From Direction
    $s2$: To Segment, $d2$: To Direction
output: Δdirection
 1: if $d1$ is $forward$ then
 2:     $points \leftarrow ST\_NPoints(s1)$
 3:     $from\_dir \leftarrow ST\_Azimuth(ST\_PointN(s1, points - 1), ST\_PointN((ST\_PointN(s1, points))$
 4: else if $d1$ is $backward$ then
 5:     $from\_dir \leftarrow ST\_Azimuth(ST\_PointN(s1, 2), ST\_PointN((ST\_PointN(s1, 1))$
 6: if $d2$ is $forward$ then
 7:     $to\_dir \leftarrow ST\_Azimuth(ST\_PointN(s2, 1), ST\_PointN((ST\_PointN(s2, 2))$
 8: else if $d2$ is $backward$ then
 9:     $points \leftarrow ST\_NPoints(s2)$
10:     $to\_dir \leftarrow ST\_Azimuth(ST\_PointN(s2, points), ST\_PointN((ST\_PointN(s2, points - 1))$
    return $to\_dir - from\_dir$
```

belong to which segments. Furthermore, we must consider how we annotate this, instead of simply annotating that a segment has a traffic signal, we can annotate at which end of the segment the traffic signal is, or possible both ends.

To annotate the segments, we must first find which segments are in connection with a traffic signal. To do this, we downloaded a 2015-01-01 map from OSM, from this we extracted the locations of traffic signals. With this information, we find which segments within $1m$ of a traffic signal. We chose a distance of $1m$ as the threshold because the traffic signals are not placed exactly on top of a segment, but close.

With this list of segments that meets our distance criteria, the next step is to find whether the traffic signal is near the start, end, or both of for each segment. We find that 150112 trips (60.57%) traverse one or more segments that is affect by a traffic signal.

4.4.3 Roundabout

There are many roundabouts in Denmark, 110907 trips (44.75%) traverse one or more roundabouts, we expect that roundabouts can have a large impact on the energy consumption. Because many trips pass a roundabout, we find it important to find how large an impact they have on energy consumption. Our approach to including this feature is by denoting segments with two attributes, one that denotes a roundabout at the start of the segment, i.e. when a trip leaves a roundabout, and one shows at the end of a segment. This approach will be represented by a Boolean value, as the distance as a continuous value is not guaranteed to be the distance to a roundabout that directly affects traffic on the given segment.

Roundabouts are annotated in OSM; however, this information has been omitted in our data warehouse so we must include it ourselves. We say that a segment is connected to a roundabout if the start or end point of a given segment is within 1 meters of the segment.

4.4.4 Speed-limit

The speed-limit is sparsely annotated in OSM. There are 749371 segments in OSM and 693500 segment does not include the speed-limit. However OSM provides rules for imputing speed-limit in Denmark [Map, 2017]. For all segments, without speed-limit, we use the SpeedLimit function in Algorithm 2. This approach is naive because there are other speed limits in Denmark than those used in the algorithm. However the majority of segments follow these common speed-limits.

Algorithm 2 Speed-limit function

```
function SpeedLimit($zone, category$)
    if $category = motorway$ then return 130
    else if $zone = city$ then return 50
    else return 80
```

In Fig. 4.12 we show the how the derived attributes in stored in the data-warehouse. Here id : bigint is a reference to a specific $t(i)$ pair.

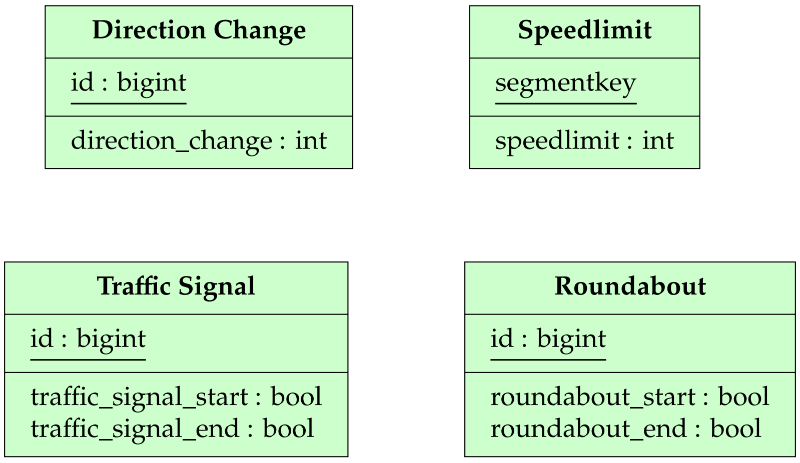


Figure 4.12: Derived Attributes Contribution

4.5 Overview

In this section, we provide an overview of the data warehouse, and the contributions we have made. We mark our contributions with green, while the existing information is marked with a yellow color. The overview is seen in Fig. 4.13.

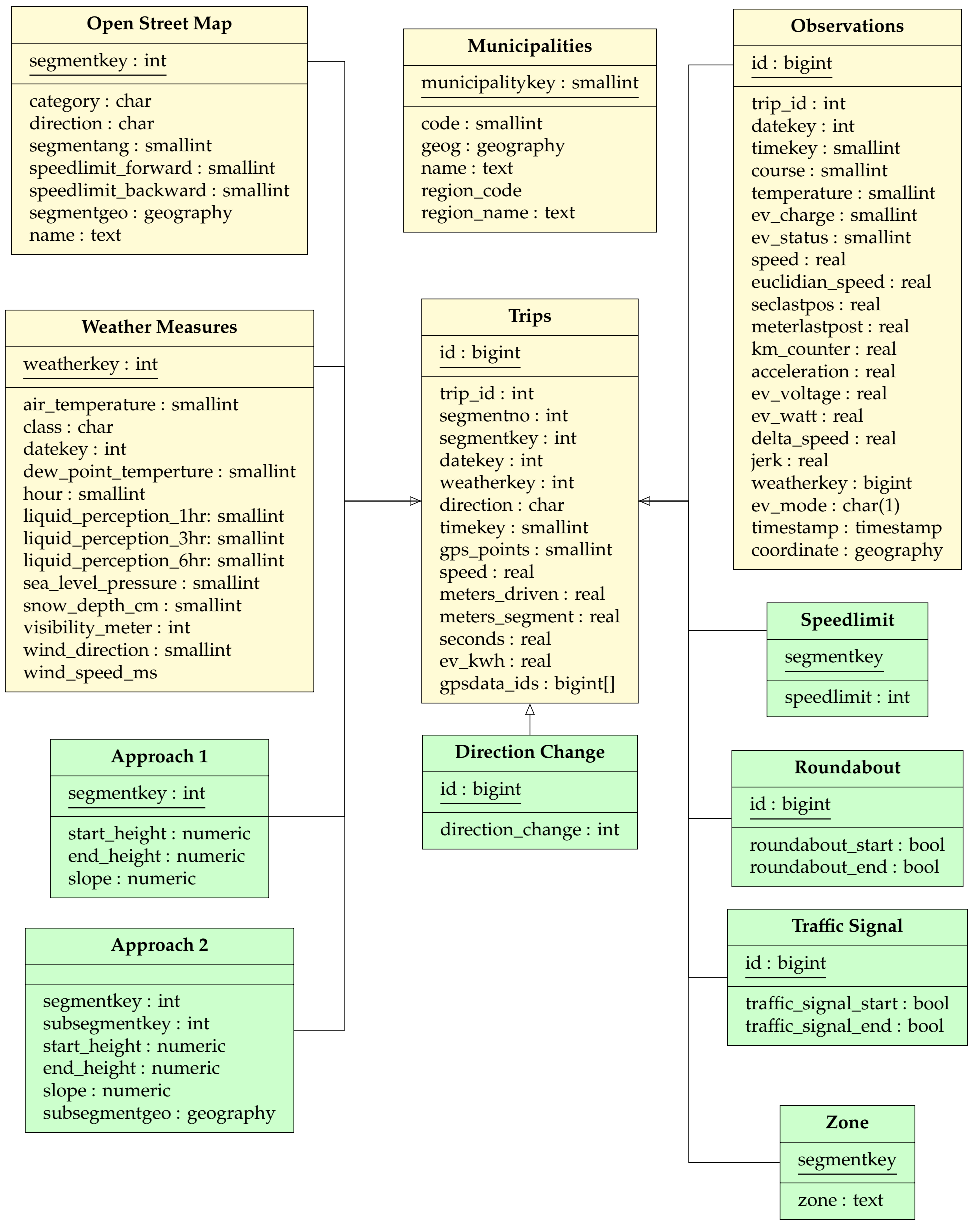


Figure 4.13: An overview of the attributes in the data-warehouse, and our contributions. Our contributions are green.

Data Analysis 5

In this chapter we are going to test our hypotheses from Section 1.1, this will give us an insight of the factors which impact the energy consumption. We begin our data analysis by finding the mean energy consumption for our data, which is $\mu = 165.3\frac{Wh}{km}$ and the standard deviation is $\sigma = 327\frac{Wh}{km}$. The μ is roughly 1.5 times higher than the energy consumption specified by [Peugeot, 2017], which is $106\frac{Wh}{km}$.

5.1 Speed

Hypothesis 1 A higher speed will result in a higher energy consumption.

We know from physics that kinetic energy is given as $E = \frac{1}{2}mv^2$, where the velocity v has an exponent. Therefore we expect to see an exponential increase in energy consumption as the speed increases. In Fig. 5.1 we show the relationship between speed and energy consumption. Contrary to our hypothesis, the energy consumption does not increase exponentially. The graph shows that the average energy consumption from 20 $\frac{km}{hr}$ and above near stable. The standard deviation is first stable from around 80 $\frac{km}{hr}$. When the speed is below 60 $\frac{km}{hr}$, the standard deviation is high. This high standard deviation in energy consumption is caused by drivers accelerating until they reach the speed limit.

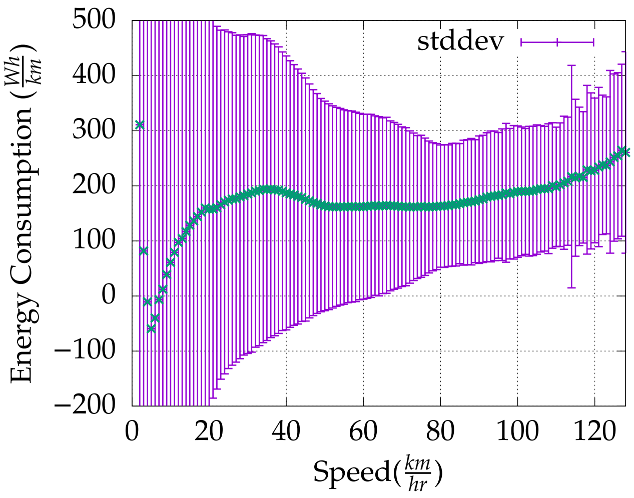


Figure 5.1

5.2 Acceleration

Hypothesis 2 A higher acceleration will result in a higher positive energy consumption, and deceleration will result in a negative energy consumption.

We know from physics that a higher acceleration requires more energy, Newton's second law dictates that $F = ma$, where F is Newton and m is the mass, and a is the acceleration. Energy in Joule is then given by $J = N{\cdot}meters$. From this, we expect that the relationship between acceleration and energy is linear.

In Fig. 5.2 we show how the energy relates with acceleration. Each point in the plot is an observation. It is clear that deceleration provides energy, while acceleration consumes energy. Deceleration below -1 appears to be a near constant $5000\frac{W}{s}$. We see a multimodal distribution of observations, with peaks around -0.8, 0, and 0.8 acceleration, which correlates to decelerating, stopping, and acceleration.

We argue that the relationship is between energy and acceleration is linear for some range of acceleration. This relationship could be captured by having a different linear function for various ranges of acceleration. In Eq. (5.1) we show function $f(a)$ which takes an input acceleration a, a will fall into one of the three ranges. Depending on the range an appropriate linear function is used. We determine that our hypothesis for acceleration hold.

$$f(a) = \begin{cases} f_1(a) & \text{if } -4 < a < 0 \\ f_2(a) & \text{if } 0 < a < 1 \\ f_3(a) & \text{if } 1 < a < 4 \end{cases} = Energy \tag{5.1}$$

5.3 Temperature

Hypothesis 3 A lower temperature will cause the driver to use the heating system causing a higher energy consumption.

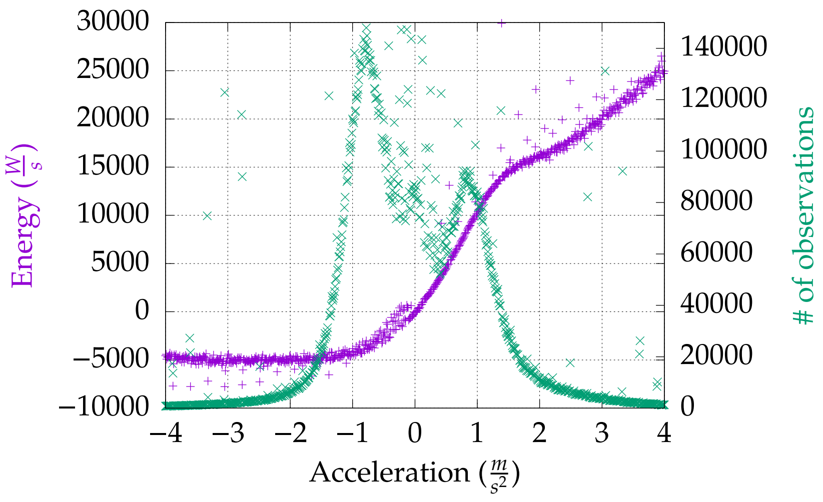


Figure 5.2: This graph shows an average energy consumption for a given acceleration.

Our hypothesis is that a EV uses more energy when the temperature is low because of heating system in the car. We test this hypothesis by showing the relationship between weather data and observations. The result is seen in Fig. 5.3. The figure shows the average energy use for each temperature.

The graph shows that the temperature has a high impact on the energy consumption. The consumption is almost twice as high at -10 $C°$ compared to 20 $C°$. We determine that our hypothesis hold, despite being unable to determine the direct cause of the energy consumption increase.

5.4 Speed & Temperature

We have found that speed and temperature has a significant impact on the energy consumption. In this section, we consider the energy consumption relationship between both speed and temperature. We begin by showing the relationship between speed and energy consumption at the four different seasons of the year, in Fig. 5.4. The graph shows the average energy consumption for given speed. Energy consumptions below $20\frac{km}{hr}$ and above $100\frac{km}{hr}$ are fluctuating and are as a result not shown. In the graph, we see that the different between the seasons are significant. Fall and Spring are similar while Winter and Summer are different by a factor of near 2.

In Fig. 5.5, we the average energy consumption for a given speed and temperature. It appears that speed and temperature has a near independent effect on energy consumption.

5.5 Driving Style

Hypothesis 4 Different drivers have different pattern, causing different energy consumption.

It is known that people drive differently, some drive aggressively accelerating fast, while others

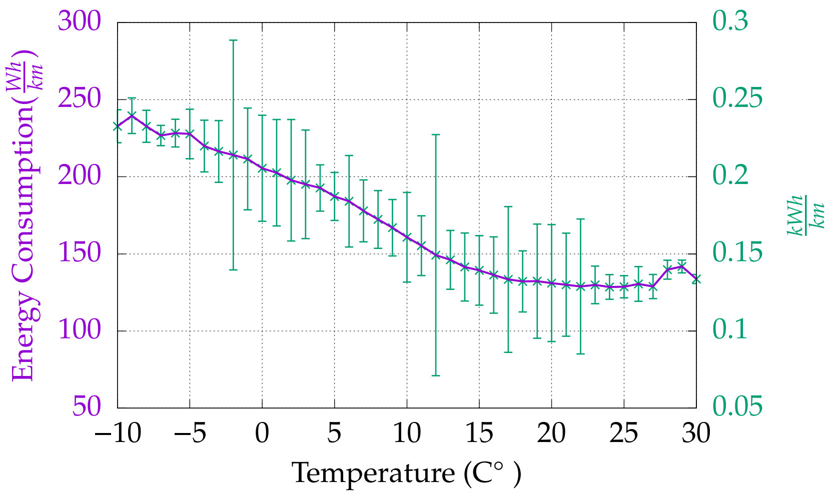


Figure 5.3: Average Energy consumption for each temperature

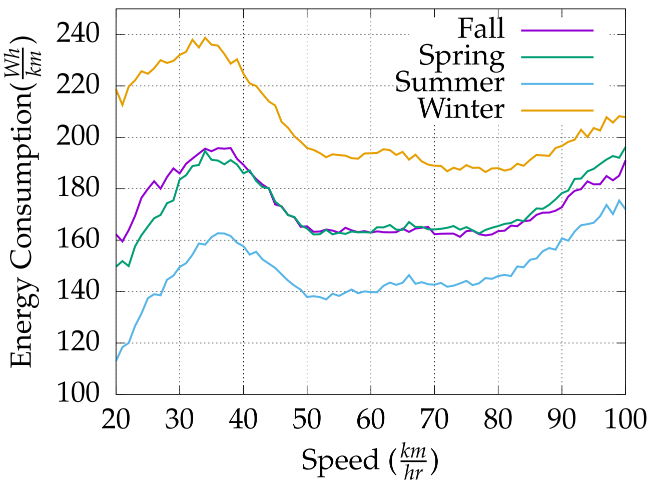


Figure 5.4: Average Energy consumption for each season over speed

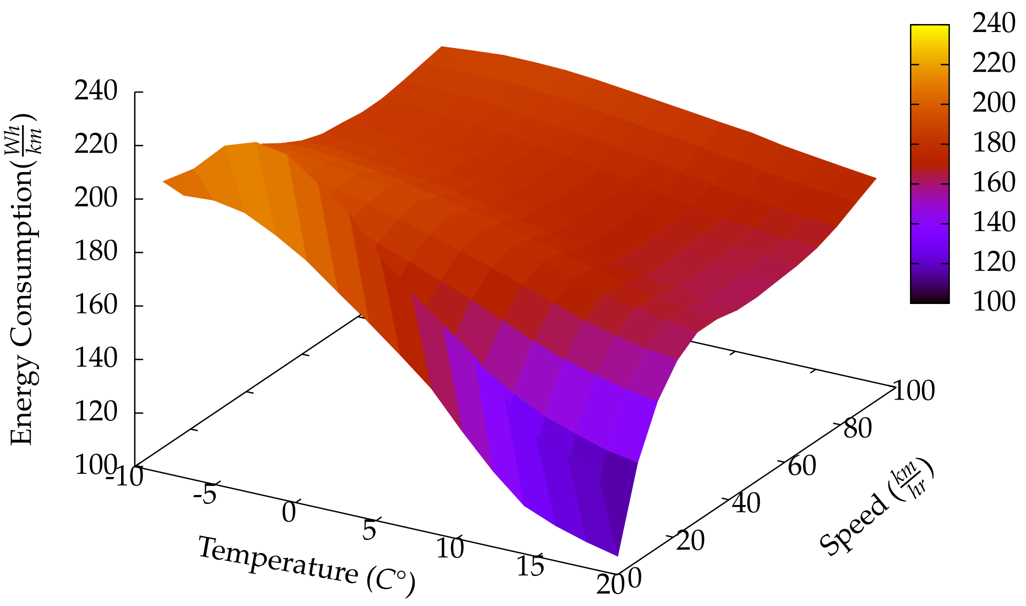


Figure 5.5: Average Energy consumption for each temperature and speed

are opposite and some are in between. We want to show the impact of these factors on the energy consumption. Our approach to answering this question is by looking at routes with a large set of trips. Given these trips and the observations for them, we will construct a combination of different attributes can be used to cluster trips. The target of this analysis is to find if reasonable clusters can be created from our data. If clusters can be created we want to find if how much they impact energy consumption.

For this analysis, we select two routes with a large number of trips from our data-set. Name, coordinates, length, a total number of trips and category is shown in Table 5.1.

Our analysis is based on techniques from [Constantinescu et al., 2010], in which the goal is to group drivers into groups based on how aggressively they drive. Our assumption is that trips which belong a cluster which is aggressive should have a higher energy consumption compared to trips that are in a passively driven cluster.

In the paper they use the following parameters, which they derive from their data-set:

- Speed over speed-limit - The percentage (V_{limit}) of time spent above the speed limit where the vehicle is moving.
- Speed - The average speed (V_{avg}) and standard deviation (V_{sd}) for observations where the vehicle is moving.
- Acceleration - The standard deviation (A_{sd}) for observations where the vehicle is moving.
- Positive acceleration - Average acceleration (A_{+avg}) and standard deviation (A_{+sd}) for all positive accelerations where the vehicle is moving.
- Breaking - Average deceleration (A_{-avg}) and standard deviation (A_{-sd}) for all observations where the vehicle decreases in speed. The free deceleration, decrease in speed without breaking is ignored based on a threshold for negative acceleration. Negative acceleration in this threshold is considered as free deceleration.
- Mechanical work - The sum W for of all positive kinetic energy values required to increase the vehicle speed.

We have calculated each of the driving parameters as described above. However in our test-cases, for simplicity we do not use a threshold for free deceleration, assuming all negative accelerations are stepping on brake.

In [Constantinescu et al., 2010] they conclude there are two *extreme* clusters based on the parameters. They use a Hierarchical Cluster Analysis (HCA) with Ward's method and Euclidean distance. We attempt to cluster the trips from our two routes, in the same approach. All parameters are normalized to values between zero and one. The result is shown in Table 5.3 and Table 5.2. In the first cluster from "Odense/Otterup" trips tend to follow the speed-limit, and drive slower, while trips in the second cluster indicate harder acceleration and heavy

Name	From (Lat,Long)	To (Lat,Long)	Length	No. trips	Category
Esbjerg/Varde	(55.527234, 8.458269)	(55.600499, 8.502106)	8.6 km	433	Primary
Odense/Otterup	(55.423181, 10.370683)	(55.504989, 10.394733)	9.6 km	428	Secondary

Table 5.1: Selected trips for clustering

braking. This results in less energy consumption in the first cluster. In the first cluster from "Esbjerg/Varde" the speed-limit is mostly respected, and the driven speed is slower, however there is a higher variation in speed compared to the second cluster.

Overall it appears the most aggressive drivers are in cluster 2 (Odense/Otterup) which also result in higher energy consumption. Cluster 1 (Esbjerg/Varde) has a higher energy consumption, despite having the appearance of being the least aggressive cluster based on softer braking and accelerations. This points towards the relation aggressive and passive clusters may not impact our energy consumption based on our driving parameters. We assume that this lack of impact may also be caused by other factors such as weather, traffic lights, etc.

We normalized the energy consumption in relation to the season. As seen in Section 5.3 there is almost a constant gap between the seasons. The average energy consumption is therefore calculated for each season. The average energy consumption for each season is used to calculate a factor to convert the energy consumption to the winter season. These factors are:

- Fall: 1.28
- Spring: 1.31
- Summer: 1.51

These factors are used to calculate a normalized energy consumption. For the (Esbjerg/Varde) route this not make a difference, but for the (Odense/Otterup) route there is a swap in the largest energy consumption. The most aggressive drivers are in cluster 2 (Odense/Otterup) and cluster 2 (Esbjerg/Varde), opposed to our hypothesis the energy consumption are lower for these. We conclude that the variation of speed is the best indicator for clustering. However the effect is near insignificant. Thus our conclusion is clustering on these parameters is not reasonable.

Parameter	Both	Cluster 1	Cluster 2
Size	428	336	92
Energy	1.55 kWh (0.41)	1.55 kWh (0.39)	1.56 kWh (0.46)
Energy normalized	1.97 kWh (0.50)	1.98 kWh (0.50)	1.94 kWh (0.50)
V_{limit}	32.05 % (20.07)	24.53 % (14.16)	59.51 % (13.47)
V_{avg}	79.20 km/h (7.32)	79.03 km/h (7.47)	79.79 km/h (6.77)
V_{sd}	17.76 (5.24)	17.79 (5.44)	17.67 (4.48)
A_{sd}	0.74 (0.54)	0.75 (0.59)	0.68 (0.29)
A_{+avg}	0.68 m/s^2 (0.84)	0.69 m/s^2 (0.95)	0.64 m/s^2 (0.14)
A_{+sd}	0.55 (0.29)	0.55 (0.32)	0.54 (0.16)
A_{-avg}	-0.73 m/s^2 (0.24)	-0.73 m/s^2 (0.26)	-0.70 m/s^2 (0.16)
A_{-sd}	0.79 (0.78)	0.82 (0.84)	0.70 (0.54)
W	17938.60 J (8124.08)	18093.94 J (8594.23)	17371.28 J (6113.59)

Table 5.2: Odense/Otterup clustering of trips

Parameter	Both	Cluster 1	Cluster 2
Size	433	338	95
Energy	1.22 kWh (0.25)	1.23 kWh (0.24)	1.17 kWh 0.26)
Energy normalized	1.56 kWh (0.35)	1.58 kWh (0.36)	1.49 kWh (0.33)
V_{limit}	28.34 % (19.37)	28.16 % (19.55)	29.01 % (18.78)
V_{avg}	76.87 km/h (9.56)	76.56 km/h (10.34)	77.96 km/h (5.91)
V_{sd}	14.98 (5.16)	15.19 (5.52)	14.26 (3.52)
A_{sd}	0.78 (0.51)	0.55 (0.20)	1.60 (0.10)
A_{+avg}	0.59 m/s^2 (0.12)	0.58 m/s^2 (0.12)	0.62 m/s^2 (0.10)
A_{+sd}	0.45 (0.15)	0.45 (0.15)	0.48 (0.15)
A_{-avg}	-0.70 m/s^2 (0.25)	-0.60 m/s^2 (0.12)	-1.07 m/s^2 (0.27)
A_{-sd}	1.01 (1.05)	0.51 (0.30)	2.78 (0.76)
W	15536.28 J (5808.78)	14551.81 J (5387.96)	19038.92 J (5925.28)

Table 5.3: Esbjerg/Varde clustering of trips

5.6 Time

Hypothesis 5 The time of day, has an impact on the energy consumption due to various congestion.

It is known that during peak hours there are more cars on the road network. Moreover, when there are many cars on the roads, this affect the speed and acceleration. We will, therefore, investigate how the energy consumption change during the day based on this assumption about change in congestion. The result is seen in Fig. 5.6. The graph shows that the energy consumption varies between 40 ($\frac{Wh}{km}$) during the day. As seen in Section 5.3 the temperature has a significant impact on the energy consumption, and therefore the temperature is also included in the graph.

5.7 Slope

Hypothesis 6 The slope of a road has an affect on energy consumption.

We believe that there is a correlation between the increase in energy consumption and the increase in slope. In order to test this hypothesis we use our implementation of Approach 1. In Fig. 5.7 we see a linear correlation. We say that the hypothesis passes, despite the standard deviation being large.

5.8 Derived Attributes Analysis

In Chapter 4 we derived new attributes that contain information about, direction change, traffic signals, and roundabouts. In this section we analyze them, such that we can determine their impact.

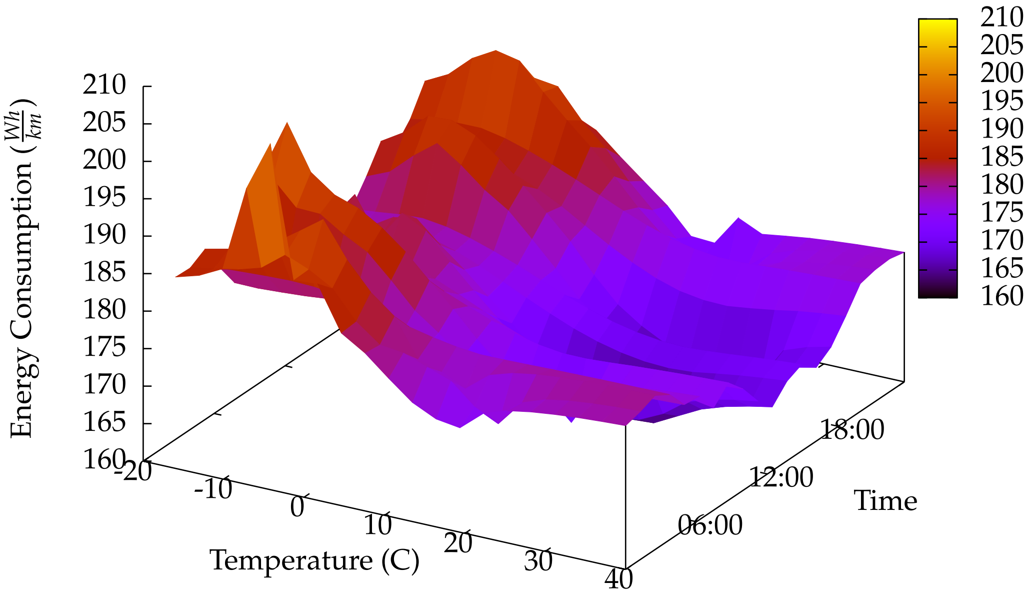


Figure 5.6

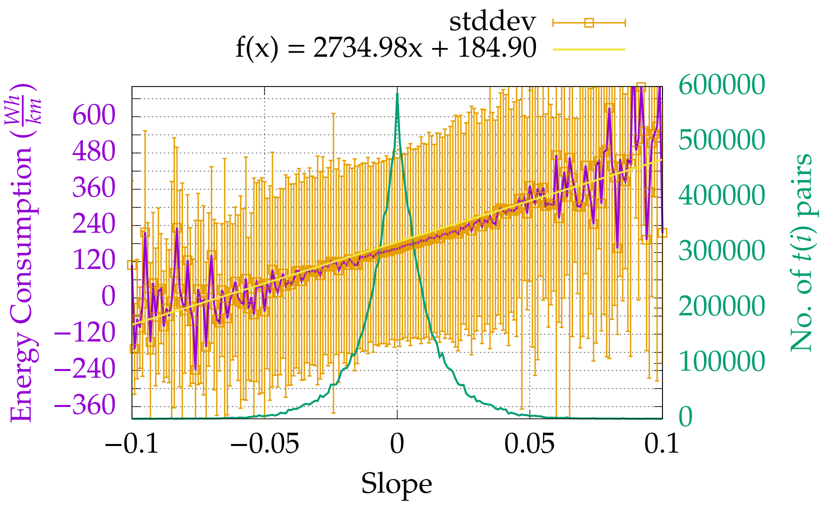


Figure 5.7: Average Energy Consumption for slopes

5.8.1 Direction Change

Using this feature, we can find how much impact direction changes has. In Table 5.4 we show how different angles impact the average energy consumption, which we have labeled straight, right, left, and back. The table shows the average cost. Compared to the average energy consumption for all directions, the direction seems to have a large impact on the average energy consumption. Especially we see that right turns are cheap compared to the average energy consumption. Based on this we assume it is a useful feature.

5.8.2 Traffic Signals

Trips that are affected by one or more traffic signals has an average energy consumption of $166.74\frac{Wh}{km}$, opposed to $185.42\frac{Wh}{km}$ for trips that are not affected by traffic signals. This difference is a significant, in Table 5.5 we show additional information about energy consumption segments with traffic signals.

5.8.3 Roundabouts

In Table 5.6 we show how the different scenarios affect the energy consumption. It is interesting to see how much a roundabout affects the energy consumption, especially driving into and out of a roundabout effects the energy consumption greatly. As driving into a roundabout reduces the energy consumption, this is because the driver should decelerate when approaching a roundabout.

We also find that trips that are affected by one or more roundabouts on average uses $166.50\frac{Wh}{km}$ opposed $179.31\frac{Wh}{km}$ for trips that are not affected by roundabouts. We conclude that roundabouts are a good feature for energy consumption as it has a large impact as shown in Table 5.6.

Direction (Range)	Average ($\frac{Wh}{km}$)	Standard Deviation ($\frac{Wh}{km}$)	Variance ($\frac{Wh}{km}$)	$t(i)$ pairs
Right (80° – 100°)	156.48	457.84	209.62	307522
Straight (350° – 10°)	161.26	464.03	215.32	3222907
Left (260° – 280°)	164.61	561.09	314.82	190707
Back (170° – 190°)	387.65	1051.84	1106.36	85397

Table 5.4: Direction Impact

Traffic Signal	Average ($\frac{Wh}{km}$)	Standard Deviation ($\frac{Wh}{km}$)	Variance ($\frac{Wh}{km}$)	$t(i)$ pairs
Start & End	354.16	455.50	207.48	231488
Start	297.25	284.95	81.19	599451
End	34.71	665.24	442.54	598650

Table 5.5: Traffic Signal Impact

Roundabout Location	Average ($\frac{Wh}{km}$)	Standard Deviation ($\frac{Wh}{km}$)	Variance ($\frac{Wh}{km}$)	$t(i)$ pairs
Start & End	219.27	247.26	61.14	533202
Start	327.40	237.36	56.34	231075
End	29.38	349.77	122.34	230712

Table 5.6: Roundabout Impact

5.9 Problem Statement

Based on our initiating problem in Section 1.2 we have investigated some hypotheses about our problem, furthermore we have extended our data foundation in Chapter 4 to make a better prediction of energy consumption. The results from this work are the basis for the updating of our problem to the following problem statement.

> *Based on the data analysis we have found attributes that have an influence on the energy consumption. Is it possible to accurately predict an energy consumption based on these attributes for any arbitrary trip from point A to B, using machine intelligence techniques.*

Furthermore, we have chosen to make some delimitation based on the initial analysis. We will implement and test the elevation approaches 1 and 2 such that we can determine whether the more detailed approach adds value to the predictions. The driving style will not be used for predictions because the current finding does not contribute to a significant difference in energy consumption.

Method 6

The observations in our data are richly structured, i.e. the observations are associated with segments. This association of the data is taken into consideration for the development of our models. The objective is to predict an accurate energy consumption, for a new route based on features and the available historical observations.

The set of observations that are associated with a segment varies in size. Some segments have many observations, while others have few, or none. This association can be utilized, under the assumption that associated observations depict the energy consumption more accurately.

Since there are segments without associated observations, we consider how to predict the energy consumption on these. The energy consumption is given as a numerical value, this value is know for all $t(i)$ pairs. To predict a numerical value, we consider models from the machine learning class "Regression".

In Section 6.1 the notation which is used for the models are presented. Section 6.2 describes our Linear Regression Model, while Section 6.3 describes our Neural Network approach. These models are chosen on the premise that we are interested to see if features are independent. If the features are not independent the Neural Network will perform better. We devise a strategy to use the association between observations and segments in Section 6.4. The construction of features is described in Section 6.5

6.1 Model Notation

In this section, we introduce the notation used to describe our models. We have a set of trips $\mathbb{T} = \{t_1, \ldots, t_n\}$, and a set of segments $\mathbb{S}$. Each trip $t \in \mathbb{T}$ is a list of segments, $t = (s_1, \ldots, s_n)$, where each $s \in \mathbb{S}$. A trip is represented as a list because the sequence of the segments, signifies the order of which they are traversed.

For a $t \in \mathbb{T}$ we have a finite number of $t(i)$ pairs where i is i-th segment in the list t i.e. a trip on a specific segment. For each of these $t(i)$ pairs, there are a set of features. There are static features which are bound to the segment, such as category, and speed-limit. While other features are dynamic and depend on the specific $t(i)$ pair, such as time, and weather. All features are for a $t(i)$ pair is given by the set $F(t, i)$. The features are described in Section 6.5. In addition to the features, the $t(i)$ pairs are also labeled with an energy consumption.

To retrieve all $t(i)$ pairs for a given segment we have function $f_s : s \to t(i)$ such that $f_s(s) = \{t(i) \mid t \in \mathbb{T} \land s \in t\}$ which returns a set of $t(i)$ pairs that belongs to a given segment. This set may be $\emptyset$ as not all segments are traversed.

6.2 Linear Regression Model

Linear regression is one of the most commonly used models. In this section, we describe a multivariate linear regression model, and how the coefficients can be learned.

For our linear model we introduce a feature vector $\mathbf{x}_j$ for each of our $t(i)$ pairs. The feature vectors contain the values given by $G(t, i)$.

$$\mathbf{x}_j = \begin{bmatrix} time & temperature & \dots & category & speedlimit \end{bmatrix}^T \tag{6.1}$$

For each feature vector $\mathbf{x}_j$, we know the corresponding energy consumption e_j. The vector of all energy consumptions are represented as $\mathbf{e}$. The function for linear multivariate regression is then given by $f(\mathbf{x}_j) = w_0 + \sum_i w_i x_{j,i}$. In this function the coefficients we will learn are $w_0, \dots, w_n$. w_0 stands out, and this is the intercept. To generalize the function, it is common to introduce an attribute $x_{j,0}$ for all examples which are always equal 1[Russell et al., 1995] this allows us to define our function as the dot product of $\mathbf{w}$ and $\mathbf{x}_j$

$$f(\mathbf{x}_j) = \sum_i w_i x_{j,i} = \mathbf{w} \cdot \mathbf{x_j} \tag{6.2}$$

Given a vector of weights, we are interested the minimizing the loss. The loss is a measure of how much our prediction $f(\mathbf{x_j})$ deviates from the actual result e_j. A commonly used loss function is Mean Squared Error (MSE).

$$MSE = \frac{1}{n} \sum_{j=1} (e_j - \mathbf{w} \cdot \mathbf{x}_j)^2 \tag{6.3}$$

We show the dot product $\mathbf{w} \cdot \mathbf{x_j}$, to emphasize that the weights remain the same for each example $\mathbf{x_j}$ when we evaluate the loss. e_j is the observed energy consumption for the features $\mathbf{x_j}$. Our objective is to find the weights $\mathbf{w}^*$ that minimizes the loss function.

$$\mathbf{w}^* = \underset{\mathbf{w}}{\text{argmin}} \frac{1}{n} \sum_{j=1}^{n} (e_j - \mathbf{w} \cdot \mathbf{x_j})^2 \tag{6.4}$$

There are two common approaches for finding $\mathbf{w}^*$, namely normal equation and Gradient Descent (GD). Both approaches uses a gradient vector for the linear function. This vector is composed of partial derivatives, as shown in Eq. (6.5).

$$\nabla_{\mathbf{w}} MSE = \left(\frac{\partial MSE}{w_0}, \dots, \frac{\partial MSE}{w_n} \right)^T \tag{6.5}$$

These derivatives describes our the slope of our loss function. Suppose we have a simple function $f(x) = x^2$, the derivative for this function is $\frac{\partial f}{\partial x} x^2 = 2x$. This means for $x = 2$ the slope is $2 \cdot 2 = 4$. We are interested in finding $\nabla_{\mathbf{w}} MSE = 0$, i.e. the a minimum of the loss function.

The normal equation approach to finding $\nabla_{\mathbf{w}} MSE = 0$ is analytical. In the normal equation the weights $\mathbf{w}$ are isolated. Where the solution is given as $\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{e}$. Where $\mathbf{X}$ is a matrix of all $\mathbf{x}_j$. The normal equation approach is analytical, however this approach does not scale with many features and a large data set [Nielsen, 2015].

Instead we use the GD approach, where $\mathbf{w}^*$ is approximated. In GD the gradient is used to search for convergence by finding the optimal weights. There are some different methods rely on GD to find the optimal weights. We will explain GD, and a modified version called Stochastic Gradient Descent (SGD). In GD the weights are updated using the following equation:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} \sum_{j} (e_j - f(\mathbf{x}_j))^2 \tag{6.6}$$

In this equation $\nabla_{\mathbf{w}} \sum_j (e_j - f(\mathbf{x}_j))^2 \equiv \nabla_{\mathbf{w}} MSE$, we change the notation such that the summation of all data is shown explicitly. The result is GD uses all data in each iteration to find the weights. The learning rate is given by α, which determines the degree of the descent. A smaller learning rate will increase the number of iterations that is required to find a minimum. While a larger number can result in missing a minimum. As a result the choice of α is important for convergence.

The second approach is SGD differs from GD by updating the weights for each example $\mathbf{x_j}$ from our training examples.

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} (e_j - f(\mathbf{x_j}))^2 \tag{6.7}$$

The steps that are taken in a SGD process are shown in Algorithm 3. First, the training examples are shuffled, such that the order of the examples are changed. Then each example $\mathbf{x_j}$ from the shuffled training examples is iterated, and the weights are updated.

Algorithm 3 Stochastic Gradient Descent

```
Input: Training Examples
Shuffle Training Examples
for $\mathbf{x}_j$ in Training Examples do
    $\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} (e_j - f(\mathbf{x_j}))^2$
Output: $\mathbf{w}$
```

An optimization for SGD is called momentum which keeps information from previous iterations [Ruder, 2016]. The momentum is added by constructing a new vector $\mathbf{v}$ with the same dimensions as $\mathbf{w}$. The purpose of momentum finding a minimum in fewer iterations, by using knowledge from previous iterations.

$$\begin{aligned} \mathbf{v} &= \gamma \mathbf{v} + \alpha \nabla_{\mathbf{w}} (e_j - f(\mathbf{x_j}))^2 \\ \mathbf{w} &\leftarrow \mathbf{w} - \mathbf{v_t} \end{aligned} \tag{6.8}$$

Here γ determines the significance of the momentum. It is called the momentum term and the value is commonly set to 0.9 [Ruder, 2016]. The value is in the range $\gamma \in (0, 1]$.

The difference in the two approaches GD and SGD makes them suitable for difference purposes. Our choice is SGD because our training examples are of a relatively large size.

6.2.1 Feature Selection

For multivariate linear models, there is a concern of over-fitting where features that are irrelevant can by chance appear to be useful. Some approaches can reduce over-fitting and as a result enhance the generalization of the model.

One common strategy is a greedy approach in which different combinations of features are evaluated on a validation set. The best combinations of features are used on the test set.

Another popular method is Lasso Regularization, in which a regularization term is added to the loss function, shown in Eq. (6.9)

$$\frac{1}{n}\sum_{j=1}(e_j - \mathbf{w}\cdot\mathbf{x}_j)^2 + \frac{\lambda}{2}\sum_{i=1}|w_i|^q \tag{6.9}$$

Where λ is the regularization coefficient, which controls the impact of the loss function and the regularization term. The *lasso* is given by q, the lasso is commonly set to 1 or 2. Because we our objective is feature selection, we chose lasso $q = 1$. Which drives the weights to 0, resulting in a generalized model [Bishop, 2006]. Once the model has been constructed we can review the weights. Using the weights we can determine the impact of our features, if the weight has been set to zero the feature does not have an impact.

For our Linear Model, we use SGD with momentum, and we use Lasso Regularization.

6.3 Neural Network

Neural Networks are different from a linear regression because they have the capability to learn dependencies between input variables. There are different variations of Neural Networks. One common variation is feed-forward neural networks. The objective of a feed-forward network is to approximate a function $y = f^*(\mathbf{x};\boldsymbol{\theta})$. In this setting, the values of the parameters $\boldsymbol{\theta}$ are learned. The function maps the input $\mathbf{x}$ to a scalar output y. The $\mathbf{x}$ contains all the feature values, and the y is the known energy consumption. The vector with our feature values is referred to as the *Input Layer*, and the output is known as the *Output Layer*. In Fig. 6.1 we show an exemplary Neural Network with a *Hidden Layer* between in the Input and Output layers. We have one output unit, as our Neural Network will predict one scalar value. There can be any number of hidden layers in a network. The theory in the section is based on [Goodfellow et al., 2016; Nielsen, 2015].

Feed-forward networks are given this name as they are directed graphs with no cycles. Which means all information is fed forward from the input $\mathbf{x}$ to the output $\hat{y}$, which is the predicted energy consumption in our case. Typically a feed-forward neural network consists of several functions composed together. For example the three functions $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ are combined into a chain, each function is a layer. The number of functions there are in the chain is known as the depth of the network. The chain is seen in Eq. (6.10) where the $f(\mathbf{x})$ is the output and $f^{(1)}$ is the first layer (input layer), $f^{(2)}$ is the second and so on. The final layer $f^{(3)}$ is the output layer. Each layer between the input and output layer are called hidden layers.

$$f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(x))) \tag{6.10}$$

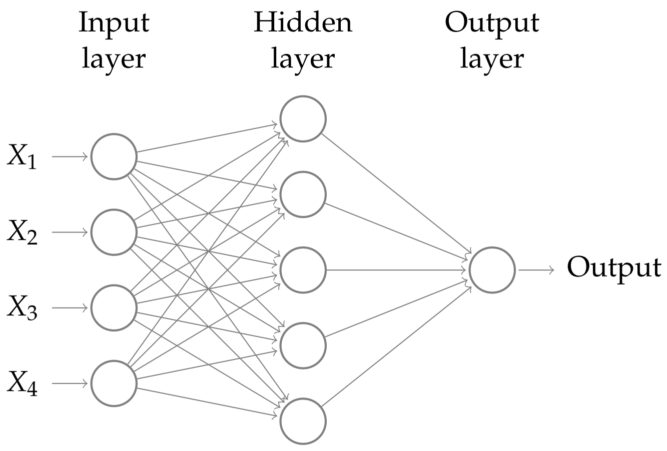


Figure 6.1: Exemplary Neural Network with 4 inputs nodes, 1 hidden layer and 1 output node

In each layer, there are a number of units (neurons). The number of units is also called the width. Each unit receives the inputs from the units in the preceding layer. The output of a unit is calculated based on an activation function, the inputs, some learned weights, and a bias. This is seen in Eq. (6.11) where the input to the activation function is z, $\mathbf{w}^T$ are the weights, $\mathbf{x}$ are the inputs and b is the bias. The output a from the unit is calculated using an activation function $a = g(z)$ where g is some activation function.

$$z = \mathbf{w}^T\mathbf{x} + \mathbf{b} \tag{6.11}$$

This output from $g(z)$ is used as the input to the next layer. The transmission between layers is shown in Fig. 6.1 where each units in each layer points to the each unit in the next layer. In our model, the target is to predict a scalar value $\hat{y}$ based on the input $\mathbf{x}$; this means in our model we have a single unit in the output layer. Our target is that the result from the output unit is the mean energy consumption of the Gaussian distribution based on the inputs $\mathbf{x}$.

Different activation functions are used in neural networks. Activation functions are used for the hidden layers and the output layer. For the output layer we use a linear activation function which combines the units from the previously hidden layer into one output. The output of a linear unit is given in Eq. (6.12) where a is the input from the layer before.

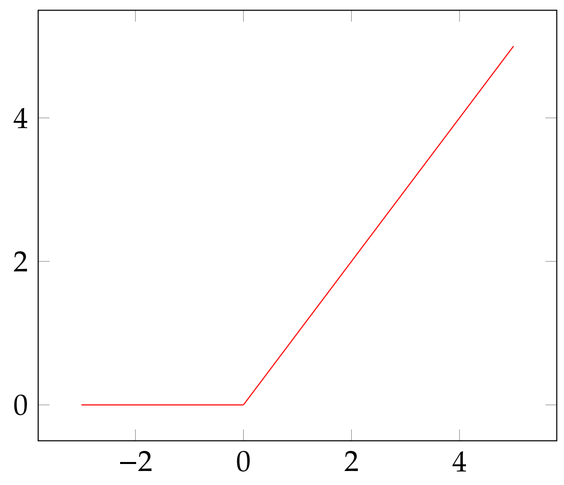
$$\hat{y} = \mathbf{w}^T\mathbf{a} + \mathbf{b} \tag{6.12}$$

For the hidden layers, we use rectifier activation function which is widely used, because learning a model with Rectified Linear Unit (ReLU) is fast [Goodfellow et al., 2016]. The ReLU is seen in Eq. (6.13) and plotted in Fig. 6.2. ReLU is zero for all z values below zero. A potential problem for ReLU is when the activation is zero; then the unit can not be learned using gradient-based methods. This problem can be solved using values above zero for the bias when the Neural Network is initialized.

$$g(z) = max\{0, z\} \tag{6.13}$$

When the network is trained the objective is to match a function $f(\mathbf{x})$ to the $f^*(\mathbf{x};\theta)$. Each sample in the training data includes the $\mathbf{x}$ and a know value y. For each sample, the function $f^*(\mathbf{x};\theta)$

Figure 6.2: ReLU



must result in a value as close as possible to y. Based on the input and output the learning algorithm is responsible for deciding how to use each layer to produce the expected output.

Neural networks are learned iteratively using a gradient-based optimizer where the target is to minimize the result from a loss function, this means an important aspect for learning the parameters **w** and **b** is to chose an appropriate loss function.

In our data, there is a large variation in the energy consumption for $t, (i)$ pairs that belong to the same segment. Because of this, we have chosen to use Mean Absolute Error (MAE) as the loss function. With this loss function, outliers in the data will have less influence on the final model. MAE is defined in Eq. (6.14), where n is the number of samples.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y - \hat{y}| \tag{6.14}$$

Before learning the model, the weights in the neural network are initialized assigned to random values. The loss function is optimized using SGD which is described in Section 6.2. Because there are multiple layers in a Neural Network and difference functions, the computation of the gradient is different. To compute gradient, we use a back-propagation algorithm. For each sample in training set, the predicted value $\hat{y}$ is calculated using the feed-forward procedure based on the current weights and bias. After the feed-forward process, the back-propagation is used for each sample to compute the gradient.

For referring to concrete weights, activation outputs and bias' in a neural network, we add a new notation.

- w^l_{jk} is the weight from the kth unit in the $(j-1)th$ layer to jth unit in the jth layer.
- b^l_j is the bias in the jth unit in the lth layer.
- a^l_j is the activation in the jth unit in the lth layer.

In Fig. 6.3, we show a diagram where the notation is used. The a^l_j is the activation outputs from

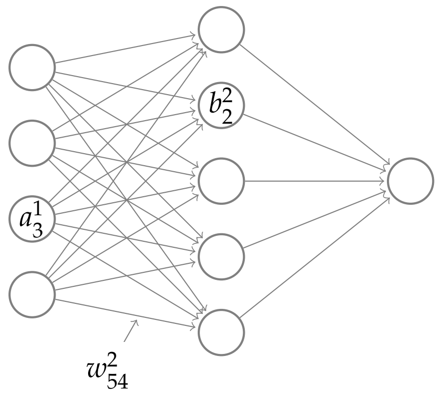


Figure 6.3: Neural Network with notation

the $(l-1)$ layer. a_j^l is given with the sum over all the units k in the $(l-1)$ layer. The formal definition is given in Eq. (6.15).

$$a_j^l = g(\sum_k w_{jk}^l a_k^{l-1} + b_j^l) \tag{6.15}$$

Eq. (6.15) can be written as vector in Eq. (6.16)

$$\mathbf{a}^l = g(\mathbf{w}^l \mathbf{a}^{l-1} + \mathbf{b}^l) \tag{6.16}$$

The idea of back-propagation is an expression to computing the partial derivative $\frac{\partial L}{\partial w}$ and $\frac{\partial L}{\partial b}$ for the loss function L for any weight and bias in a neural network. The expression says how fast the loss changes in relation to change of weights and biases.

To use back-propagation we make two assumptions about the loss function. The first assumptions are that the function can be written as an average over individual training set examples i.

This is the case for MAE. The total loss is $L = \frac{1}{n}\sum_i L_i$ and the loss for a single training example is $L_i = |y - a^l|$ where a^l is the vector of activation outputs for lth layer. This is requried to construct partial derivatives $\frac{\partial L_i}{\partial w}$ and $\frac{\partial L_i}{\partial b}$. Then we can compute the total derivatives $\frac{\partial L}{\partial w}$ and $\frac{\partial L}{\partial b}$ by averaging over training examples.

The second assumption is that we can find the loss as an output of the neural network, and this is also the case for MAE where the definition for a single training set example can be written as: $L = |y - a^l|$.

To calculate the $\frac{\partial L_i}{\partial w_{jk}^l}$ and $\frac{\partial L_i}{\partial b_j^l}$ we have to compute the loss δ_j^l. The δ_j^l is a little change to the z which influence the overall loss. To compute the loss in the output layer δ^L the following in Eq. (6.17).

$$\delta_j^L = \frac{\partial L}{\partial a_j^l} g(z_j^L) \tag{6.17}$$

This can be written as a matrix for δ^L as in Eq. (6.18), using the Hadamard product of $(a^L - y)$ and $g'(z^L)$.

$$\delta^L = (a^L - y) \odot g'(z^L) \tag{6.18}$$

This expression will result in a small error if the loss does not depend much on the specific unit j. The first part $\frac{\partial L}{\partial a_j^l}$ is compounded using the loss function. $\frac{\partial L}{\partial a_j^l} = (a_j^L - y_j)$. For the errors in the following layers, the definition is seen in Eq. (6.19).

$$\delta^L = ((w^{l+1})^t \delta^{L-1}) \odot g'(z^L) \tag{6.19}$$

Using these two definitions for errors, the error δ^l can be computed for all layers in the network. The rate of change for the bias is equal to the error. This rate of change is formally given in Eq. (6.20).

$$\delta_j^L = \frac{\partial L}{\partial b_j^l} \tag{6.20}$$

In the same way the rate of change in the weight is given by in Eq. (6.21). The computation of δ_j^l and a_k^{l-1} is already explained.

$$\frac{\partial L}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \tag{6.21}$$

Overall the back-propagation algorithm works as follows:

- Initialize: Set the starting weights and bias.
- Input: Input the features $\mathbf{x}$ to the input layer.
- Feed-forward: For each layer calculate $\mathbf{z^l} = \mathbf{w}^l \mathbf{a^{l-1}} + \mathbf{b^l}$ and $\mathbf{a^l} = g(\mathbf{z^l})$.
- Output error: Calculate the errors for the output layer $\boldsymbol{\delta}^\mathbf{l} = (\mathbf{a^l} - y) \odot g'(\mathbf{z^l})$.
- Back-propagate the error: For each layer from the output to the input calculated the error using $\delta^L = ((w^{l+1})^t \delta^{L-1}) \odot g'(z^L)$.
- Output: The gradients is calculated using $\frac{\partial L}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$ and $\frac{\partial L}{\partial b_j^l} = \delta_j^L$.

Hyperparameter optimization is described in Section 7.2.

6.3.1 Regularization

Similar to most other models, Neural Network will fit too well to the training set especially when the amount of data is large, this is called over-fitting and will potentially cause the model to predict poorly on new samples. Over-fitting can be solved using different approaches for making the model more general.We have chosen two different approaches in order to avoid over-fitting.

The first approach is Early Stopping, and instead of trying to find the lowest loss on the training set, the learning is stopped when the loss on the validation not are improved in some amount of time. In other words, the model is continuously tested on the validation set the model is stopped when there no improvements[Goodfellow et al., 2016].

The second approach is dropouts. Dropouts will with a defined probability randomly disable some units in the neural network when the network is trained. The disabled units will change for each new sample. This will result in various models with different amount of units are trained. When the model is used on the test data, the output of each unit will be adjusted about the probability. This method results in a model which are more robust to new data, and the model will have lower tendency of overfitting [Srivastava et al., 2014; Goodfellow et al., 2016].

6.4 Combination of Model and Observations

In addition to the linear model and the neural network, we construct a method to include the observations into our prediction $\hat{y}$.

As mentioned we have a variable number of observations linked to a segment, which can be combined with a model. In other words, for the route that we are going to predict, we may have some historical observations on some of the segments. We intend to include these observations in our prediction. In Eq. (6.22) we show an equation that uses the observations. Where $\hat{y}$ is the combined result, n is the number of observations, $\overline{x}$ is the average energy consumption from the observations, k is a learned weight, and m is the result from a model.

$$\hat{y} = \frac{n}{n+k} \cdot \overline{x} + \frac{k}{n+k} \cdot m \tag{6.22}$$

We have already a method to learn weights, namely SGD. We are going to apply to apply this method for approximating the optimal k value. We use the loss function MSE in our SGD to approximate k. The learned k will describe how much we will trust the prediction from a model m and the average energy consumption for a segment x, depending on the number of observations.

$$MSE = \frac{1}{n} \sum_{t} (y_t - \hat{y}_t)^2 \tag{6.23}$$

Where the y_t is the actual energy consumption and $\hat{y}_t$ is our prediction, from Eq. (6.22).

6.5 Feature Construction

In this section, we construct features from the variables that we have analyzed in Chapter 5. We describe how each feature is constructed. The feature construction is necessary to make a model for prediction energy consumption.

We construct each feature such that they belong to a $t(i)$ pair. All the features for a $t(i)$ pair will be represented as a vector.

The features are divided into two groups, static features, and dynamic features. Features that are from the segment of a $t(i)$ pair are called static because they will remain the same, independent of the trip. The dynamic features change according to the $t(i)$ pair. An example would be two different $t(i)$ pairs on driving in the same segment would have the same static speed-limit while the actual speed is dynamic. In Table 6.1, and Table 6.2 the static and dynamic features are listed, respectively.

Label	Type
Category	Discrete
Slope	Continuous
Zone	Discrete
Region	Discrete
Length	Continuous

Table 6.1: Static Features

Label	Type
Time of Day	Discrete
Day of Week	Discrete
Day of Year	Discrete
Speed	Continuous
Roundabout	Discrete
Traffic Signal	Discrete
Air Temperature	Discrete
Direction Change	Discrete

Table 6.2: Dynamic Features

6.5.1 Feature Scaling

We are working with features that contain outliers, for example, we know that the slope of a segment is a small number, but there are some segments with a large slope. For this reason, we standardize the slope, speed, and length. In Eq. (6.24) how standardization is performed. The average value for a feature is given by μ, while σ is the variance. Thus for a new value x the scaled value is given by x'.

$$x' = \frac{x - \mu}{\sigma} \tag{6.24}$$

6.5.2 Static Features

Static Features an independent of the trip because they a related to a specific segment.

Length

The length is the total length of a segment from OSM. The unit of the length is in meters. The features length is represented as a floating point in the feature vector as a one dimension vector.

Zone

As mentioned in Section 4.3 we have mapped all segments to City, Rural, Cottage, or Unknown zones. This information is represented as a binary vector of length four.

Common for our representation of binary vectors is that 1 signifies membership, while 0 means absence of membership. As mentioned the binary vector for zone has four elements, in Eq. (6.25) we show which element corresponds to to which zone.

$$\begin{bmatrix} City_{e1} & Rural_{e2} & Cottage_{e3} & Unknown_{e4} \end{bmatrix} \tag{6.25}$$

For example, to represent that a $t(i)$ is in rural zone the vector is $[0, 1, 0, 0]$. This representation makes sense because there is no definition of closeness between two zones.

Slope

In Section 4.2.3 we showed how the slope found for the segments. We found that the slope has a linear relationship with the energy consumption. Now we consider how the slope information should be included. In our project, we have chosen to only test Approach 1 and Approach 2. How we construct the feature for each of them is described separately below.

Slope - Approach 1 In Approach 1 we found the slope for each segment. Each slope is a numeric value. Because there is one numeric value per segment and the correlation with the energy consumption is linear the feature is represented as a numeric value.

Slope - Approach 2 In approach 2 we divided segments into the sub-segments and found a slope for each sub-segment. The number of sub-segments for each segment is not always the same. As such we consider a how the information can be used. If we were to construct a feature for each slope, we would have a disparity, and this will not work. Instead, we consider how we can represent the information in a fixed number of features. Here we consider aggregating information into an average slope and the standard deviation. We know the sign of the slope is important, as such we also include the average and standard deviation for negative slopes and positive slopes.

From these considerations, we construct the following features for slope.

1. Average for all slopes
2. Standard Deviation for all slopes
3. Average for all positive slopes
4. Standard Deviation for all positive slopes
5. Average for all negative slopes
6. Standard Deviation for all negative slopes

The attributes are chosen because they can express both the uphill and downhill slopes for the sub-segments.

Each of these features is represented as a numeric value. If there is only one slope for a given segment the standard deviations are set to 0. If a slope is negative the positive average is set to 0, and the other way around.

Region

As mentioned in Section 3.2, we have access to information about which region each segment is in. There are five regions in Denmark, however 452 segments are not covered by a region. For this we construct a binary feature vector.

$$\begin{bmatrix} Nordjylland_{e1} & Midtjylland_{e2} & Syddanmark_{e3} & Sjælland_{e4} & Hovedstaden_{e5} & Unknown_{e6} \end{bmatrix} \tag{6.26}$$

If a segment is not in a region they are represented as Unknown. For example if a segment in not in a region it will be represented as $[0, 0, 0, 0, 0, 1]$.

Category

From OSM we know that category of each segment. There is a total of 17 different categories. We represent this knowledge as a binary feature.

$$\begin{bmatrix} Motorway_{e1} & MotorwayLinks_{e2} & Trunk_{e3} & \ldots & Unpaved_{e17} \end{bmatrix} \tag{6.27}$$

The full list of categories is Motorway, Motorway Links, Trunk, Trunk Links, Primary, Primary Links, Secondary, Secondary Links, Tertiary, Tertiary Links, Unclassified, Residential, Living Street, Service, Road, Track, Unpaved.

6.5.3 Dynamic Features

The dynamic features are the features which change according to a given trip. For example, time is dependent on when a trip started and how fast the driver is going on the trip. In this subsection, we describe how we handle the dynamic features.

Time of Day

The time feature is special for our project, and it specifies when during the day the trip is taking place. For each $t(i)$ for a given trip t, we have a time key that specifies when the segment was reached. Because we are emulating a real world setting, we only use the starting time of a trip. Using the starting time we estimate the time that a $t(i)$ will reach a segment. In Table 6.3 we show an example of the actual times for a trip and the time we estimate.

The time estimate is calculated by taking the current time and adding the time it takes the traverse the segment at the speed-limit for the segment, and we show this in Eq. (6.28).

$$t(i).timeestimate = t(i{-}1).timeestimate + t(i{-}1).length \cdot t(i{-}1).speedlimit \tag{6.28}$$

With the exception of the first $t(i)$ pair where the time estimate is set to the actual start time of the trip. For the purpose of the calculations the seconds are included, the seconds are not saved for the estimate instead the time is rounded to the nearest minute. We round to the nearest minute, s.t. the estimated time is similar to the actual time.

In the current state, our time feature is a discrete feature with $24 \cdot 60 = 1440$ possible values, i.e. a value for each minute of the day.

Pair	Actual Time	Estimated Time	Actual Speed *km/hr*	Estimated Speed *km/hr*	Segment Length *m*
$t(1)$	14:56	14:56	22	50	30
$t(2)$	14:56	14:56	23	50	484
$t(3)$	14:58	14:56	30	50	142
$t(4)$	14:58	14:56	43	50	235
$t(5)$	14:58	14:56	64	80	2908
$t(6)$	15:01	14:58	68	80	733
$t(7)$	15:02	14:59	50	80	397
$t(8)$	15:02	14:59	59	80	293
$t(9)$	15:02	14:59	62	80	47
$t(10)$	15:02	14:59	63	80	201
$t(11)$	15:03	15:00	26	50	686

Table 6.3: An example from our data, showing the actual time for a trip and the time that has been estimated.

We represent time with two features, and these features are shown in Eq. (6.29). Using these two features to represent time, we ensure that each minute receives a unique value. This method is circular, and this means that it captures the gap between 23:59 and 00:00 [Bishop, 2006].

$$time_{cos}(minute) = cos(2 \cdot \pi \cdot minute/1440) \tag{6.29a}$$

$$time_{sin}(minute) = sin(2 \cdot \pi \cdot minute/1440) \tag{6.29b}$$

In Table 6.4 we show how the combination of $time_{cos}$ and $time_{sin}$ remains unique throughout the minutes of a day, and how the the gap between 23:59 and 00:00 is captured.

Day of Week & Day of Year

We use the same approach for representing the day of the week, and day of the year as in the time of day. Equation (6.29) is modified such that for the day of the week the value is divided with seven instead of 1440, and for the day of the year, it is divided with 365.

Speed

For future trips we construct the speed feature as the speed limit for a segment. While we train our model with the historical data, were we have access to the actual speed for a given $t(i)$ pair.

HH:MM of Day	$time_{cos}$	$time_{sin}$
00:00	1	0
00:01	0.99999	0.00436
00:02	0.99996	0.00872
⋮		
06:00	0	1
06:01	-0.00436	0.99999
06:02	-0.00872	0.99996
⋮		
12:00	-1	0
12:01	-0.99999	-0.00436
12:02	-0.99996	-0.00872
⋮		
18:00	0	-1
18:01	0.00436	-0.99999
18:02	0.00872	-0.99996
⋮		
23:59	0.99999	-0.00436
00:00	1	0

Table 6.4: Values of $time_{cos}$ and $time_{sin}$

Roundabout

For a $t(i)$ pair we know whether there is a roundabout at the beginning of the segment, or at the end, or none. We can represent this information in a binary vector with two elements. Different from the previous binary vectors, this vector can contain any combination of zeros and ones.

$$\textbf{roundabout} = \begin{bmatrix} Beginning & End \end{bmatrix} \tag{6.30}$$

For example, a segment with roundabouts at the beginning and start would be represented as $[1, 1]$ while a segment without any roundabouts would be $[0, 0]$.

Traffic Signal

The traffic signal feature is modeled in the same way as the roundabout feature. For a $t(i)$ pair we if there is a traffic signal at the beginning and end of the segment. This information is represented by a binary vector with two elements. This vector can contain any combination of zeros and ones.

$$\textbf{trafficsignal} = \begin{bmatrix} Beginning & End \end{bmatrix} \tag{6.31}$$

Direction Change

The direction change as a discrete value between 0 and 359. Where 0 represents going straight, 90 is right, 180. In our data analysis we found that several directions are only observed few times. This means we should consider over-fitting, especially in our linear model. As a result we have decided to group the direction changes into 4 distinct groups, namely *Forward*, *Right*, *Backward*, and *Left*. We show this in Fig. 6.4, where forward is between [330°−30°] , right between (30°−150°), backward between [150° − 210°], and left between (210°−330°).

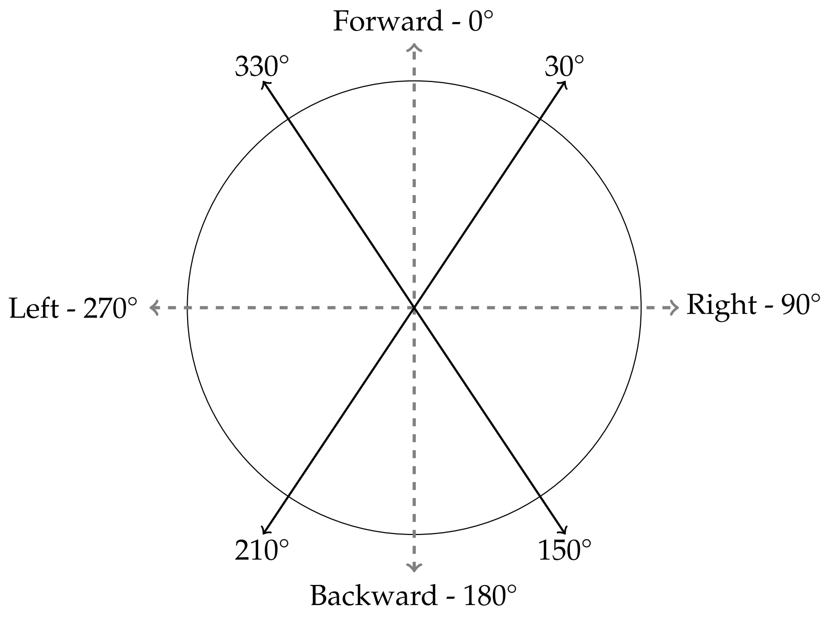


Figure 6.4: Different direction changes

This means we can store this information in a binary vector with 4 elements.

Experiments 7

In this chapter, we consider the dataset used for experiments. How the hyperparameters for the models is adjusted. The baselines that will be used for comparison to our models, and the results from our experiments.

7.1 Data Set

We require three data sets, namely a training, validation, and test set. We include a validation set such that we can adjust hyper-parameters of our models.

A common approach is to put trips into training, validation and test sets randomly. This method means we would pay no attention to the fact that we want to emulate future trips. As the training, validation and test sets would possibly contain trips from all periods of time. A better approach to do this with respect to the problem statement would be to split the dataset by date and time. Then our training set can be considered as historical data, while the validation and test set as future data. This approach would emulate a real world setting closely. Our choice is the second approach because it emulates a real world setting.

Our data-set includes 247 832 trips which consist of 159 913 546 observations. We do not use the individual observations directly but instead an aggregated value for each $t(i)$ pair. The data set includes 13 166 073 $t(i)$ pairs over 197 437 segments.

The dataset is split up into a training set, validation set, and test set as follows:

Training Set Samples used to learn a model. (198 266 trips 80%)
Validation Set Samples used to adjustment the model. (12 488 trips 5%)
Test Set Samples used to evaluate the model. (37 078 trips 15%)

This split ratio is commonly used, the training set is larger such that the model becomes more robust.

Emulating the real world setting as close as possible will give a larger trustworthiness of the predictions. As mentioned earlier in Section 3.4 only a subset of the road network has observations. Therefore consider that our validation and test set should include segments without observations. As shown in the Section 5.3 the energy consumption varies in different seasons, due to the temperature. Therefore our sets must be representative of the seasons, and this has we ensured by splitting on the following ranges.

Training Set From 01-04-2012 to 01-06-2013
Validation Set From 01-06-2013 to 26-05-2014

Test Set From 01-06-2013 to 26-05-2014

While the validation and test cover the same period, they do not contain the same trips. In other words, we first split our data into training and test sets. Then from the test set, we randomly take 12 488 trips and use these for our validation set. In this approach our validation set is representative of our problem, allowing us to perform our hyper-parameters adjustment using the validation set. From the above-described split, we validate that our validation set includes 6724 segments which are not a part of the training set while the test set contains 11 939 segments which are not.

7.2 Hyper-parameter optimization

The hyperparameters for the neural network used in our experiments is found using a random search approach because it is an efficient way for hyperparameter optimization [Bergstra and Bengio, 2012]. The random search is done on only a subset of the training set (1000000 samples) and the full validation set.

In the random search the following hyper-parameter are adjusted:

- Number of hidden layers
- Number of units in each layer
- Learning rate
- Momentum

Based on the result of the random search our neural network has the following values:

- Number of hidden layers: 1
- Number of units in each layer: 125
- Learning rate: 0.01
- Momentum: 0.83
- Initial bias: 0.2
- Initial wights: random
- Dropout: 0.5

Initial bias, initial weights, and dropout are selected based on our knowledge and the selected activation function.

For the linear regression, we have manual try to adjust the hyper-parameter and evaluate them on the validation set. It is led to the following parameters:

- λ for L1 Regularization: 0.003
- Learning Late: 0.001
- Momentum: 0.90

7.3 Baselines for Comparison

In this section, we construct three baselines. The purpose for these baselines is to determine how well our models perform.

We consider the following three baselines.

Average $166.2\frac{Wh}{km}$ based on the training set we find the average energy consumption for all segments.

Peugeot $106\frac{Wh}{km}$ The energy consumption specified by [Peugeot, 2017].

Category The average energy consumption for all categories based on the training, the values are shown in Table 7.1.

7.4 Evaluation Measures

In this section we consider different evaluation measures. The predictions are performed for each $t(i)$ pair, the evaluation is performed on the aggregated energy consumption for all $t(i)$ paris in a trip t. In our data there are trips of many different lengths, which results in vastly different energy consumptions. Because of the high variation, the focus will be on scale-independent metrics. We chose to use two measures namely Mean Absolute Percentage Error (MAPE) and Symmetric Mean Absolute Percentage Error (sMAPE), shown in Eq. (7.1) and Eq. (7.2). Where n is the total number of samples in the test set, a_i is the actual value and p_i is the predicted value for each sample. MAPE is sensitive to outliers, i.e. if a the actual energy consumption is near zero [Hyndman and Koehler, 2006], and our prediction is a larger number the error becomes large, as shown in Eq. (7.3). For this reason sMAPE is included, as outliers can not be penalized

Category	Average Energy Consumption ($\frac{Wh}{km}$)
Road	278.17
Tertiary Link	276.83
Living Street	269.74
Service	223.60
Residential	197.75
Unclassified	170.27
Unpaved	173.88
Trunk	167.34
Motorway	164.45
Primary	161.31
Tertiary	157.69
Track	154.33
Secondary	152.86
Motorway Links	130.85
Primary Links	96.73
Trunk Link	82.99
Secondary Links	63.22

Table 7.1: Average energy consumption for category sorted by energy consumption.

more than 200%.

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{a_i - p_i}{a_i} \right| \tag{7.1}$$

$$sMAPE = \frac{100\%}{n} \sum_{i=1}^{n} \frac{|p_i - a_i|}{(|a_i| + |p_i|)/2} \tag{7.2}$$

$$\frac{100\%}{1} \left| \frac{-0.09 - 2}{-0.09} \right| = 2322.2\% \tag{7.3}$$

Using this combination of measures, we are able to identify outliers with MAPE, while sMAPE provides a general indication. It is noteworthy that sMAPE, despite the name, is not symmetrical. It is not symmetrical because it does not treat over- and underestimations the same, underestimations are penalized by a slighter higher percentage.

7.5 Results

The results are shown and interpreted in this section. First a brief evaluating of the baselines are shown. Then we evaluate the quality of our features. Lastly, we evaluate the performance of the methods, to find in which areas the performance is best, and which it is lacking.

7.5.1 Baseline

As mentioned in Section 7.3 we will compare our results with three different baselines. The baselines are evaluated on the test set, and the results are shown in Table 7.2. The values in parentheses are sMAPE and the other values are MAPE. For all trips, we that MAPE and sMAPE are different for “Mean” and “Category” while they remain nearly the same for ’Peugeot’. Because the values are different for Mean and Category, we can determine that are some outliers which increase the percentage error. While the higher sMAPE for Peugeot is expected as the energy consumption is very low which causes underestimations. This tendency of underestimation is present for all trip lengths for Peugeot. The Category has the best performance of the three baselines.

7.5.2 Features Evaluation

We have constructed some features, in this section, we examine the quality of these features. We have already shown in Chapter 5 that there are many factors which impact the energy

Name	All Trips	Trips over 2 km	Trips over 10 km
Peugeot	55.02% (55.20%)	43.26% (50.59%)	34.37% (43.10%)
Mean	53.23% (32.46%)	34.45% (27.21%)	20.07% (19.13%)
Category	55.22% (31.80%)	34.77% (26.89%)	19.30% (18.90%)

Table 7.2: Baselines Performance

consumption. We show how the features impact the models. For the feature evaluation, we use our models.

We begin by using Linear Regression with L1 regularization. The L1 regularization tends to produce sparse coefficients, which means it sets coefficients to zero. The coefficients of features which are set to zero have no impact. With this knowledge, we can determine which features contribute to the prediction. We find that the following features contribute to the prediction.

- Category
- Zone
- Region
- Direction Change
- Roundabouts
- Slope
- Day of year
- Traffic signal
- Speed-limit
- Temperature

The features which have a zero weight are the following:

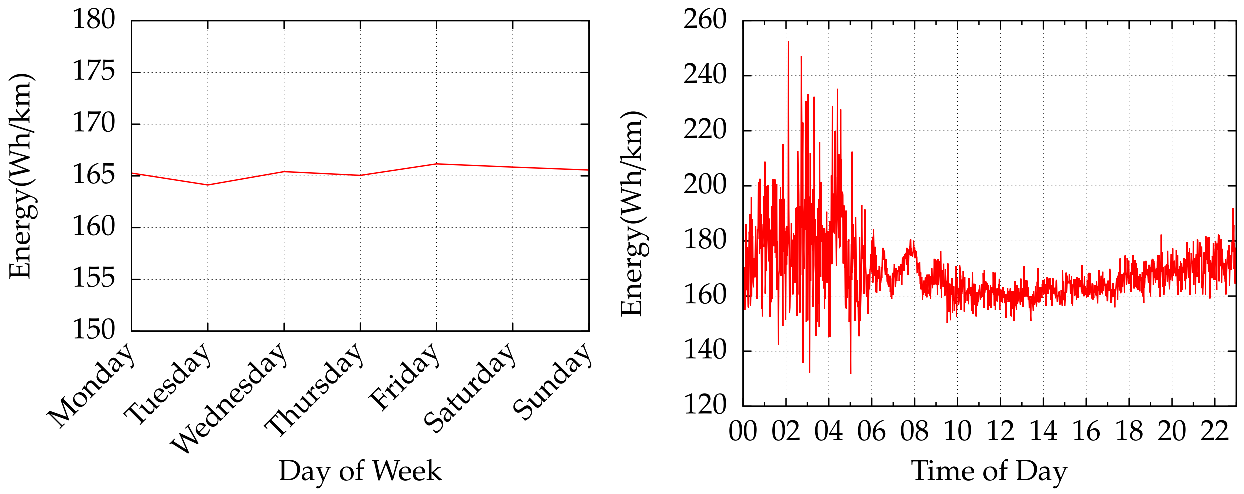
- Historical average energy
- Number of historical observations
- Time
- Day of week

We anticipated that the Linear Regression would not improve using a historical average energy consumption as this feature is sparse. It is sparse as we do not have observations for all segments. Thus segments without observations have no average energy consumption. The same case is true for the number of historical observations.

Time of day and day of the week have no impact on the Linear Regression. In Fig. 7.1, we show the average energy consumption during the week and the day. From the figure, we see that the average energy consumption during the week is almost stable. In contrast, the energy consumption over time of day has a large fluctuation. However, fluctuation remains around the same energy consumption. Because the energy consumptions are stable, it is sensible that they do not impact the result.

We have tested different combination of features with the same hyper-parameters in our Neural network. Despite the hyper-parameters remaining the same, we assume the results are valid indicators of which features provide an impact. In Table 7.3 the combination of features is shown, the features which are not shown in the table are always included. The values in parentheses are sMAPE and the other values are MAPE. The first model in the table is *NN1*, which will be our reference model. It is used as the reference because it contains all features, except actual time and speed, and slope 2. We are then able to see the effect of using the actual time and speed, and a richer notation of slope by using these features in other models.

In *NN2*, our direction change feature is omitted. Contrary to our analysis, the direction change feature does not improve the performance of the model. Upon reviewing the direction change



(a) Day of Week vs. energy consumption (b) Time vs. energy consumption

Figure 7.1: Energy consumption for time and day of week

	Performance			Features									
Model	All Trips	Trips over 2 km	Trips over 10 km	Slope 1	Slope 2	Speed-limit	Estimated Time	Actual Speed	Actual Time	Roundabout	Traffic Signal	Air Temperature	Direction Change
NN1	46.41% (30.59%)	30.75% (**24.83%**)	19.04% (17.97%)	✓	×	✓	✓	×	×	✓	✓	✓	✓
NN2	46.93% (**30.32%**)	30.59% (24.85%)	18.21% (**17.36%**)	✓	×	✓	✓	×	×	✓	✓	✓	×
NN3	48.07% (30.74%)	31.35% (25.26%)	19.56% (18.51%)	✓	×	✓	✓	×	×	×	✓	✓	✓
NN4	46.17% (31.03%)	30.36% (25.20%)	18.79% (17.66%)	✓	×	✓	✓	×	×	✓	×	✓	✓
NN5	47.06% (32.61%)	32.04% (26.70%)	19.67% (18.86%)	×	×	✓	✓	×	×	✓	✓	✓	✓
NN6	45.60% (31.21%)	30.15% (25.15%)	18.47% (17.67%)	✓	×	✓	✓	×	×	✓	✓	×	✓
NN7	44.71% (32.59%)	30.01% (26.46%)	17.99% (18.21%)	×	✓	✓	✓	×	×	✓	✓	✓	✓
NN8	**41.70%** (33.59%)	28.77% (27.47%)	16.97% (18.68%)	✓	×	×	×	✓	✓	✓	✓	✓	✓
NN9	42.05% (32.72%)	**28.48%** (26.59%)	**16.37%** (17.59%)	×	✓	×	×	✓	✓	✓	✓	✓	✓

Table 7.3: Combination of Features for Evaluation

feature, we find that the direction of the last $t(i)$ pair in each trip is set to 0. Because it is set to 0, the feature is set to be a forward direction. Instead, it should have been a separate direction change, such that it could have been labeled as 'End'.

The roundabout feature provides an increase in performance, by removing it in *NN3* the overall MAPE increases by ~1.3%.

MAPE for Traffic signal in *NN4* is slightly increased. Our presumption was that traffic signal information would improve the prediction. In Fig. 7.2 we show the variance is significantly higher when there a traffic signal at the end of a segment. In addition to the variance, the distribution is multimodal. The distribution is multimodal on segments with traffic signals at the end because there are three distinct possibilities at a traffic signal; A driver makes a green light he does not stop; A driver stops completely for a red light causing a regenerative deceleration; A driver slows down while waiting for a green light without stopping. This multimodal distribution makes it difficult predict segments with traffic signals at the end.

Against our expectations, the temperature does not increase the performance seen in *NN6*. We suspect this is caused by a high variance in energy consumption for temperatures. This is seen in Fig. 5.3. The high variance makes it difficult to makes a good prediction. Another factor may be that the Neural Network makes better use of the day of year feature, and can catch the change in temperature and weather.

As expected the slope improves the performance, as *NN5* without slope information has an increase in error. In addition the more detailed slope 2 in *NN7* improves the MAPE significantly.

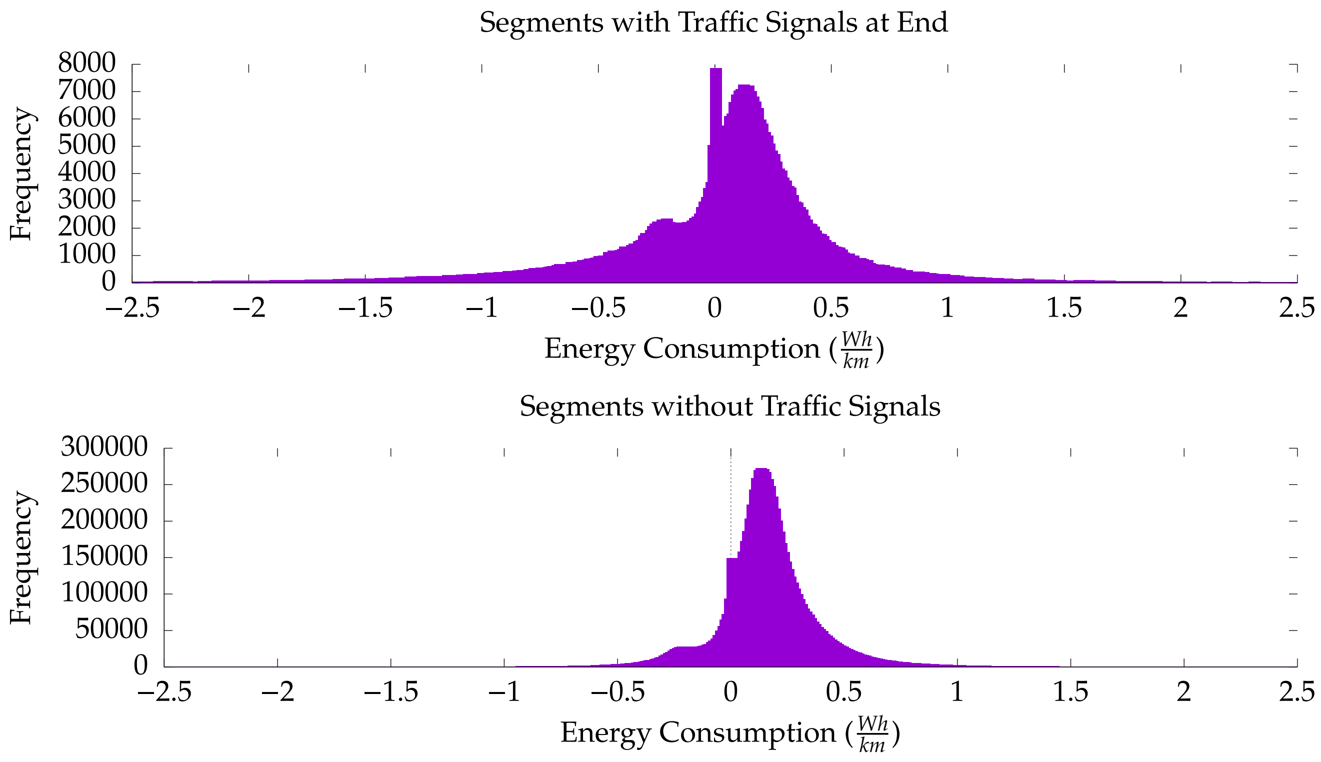


Figure 7.2: Distribution of energy with and without traffic signals

In *NN8* and *NN9* MAPE is improved. This improvement means the actual speed and time is better than our predicted values. For the total value, it seems that *NN8* is better than *NN9*. This result is probably caused by high variance in energy consumption especially when the speed is low as seen in Section 5.1. For longer trips, *NN9* performs the best.

We consider *NN7* to be our best Neural Network as *NN8* and *NN9* uses features we consider to be unavailable.

7.5.3 Performance Evaluation

In this section will we examine the performance of our three models performs. It is the following models; Linear Regression (LR) using all our features and L1 regularization; The neural network with the best MAPE *NN7*; A combination of the *NN7* and our observations called *NN-Observations*. We have used the validation set to learn the value k as described in Section 6.4. The resulting k value is 24.4742. With this k value the observations and the model prediction are weighted as seen in Fig. 7.3. When we have 35 observations for a segment, the observations and prediction are weighted equally.

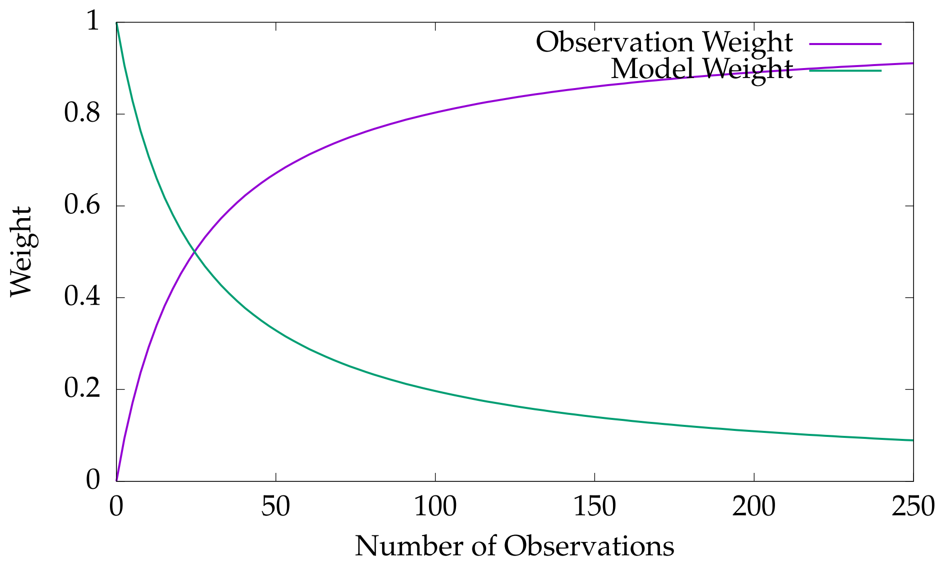


Figure 7.3: Weights in Linear combination

The overall results for each of the model is shown in Table 7.4. The values in parentheses are sMAPE, the other values are MAPE. The LR model has a high MAPE when compared to NN7 (NN), however the sMAPE is lower. From the results we see that NN performs worse on shorter trips. NN-Observations is our Neural Network combined with observations. This combination outperforms the two other models.

In Fig. 7.4 we show how the models predict for all trips. In the figures, the predicted energy consumption is compared to the actual energy consumption. We see that Mean & Category predictions are distributed around the diagonal evenly. This behavior is expected as they represent the total average energy consumption and the average for categories, respectively. We see that Peugeot underestimates the energy consumption on the majority of trips.

Name	All Trips	Trips over 2 km	Trips over 10 km
LR - Linear Regression	51.26% (29.74%)	33.02 % (24.98%)	20.73% (18.78%)
NN7 - Nerual Network	44.71% (32.59%)	30.01 % (26.46%)	17.98% (18.21%)
NN-observations - Linear combination	**19.61% (14.08%)**	**12.08% (10.34%)**	**5.74% (5.90%)**

Table 7.4: Overall experiments results.

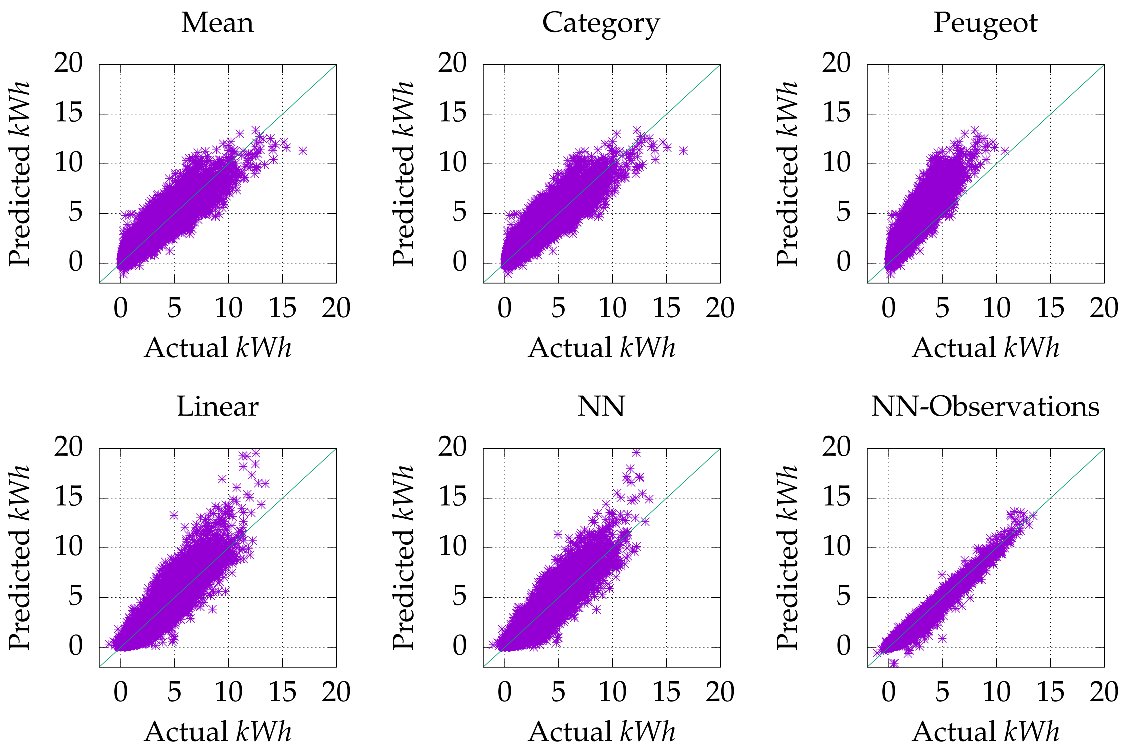


Figure 7.4: Comparison of Models. Actual energy consumption is compared to the predicted consumption.

For our models LR & NN, we see that they overestimate trips which have a significant energy consumption. Lastly, we see that our combination of observations and our Neural Network, NN-Observations estimates well. We now evaluate the performance of length, month, and hour of the day. Using these parameters for performance evaluation will provide a general overview of where the models are strong and weak.

Figure 7.5 shows the performance for predictions over length. There is a larger error for shorter trips for both our measures. This error is caused by short trips having high variation in acceleration, whereas for longer trips the acceleration tends to be stable. In other words, for a short trip, a driver must accelerate his vehicle which in turn consumes energy. We also see that NN predicts better for longer trips, compared to LR.

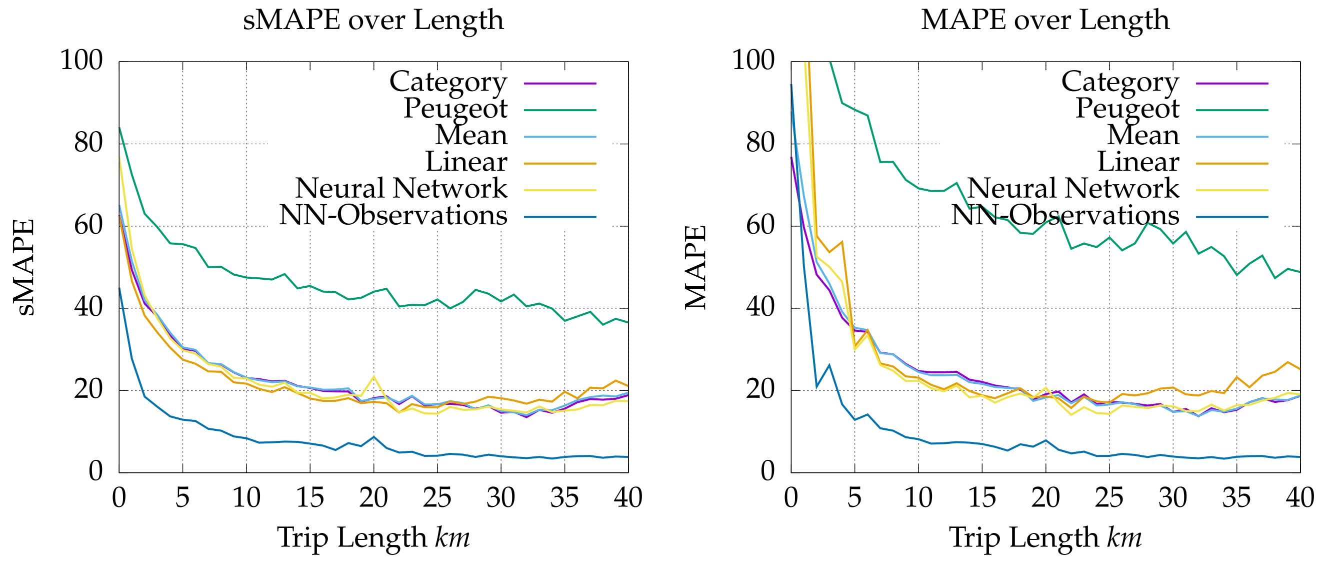


Figure 7.5: sMAPE&MAPE at given trip lengths

There are some interesting peaks Fig. 7.6 for MAPE. The peaks in March and July do not occur in sMAPE, and this indicates there are outliers. In Table 7.5, the outliers are shown. Upon further inspection, the outliers are caused by actual energy consumption that is near zero causing a high percentage error. The energy consumption for these three trips is very low when looking at the length of the trips. Unsurprisingly the curve for Peugeot forms a parabola, as the predictions are more accurate in the warmer months because the energy consumption is lower in these months. The performance of our models appears stable for each month.

In Fig. 7.7 we show the performance during the hours of the day. Again the outliers are present, and these are the outliers from Table 7.5. For these graphs, we have included the number of

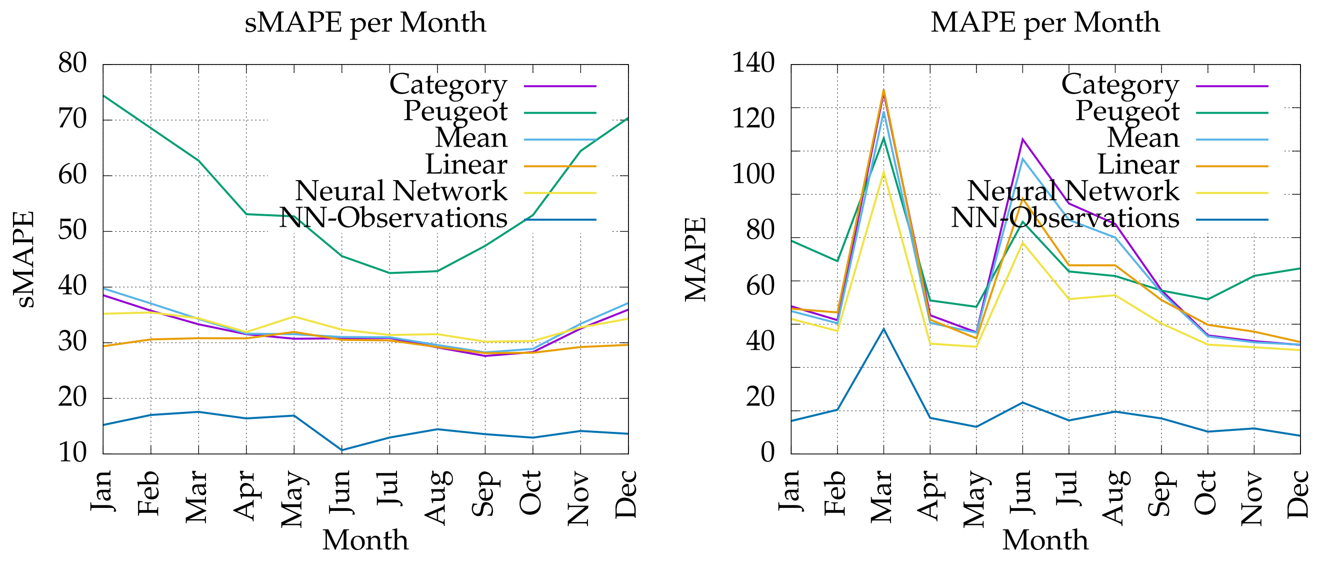


Figure 7.6: sMAPE&MAPE per month

Trip	Length (m)	Month	Actual (Wh)	Predicted (Wh)	MAPE
270257	725	March	-0.098	127.913	130322%
227433	1388	June	0.338	202.322	59772%
231683	280	June	0.149	45.278	31937%

Table 7.5: Outliers in data, the predicted value is from the linear model.

trips. We see that when there are many trips, the performance is stable.

Since we are predicting energy consumption for segments, we also evaluate the segment prediction. For the evaluation of predictions on segments, we use NN-observations. We perform this evaluation to find areas the model has strengths and where it can be improved. In Fig. 7.8, and Fig. 7.9 we show two areas of Denmark. The sMAPE values are found per segment, only segments in the test set are shown in the two figures. An interactive map is available at `http://energi.elefsennet.dk`. Details about the map can be seen in Appendix C. In Fig. 7.8 we see the prediction are best on arterial roads. The predictions on smaller roads are worse than arterial roads. In the Fig. 7.9, there are many small roads where the prediction is poor; however, we see that on the arterial roads our predictions are good. From these two maps see that the smaller roads are difficult. In Table 7.6 we show the error for segments in each region. Our predictions are better in the regions with more data in our training set, as expected. As expected we predict best on the arterial roads because of uniform driving, while the smaller roads can be improved.

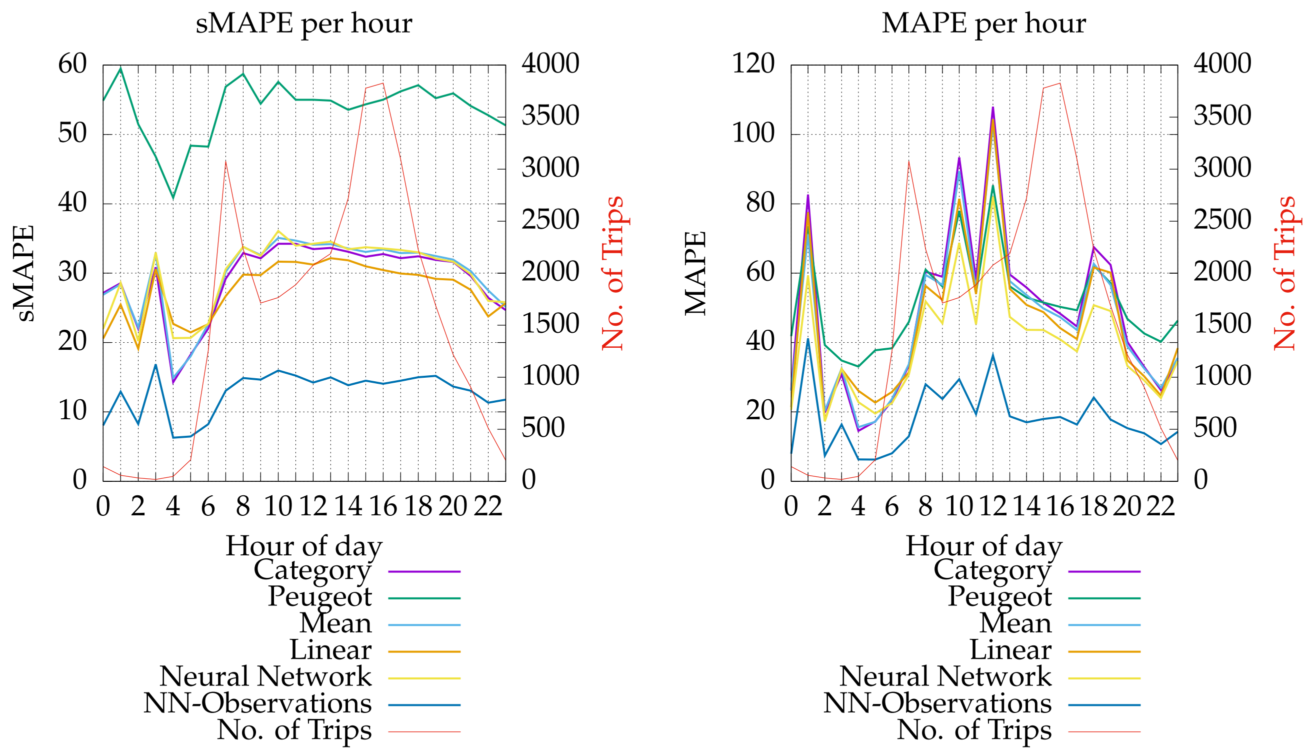


Figure 7.7: MAPE during the hours of the day

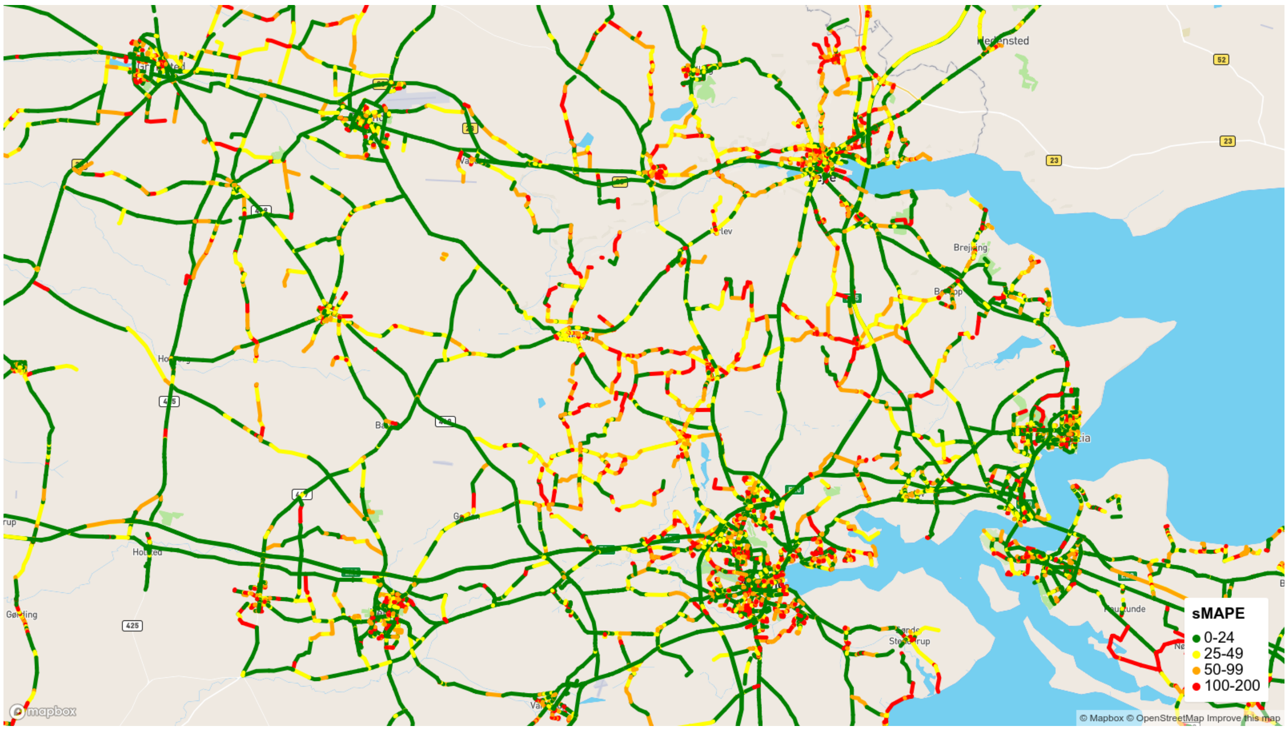


Figure 7.8: Southern Jutland

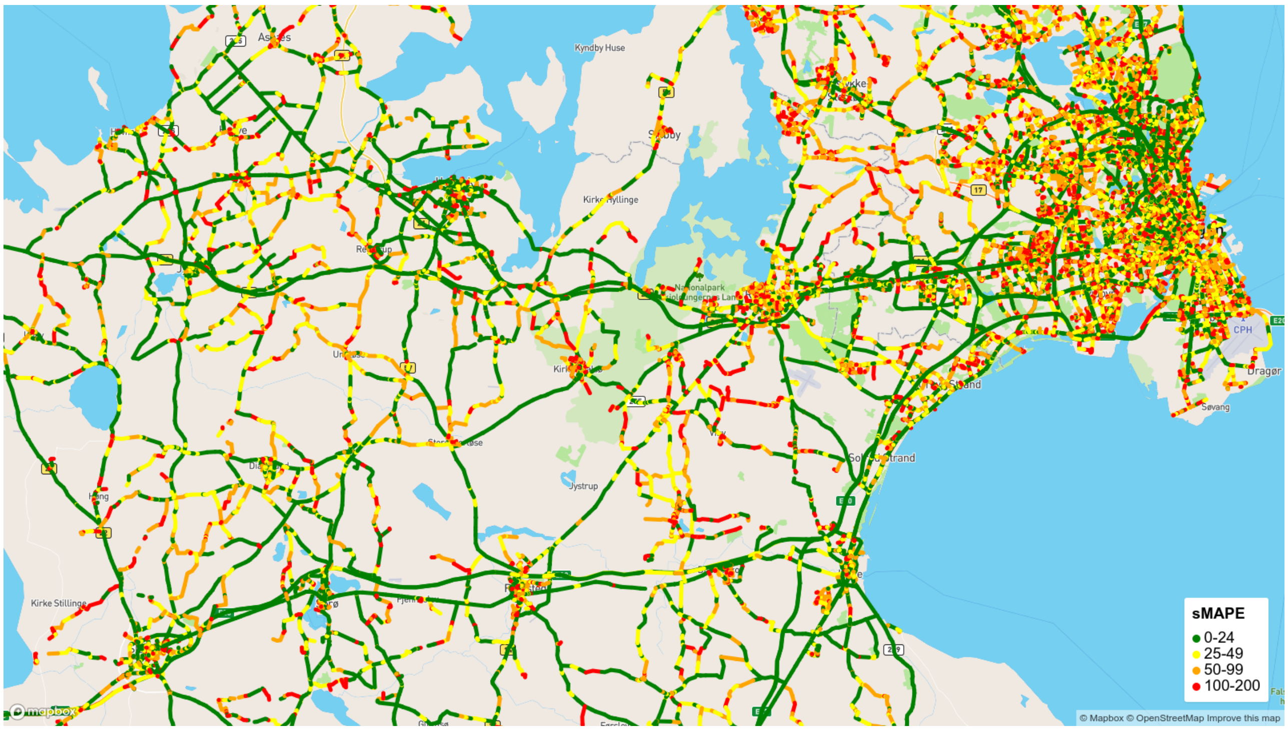


Figure 7.9: Zealand

MAPE	Trips in Test set	Trips in Training set	Region
66.9%	3497	478186	Region Nordjylland
81.3%	506	4309	Unknown
95.0%	279192	5239721	Region Syddanmark
119.4%	328288	1897996	Region Sjælland
131.0%	1277595	1361873	Region Hovedstaden
227.4%	12979	401390	Region Midtjylland

Table 7.6: MAPE for each Region

MAPE	Trips in Test set	Trips in Training set	Category
23.6%	104022	207358	Motorway
48.6%	187082	930566	Primary
49.3%	30361	92875	Trunk
70.1%	386824	2281705	Secondary
85.5%	545	5385	Primary link
95.9%	658305	3011324	Tertiary
96.2%	22869	4773	Tertiary link
122.0%	3155	6515	Trunk link
176.3%	162450	1033585	Unclassified
265.3%	308965	1594813	Residential
413.4%	930	3138	Secondary link
435.1%	33109	148242	Service
473.0%	322	4000	Living street
610.6%	2453	11124	Unpaved
1518.1%	371	900	Tertiary link
2169.0%	264	4168	Track
87407.7%	30	45	Road

Table 7.7: MAPE for each Category

Conclusion 8

In our work, we have proposed three methods that predict energy consumption for EVs given a route. The prediction of energy consumption for a route is performed on the individual segments that a route consists of. Because the prediction is performed on segments, our methods can also be used for energy-efficient routing. To the best of our knowledge, our work presents the first the methods which do not assume that the exact time a route arrives at a segment is given. In addition, the methods do not assume to know the exact driven speed. These are important distinctions as this information is not available for future trips.

We enrich the data-set of EV trips, with additional attributes. These attributes are added to achieve a higher performance for predictions. We implemented two general approaches of pairing elevation data with a road network. These approaches apply to any area with a road network and elevation data. We find that the slope of a segment has a linear correlation with the energy consumption. In addition to we have added information about direction change, traffic signals, and roundabouts. All of the attributes and information is constructed as features which are used in the methods. For our experiments we use the historical weather information, this information is not available in a real world setting. In a real world setting a weather forecast should be used for temperature. In addition we have not considered wind as a factor, despite it having an impact on energy consumption as shown by [Krogh et al., 2015].

The quality of the features which are constructed from the information we have contributed is determined using our LR and NN. We review the coefficients from our LR. The coefficients show that the information about traffic signals, roundabout, slope, and direction change has an impact. Using our NN, we tried different combinations of the features to determine the performance impact from the features. We find that including slope information improves the overall performance by ~1.3%, while traffic signals improve it by ~0.3%, and roundabouts ~1.6%. We have shown that using the actual time and speeds improves the performance by ~3%. As a result developing a speed model would provide better performance. We have shown that the higher granularity in slope, increases performance, as a result it is reasonable to consider approaches that include more information about elevation.

The combination of observations and models in NN-Observations improved the performance significantly. It is sensible to include observations into the predictions, as observations for a specific segment incorporates latent information. The method is still able to predict an energy consumption for segments without any observations.

We chose to predict the energy consumption on segments as opposed to routes because this enables the possibility of suggesting energy efficient routes. Additionally, the association between observations and routes would not longer capture the latent information that is specific for segments.

To extend our work, we consider adding confidence intervals for our predictions. Providing a confidence interval could, arguably, further reduce range anxiety. There has been some research in confidence intervals for Neural Networks. However, these confidence intervals are predictions [Zhang and Luh, 2005]. Instead of predicting a confidence interval, it could be found using a Bayesian approach.

There are many options for predicting energy consumptions for routes. Instead of predicting the energy consumption for segments, the prediction could be done for an entire route. However constructing a feature vector that properly includes information about segments is difficult as the number of segments that is traversed is variable for each trip.

Bibliography

Ove Andersen, Benjamin Bjerre Krogh, and Kristian Torp. Analyse af elbilers forbrug for perioden 2012-2013. Technical report, Institut for Datalogi, Aalborg Universitet, 2014a.

Ove Andersen, Benjamin Bjerre Krogh, and Kristian Torp. Analyse af elbilers forbrug. *Artikler fra Trafikdage på Aalborg Universitet*, 2014b. ISSN 1603-9696.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

Clever. Projekt test-en-elbil, 2017. URL `http://testenelbil.dk`. Visited 14-02-2017.

Zoran Constantinescu, Cristian Marinoiu, and Monica Vladoiu. Driving style analysis using data mining techniques. *International Journal of Computers Communications & Control*, 5(5): 654–663, 2010.

Cedric De Cauwer, Wouter Verbeke, Thierry Coosemans, Saphir Faid, and Joeri Van Mierlo. A data-driven method for energy consumption prediction and energy-efficient routing of electric vehicles in real-world conditions. *Energies*, 10(5):608, 2017.

Erhvervsstyrelsen. Plansystemdk, 2017. URL `https://erhvervsstyrelsen.dk/plansystemdk`.

Thomas Franke, Isabel Neumann, Franziska Bühler, Peter Cocron, and Josef F Krems. Experiencing range in an electric vehicle: Understanding psychological barriers. *Applied Psychology*, 61(3):368–391, 2012.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679 – 688, 2006. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2006.03.001. URL `http://www.sciencedirect.com/science/article/pii/S0169207006000239`.

Benjamin Krogh, Ove Andersen, and Kristian Torp. *Analyzing Electric Vehicle Energy Consumption Using Very Large Data Sets*, pages 471–487. Springer International Publishing, Cham, 2015. ISBN 978-3-319-18123-3. doi: 10.1007/978-3-319-18123-3_28. URL `http://dx.doi.org/10.1007/978-3-319-18123-3_28`.

Open Street Map. Osm tags for routing/maxspeed, 2017. URL `http://wiki.openstreetmap.org/wiki/OSM_tags_for_routing/Maxspeed`.

Michael A Nielsen. Neural networks and deep learning. *URL: http://neuralnetworksanddeeplearning. com/.(visited: 01.11. 2014)*, 2015.

National Oceanic and Atmospheric Administration. Noaa, 2017. URL `http://noaa.gov`. Visited 14-02-2017.

Peugeot. Peugeot ion specifikationer, 2017. URL `http://forhandler.peugeot.dk/www-spec/Personbil/iOn.pdf`. Visited 06-03-2017.

PostGIS. raster2pgsql, 2017. URL `http://postgis.net/docs/manual-2.2/using_raster_dataman.html`. Visited 27-03-2017.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL `http://arxiv.org/abs/1609.04747`.

Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:27, 1995.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Styrelsen for Dataforsyning og Effektivisering, 2017. URL `http://download.kortforsyningen.dk/`. Visited 14-02-2017.

U.S. Department of Energy. Fact #918: March 28, 2016 global plug-in light vehicle sales increased by about 80% in 2015 - dataset, 2016. URL `https://energy.gov/eere/vehicles/downloads/fact-918-march-28-2016-global-plug-light-vehicle-sales-increased-about-80`. Visited 16-02-2017.

Hai Yu, Finn Tseng, and Ryan McGee. Driving pattern identification for ev range estimation. In *Electric Vehicle Conference (IEVC), 2012 IEEE International*, pages 1–7. IEEE, 2012.

Li Zhang and Peter B Luh. Neural network-based market clearing price prediction and confidence interval estimation with an improved extended kalman filter method. *IEEE Transactions on Power Systems*, 20(1):59–66, 2005.

Bohan Zheng, Peter He, Lian Zhao, and Hongwei Li. A hybrid machine learning model for range estimation of electric vehicles. In *Global Communications Conference (GLOBECOM), 2016 IEEE*, pages 1–6. IEEE, 2016.

A Data Information

Category	Average Length	Standard Deviation	Variance
motorway	1080	1416	2005704
motorway_link	350	229	52467
trunk	462	753	566393
trunk_link	298	217	47298
primary	211	277	76571
primary_link	158	145	20989
secondary	212	280	78207
secondary_link	95	121	14556
tertiary	155	207	42839
tertiary_link	106	159	25312
unclassified	251	312	97238
residential	104	117	13805
living_street	78	67	4549
service	84	107	11517
road	287	259	67321
track	308	375	140853
unpaved	233	327	106855
All	176	292	84996

Table A.1: Length information about different categories.

Elevation Implementation B

[1] In this section we explain the data structure of raster and how elevation map is included into the data warehouse.

B.1 Raster Data Structure

We work with a geotiff raster data format, in our set each pixel is embedded with a georeference s.t. the location is known. In addition of the location, the altitude of each pixel is also embedded.

The data we have access is divided into squares that covers $1km^2$ of Denmark each. We use the command line tool `raster2pgsql`[PostGIS, 2017] to load these files into our data warehouse. There are many different flags which can be chosen for this tool it is important that the table has been created with the correct attributes, which can be done with the `-p (prepare)` flag. We run the tool with the following flags.

```
raster2pgsql -s 25832 -a -e -Y -t 250x250 *.tif experiments.elevation
```

- `-s 25832`: The SRID of our elevation data.
- `-a`: Append mode, since we have many different files, we run the command several times.
- `-e`: Execute each individual, s.t. we see the results in the database immediately.
- `-Y`: Copy mode, instead of table insert.
- `-t`: Tile size, we have chosen a size $250x250$ pixels.

When choosing the tile size, we consider the trade off between the number of rows that is required for a given size and the time required to look up in the tiles. We know that each pixel represents a $40cmx40cm$ square on a map. This means a $250x250$ tile will cover a $10000cmx10000cm$ area. With this information, and with the knowledge that Denmark covers $42\,925.46km^2$ we can estimate the number of rows we will have, which is an approximate 4 292 546 rows. After importing all the data into our date warehouse we have 4 959 800 rows of tiles.

[2]

Now we must construct an index on our data. For this purpose we use a gist index, on the convex hull of each tile. Figure B.1 shows an illustration of a convex hull on an exemplary tile. We construct an index on the convex hull s.t. only relevant information is indexed.

[1]FiXme Note: Se noter fra møde 3/4 thomas og kristian
[2]FiXme Note: Show tile map somewhere

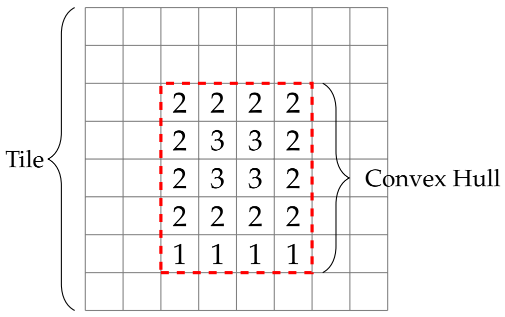


Figure B.1: This is an example of the Convex Hull for a tile. The dashed red line signifies the data in the Convex Hull, all pixels without information is not indexed.

B.2 Approach 1 Implementation

Approach 1 is a relatively simple implementation after the elevation map is loaded into the data warehouse. The goal is to include the elevation for each node in our road graph. Before we show how this is done it is important to mention that this is done edge wise. This means we will assign a height at the start and end of each edge, this will of course present of redundant data. We will ignore the redundant data as it lowers the number of joins we will later have to do.

We will construct a materialized view in our data warehouse which will hold this information. In Listing B.1, in this SQL Statement we make use of indexes on our raster data, specifically the function `ST_Intersects` makes use our gist index. If we omit the `LIMIT 800000`, the statement will not make use of our indexes.

Listing B.1: Approach 1 SQL Statement

```sql
CREATE MATERIALIZED VIEW experiments.approach1 AS
SELECT starth.segmentkey, starth.startheight, endh.endheight
FROM
    /** Find start height**/
    (SELECT road.segmentkey,
        ST_Value(rast, road.start, FALSE) as startheight
    FROM
    experiments.elevation2 ele,
    (SELECT segmentkey,
        ST_Startpoint(ST_Transform(segmentgeo::geometry, 25832)) as start
        FROM maps.osm_dk_20140101
        LIMIT 800000) road
    WHERE ST_Intersects(ele.rast, road.start)
    ) starth,
    /** Find end height**/
    (SELECT road.segmentkey,
       ST_Value(rast, road.endheight, FALSE) as endheight
    FROM
    experiments.elevation2 ele,
    (SELECT segmentkey,
        ST_Endpoint(ST_Transform(segmentgeo::geometry, 25832)) as endheight
        FROM maps.osm_dk_20140101
        LIMIT 800000) road
    WHERE ST_Intersects(ele.rast, road.endheight)) endh
WHERE endh.segmentkey=starth.segmentkey;
```

Once this materialized view has been created, the process of calculating a slope is straight forward, since we have 2 points.

$$slope = \frac{y_2 - y_1}{x_2 - x_1} \tag{B.1}$$

In our case the height is our y axis and the x is the distance between the start and end point. We can reformulate the slope equation to our case. We say that $x_2 - x_1$ is simply the length of the road, given in the data warehouse.

For convenience we construct a view that calculates a slope from our `experiments.approach1` table, which is show in Listing B.2

Listing B.2: Approach 1 Slope SQL Statement

```sql
CREATE VIEW experiments.approach1-slope AS
SELECT map.segmentkey,
    CASE WHEN meters != 0 THEN
        (endheight-startheight)/meters
    ELSE NULL
    END as slope
FROM experiments.approach1 app1,
    maps.osm_dk_20140101 map
WHERE app1.segmentkey=map.segmentkey;
```

Since the trips are map matched at the same segments, no map matching is required to this approach.

B.3 Approach 2 Implementation

Unlike Approach 1, Approach 2 requires map matching as we are going to alter the map provided by OSM. We will begin by subdividing all the edges in our road graph s.t. no edge has a length longer than $50m$, e.g. if an edge is $130m$ it will divided into 3 edges of length $50m$, $50m$, and $30m$. We do this by constructing a function in our database. We call the function `split_linestring`, it takes two parameters namely `split_length` which specifies the maximum length of subdivided edges, while `geo` is the edge that going to be subdivided. We make use of `ST_LineSubstring` where we can specify which part of an edge we want returned. It's done by given the start and end fraction. Where a fraction is a number between 0 and 1, 0 is the start of the edge and 1 is the end. E.g. given a $100m$ edge we can split it by using `ST_LineSubstring` twice. First using start 0.0 and end 0.5, and second start 0.5 and end 1.0. Since we are working with floats the edges wont be precisely $50m$ but the representation is close enough for our usage.

Listing B.3: Approach 2 Splitter

```sql
CREATE OR REPLACE FUNCTION split_linestring(split_length DECIMAL, geo
    geometry) RETURNS setof subsegmentkey_geometry AS $$
DECLARE
i integer := 1;
meters DECIMAL := st_length(geo::geography)::DECIMAL;
x subsegmentkey_geometry;
BEGIN
 IF meters != 0.0 THEN
   WHILE ((split_length/meters)*(i)) <= 1.0 OR ((split_length/meters)*(i-1)
       ) <= 1.0 LOOP
     IF ((split_length/meters)*(i)) <= 1.0  THEN
       x.subsegmentkey := i;
       x.geom := ST_LineSubstring(geo::geometry, (split_length/meters)*(i
           -1), (split_length/meters)*i);
       RETURN NEXT x::subsegmentkey_geometry;
       i := i + 1;
     ELSE
       x.subsegmentkey := i;
       x.geom := ST_LineSubstring(geo::geometry, (split_length/meters)*(i
           -1), 1.0);
       RETURN NEXT x::subsegmentkey_geometry;
       i := i + 1;
     END IF;
   END LOOP;
 END IF;
 RETURN;
END;
 $$ LANGUAGE 'plpgsql';
```

We run this function on all our edges, s.t. all edges are subdivided, we save this information. The next step is to add an elevation to the start and end of all new sud divided edges. This is done much like in Listing B.1, the main difference being that the `LIMIT 800000` is increased to 2 000 000 s.t. indexes are used.

As mentioned we must also map match observations to these new sub divided edges. We have constructed our sub-edges s.t. they have the original edge id along with an id that makes them individual. As such we can find all observations that belong to the original edge, which reduces the number of observations that must be considered for each sub-edge. For this purpose we write a function called `subsegment_mapmatch` which takes a single segment id (edge id), shown in Listing B.4.

Listing B.4: Approach 2 Map matching

```
CREATE FUNCTION subsegment_mapmatch(segment_key integer) returns void
   language plpgsql
as $$
DECLARE
  r RECORD;
  i INT = 0;
  a BIGINT[];
BEGIN
    FOR r IN SELECT array_agg(id) as sarray, subsegmentkey FROM(
    SELECT DISTINCT ON(id) st_distance(sq.coordinate, sq.subgeo) as mindist
        , id, sq.subsegmentkey, sq.segmentkey
    FROM (
      SELECT  coordinate, subgeo, app2.segmentkey, subsegmentkey, id
      FROM (
        SELECT gps.coordinate, segmentkey, gps.id
        FROM (
          SELECT unnest(gpsdata_ids) as id, segmentkey
          FROM experiments.point_match
          WHERE segmentkey=segment_key
          AND cardinality(gpsdata_ids)>0) id,
          experiments.factgpsdata gps
          WHERE id.id=gps.id) coor,
      experiments.approach2 app2
      WHERE coor.segmentkey=segment_key
      AND app2.segmentkey=segment_key
      AND st_dwithin(coor.coordinate,app2.subgeo, 50)) sq
    ORDER BY id, mindist)ssq
    GROUP BY subsegmentkey LOOP
        UPDATE experiments.approach2 SET gpsdata_ids = r.sarray
        WHERE segmentkey=segment_key
        AND subsegmentkey=r.subsegmentkey;
        i = i + 1;
    END LOOP;
END;
$$;
```

B.4 Derived Features SQL

Listing B.5: Direction Change Feature SQL

```sql
WITH current_next AS (
  SELECT current.id,
    current.segmentkey as currentkey, next.segmentkey as nextkey,
    current.direction as current_dir, next.direction as next_dir
  FROM experiments.viterbi current, experiments.viterbi next
  WHERE current.trip_id=next.trip_id
  AND next.trip_segmentno=current.trip_segmentno+1),
map_current_next AS(
  SELECT id,
    mapcurrent.segmentgeo::geometry as currentgeo,
    mapnext.segmentgeo::geometry as nextgeo,
    current_next.current_dir, current_next.next_dir
  FROM current_next, maps.osm_dk_20140101 mapcurrent,
  maps.osm_dk_20140101 mapnext
  WHERE mapcurrent.segmentkey=current_next.currentkey
  AND  mapnext.segmentkey=current_next.nextkey)
SELECT id,
  CASE WHEN current_dir = 'FORWARD' THEN
  degrees(st_azimuth(
    st_pointn(currentgeo,ST_NPoints(currentgeo)-1),
    st_pointn(currentgeo,ST_NPoints(currentgeo))))
  WHEN current_dir = 'BACKWARD' THEN
  degrees(st_azimuth(st_pointn(currentgeo,2),st_pointn(currentgeo,1)))
  END as currentdeg,
  CASE WHEN next_dir = 'FORWARD' THEN
  degrees(st_azimuth(st_pointn(nextgeo,1),st_pointn(nextgeo,2)))
  WHEN next_dir = 'BACKWARD' THEN
  degrees(st_azimuth(st_pointn(nextgeo,ST_NPoints(nextgeo)),
    st_pointn(nextgeo,ST_NPoints(nextgeo)-1)))
  END as nextdeg
INTO experiments.trip_degree
FROM map_current_next;
```

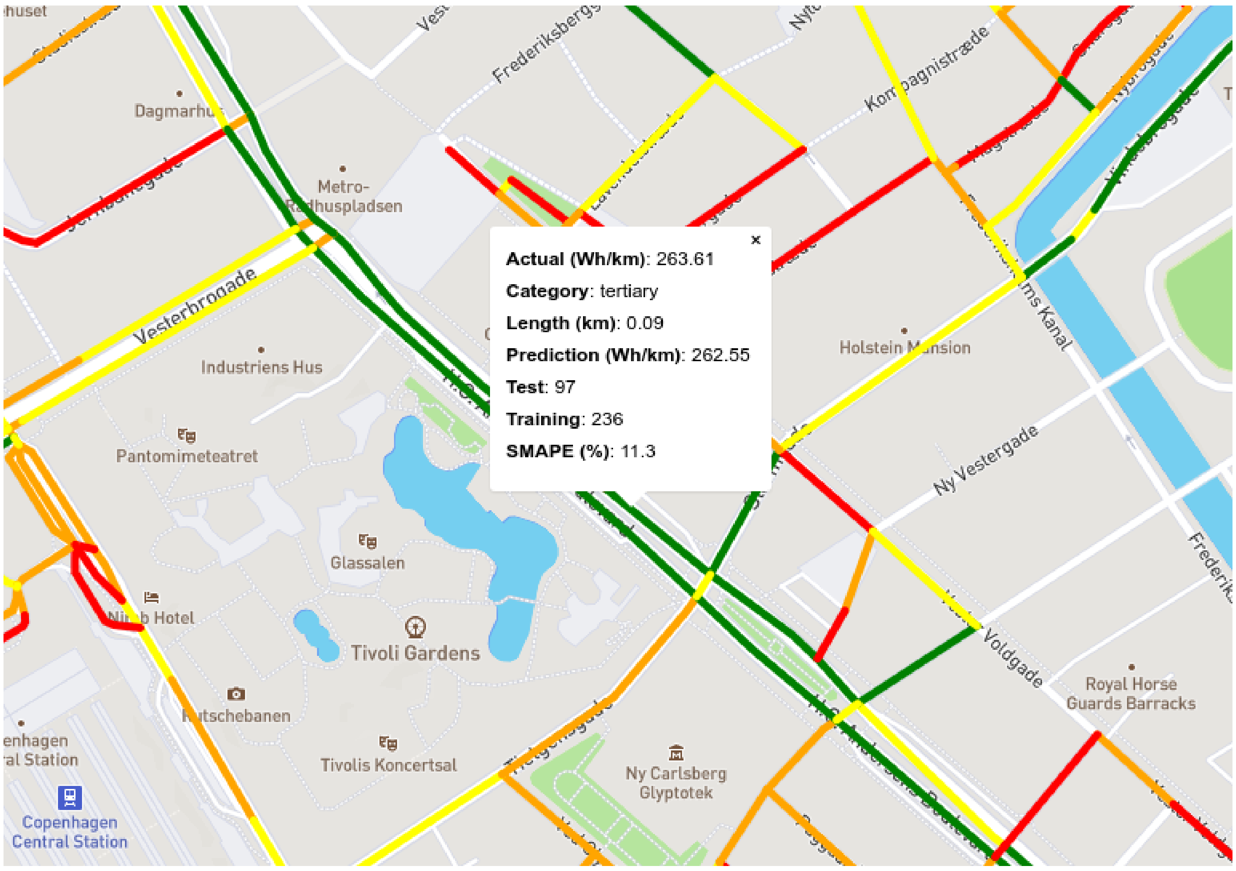
B.5 Zone SQL

Listing B.6: Zone mapping for segments

```
create function segment_to_gid(segment integer) returns void
LANGUAGE plpgsql
AS $$
DECLARE
  arow record;
BEGIN
  FOR arow IN
    SELECT segmentkey, gid, ST_LENGTH(ST_Intersection(zone.geop, map.
        segmentgeo)) as len
    FROM experiments.zone zone, maps.osm_dk_20140101 map
    WHERE st_intersects(zone.geop, map.segmentgeo)
    AND st_isvalid(zone.geom)
    AND segmentkey=segment
    ORDER BY len DESC
    LIMIT 1
  LOOP
    INSERT INTO experiments.zonemedgist VALUES (arow.segmentkey, arow.gid);
    RAISE NOTICE 'Segmentkey % Most in gid %', arow.segmentkey, arow.gid;
  END LOOP;
END;
$$;
```

Interactive Map C

Our result from the experiment NN-observations is visualized on a map at `http://energi.elefsennet.dk`. The map shows the results from the NN-observations on our test set, that means only segments from the test set are colored. We have divided the segments into 4 colors based on the sMAPE for each segment. Description of the colors can be seen on the map. As seen in Fig. C.1 the map provides detailed information about the individual segments in a pop-up. This pop-up appears by clicking on a segment. The **Training** and **Test** attributes tells how many samples are in each set. The **Actual** and **Prediction** is an average energy consumption for the segment.



Actual (Wh/km): 263.61
Category: tertiary
Length (km): 0.09
Prediction (Wh/km): 262.55
Test: 97
Training: 236
SMAPE (%): 11.3

Figure C.1: Pop-up for a Segment