Predicting power consumption of shared electric vehicles using regression and cluster analysis

Lage Bergman
Johannes Hildén

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

Master's thesis 2019

Predicting power consumption of shared electric vehicles using regression and cluster analysis

Lage Bergman
Johannes Hildén

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2019

Predicting power consumption of shared electric vehicles using regression and cluster analysis
Lage Bergman
Johannes Hildén

Supervisor: Birgit Grohe, Department of Computer Science and Engineering
Advisor: Viktor Dahl, Miveo Car-sharing Technologies AB
Examiner: Graham Kemp, Department of Computer Science and Engineering

Master's Thesis 2019
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2019

Predicting power consumption of shared electric vehicles using regression and cluster analysis
Lage Bergman
Johannes Hildén
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

When planing a trip with an electric vehicle, one has to predict how much power will be consumed in advance in order not to risk running out of power during the trip. This problem is especially relevant for car-sharing services, as they want to schedule multiple trips in close succession. Current methods for estimating power consumption all rely on information regarding the exact route that will be followed, which in a car-sharing service is not always known beforehand.

This thesis presents prediction models based on historical trip information capable of predicting power consumption for planned trips with electric vehicles without information about the specific route. The models are created by combining linear regression and unsupervised learning, in particular clustering with K-means. K-means is used to cluster users into groups based on previous driving habits. Separate regression models are then trained for each group in order to make more accurate predictions. We show that it is possible to predict power consumption using stand alone regression models and a few simple features such as booking duration and estimated distance. We also show that these predictions can be improved by the use of clustering. While these results are satisfactory on their own, we conclude that improvements can be made to the way distances are estimated in order to further improve predictions.

Keywords: Regression, Clustering, Cluster based regression, Electric vehicles, Predictions, Power consumption, Car-sharing.

Acknowledgements

First and foremost we would like to thank our supervisor Birgit Grohe. Without your support and feedback, we would not have been able to complete the project.

We would also like to thank Miveo Car-sharing Technologies AB for making the project possible. Especially, we would like to thank our company supervisor, Viktor Dahl, for his technical expertise and the company CEO, Erik Nordenfelt, for providing us with office space and always making us feel welcome.

Lage Bergman, Gothenburg, June 10, 2019
Johannes Hildén, Gothenburg, June 10, 2019

Contents

List of Figures

List of Tables

1 Introduction

Drivers of electric vehicles (EVs) face an interesting problem, how do you make sure you do not need to stop to recharge your car in the middle of a trip? There is currently no way of charging an electric car as fast as refueling a normal gas powered vehicle, so stopping and recharging during a trip might not always be an option. The problem becomes even more complicated for companies offering electric car-sharing services. To maximize revenue, car-sharing companies want to minimize the downtime when vehicles are not in use, and schedule bookings as close to each other as possible. However, in order to schedule multiple trips with the same vehicle in close succession, the companies have to try to predict the amount of power that is going to be consumed in advance of every trip. If these predictions are not accurate enough a vehicle might not have time to recharge after a booking, running the risk of following bookings starting with insufficient power. This in turn could lead to a risk of drivers getting stranded in the middle of trips which would make the service unreliable and unattractive.

1.1 Background

This project is conducted in collaboration with Miveo Car-sharing Technologies AB[1], a car-sharing company specialized in electric vehicles. Their current method for estimating power consumption heavily relies on the input of their users, where each user has to input the distance of a planned trip when placing a booking. The distance is then multiplied with the assigned vehicle's consumption per kilometer in order to get a rough estimate for power consumption. These estimations are not very precise due to the simplicity of their prediction model combined with the fact that users tend to overestimate their planned driving distance. Because of this, a lot of manual work is put into reassigning vehicles to bookings. In the absolute worst case, external taxi services are used as a replacement for bookings to prevent the driver from running out of power during a trip. This is very costly and will not scale well with company growth as more vehicles are connected to the service. A better, more scalable solution is to implement a more accurate prediction model.

Every vehicle connected to Miveo's platform is equipped with a large number of

[1] https://www.miveo.se

sensors. All the information gathered from these sensors is stored together with historical booking information in a database. Over time, enough data has been collected to allow for statistical analysis to be conducted on the data. Using such analysis, it would be possible to build models for making predictions for future bookings. Making one single model based on the historical data from all users is unlikely to yield accurate predictions since different users use the service for very different kinds of trips. Building one model per user, on the other hand, would probably give accurate predictions but would require more data from every user. A possible compromise is grouping similar users together and building one model per user group. Miveo has been able to identify users that have an unusually high power consumption when driving compared to others. This indicates that clustering techniques could be used to group users together to make better predictions.

1.2 Problem description

The goal of this project is to create prediction models that predict the power consumption for planned trips with electric vehicles. In the context of this project, a planned trip consists of a driver, a vehicle, an estimated distance, start and end time, as well as a location for pick up and drop off. A trip does not contain the specific route or the number of starts and stops that will be made. Having no information on the specific route means a trip does not contain slope, curvature or speed limits, only the estimated total distance. In order to make predictions for such trips, we propose a statistical model based on the hypothesis that there are patterns in the kinds of trips users make and that users taking similar trips can be grouped together. Examples of patterns we expect to find are commuters using freeways or drivers making shorter trips with multiple stops in urban areas, patterns that are proven to have a high impact on power consumption (Wu, Freese, Cabrera, & Kitch, 2015). By combining a user's driving pattern with the information given at the time of a booking, we expect to be able to make predictions surpassing the accuracy of Miveo's current solution. In particular, we want to determine whether this type of model is capable of achieving a lower loss when compared to the loss of the method currently used by Miveo. To achieve this we propose the following research questions:

- Is it possible to create a regression model capable of predicting the energy consumption of a planned trip with an EV without knowing which route the driver will follow?
- Can multiple models targeting different categories of drivers achieve predictions with a lower average loss when compared to a single model?

In order to answer these questions, we will start by creating a number of regression models. We will then use clustering techniques to group users into categories and create multiple new regression models based on each found category. Finally, all models will be compared to see which are more successful in predicting the power consumption of a planned trip and whether the prediction models benefit from the use of clustering. For all of our predictions, we will only consider trips that start and end on the same day, as drivers taking longer trips are expected to return their vehicle in a reasonably charged state. This means that drivers taking such longer trips do not pose the same risk of handing over a vehicle with insufficient power to the next driver.

1.3 Related work

Previous research has presented more complex, both statistical models and models based on physical properties, capable of making high accuracy predictions of power consumption in EVs. One approach based on physical properties was introduced by (Wu et al., 2015). Their equation uses the physical properties of a vehicle such as its weight and aerodynamic resistance coefficient together with its speed, acceleration and road grade to estimate the EVs instantaneous power. This makes it possible to integrate the instantaneous power over trip time in order to estimate the total power consumption of a trip. Their approach is useful for optimizing velocity when driving along a known route to minimize total power consumption. In order to use it for predicting power consumption of future trips however, one would need to know the exact route and driving velocity beforehand.

Another method using support vector regression for predicting the future power consumption of a trip has been suggested by (Grubwinkler & Lienkamp, 2015). In their approach, a separate model is created for each driver in order to make more accurate predictions. The models are based on the speed profiles of all users, how the specific user deviates from the mean of all users, as well as contextual information such as weather and time of day. Using this model, it is possible to predict the energy consumption of a trip that a specific driver makes along a given route under specified conditions.

Common to these studies is that they predict power consumption for a predetermined route. Our situation is quite different since we do not have access to any route information. It is also more difficult due to the route's high impact on power consumption (Wang, Besselink, & Nijmeijer, 2018).

There are previous studies successfully combining clustering and regression in order to make more accurate predictions. An example where K-means and Affinity Propagation are used for categorizing road incidents in urban areas has been presented by (Ghosh et al., 2016). Their method includes training separate prediction models for each incident category to predict the duration of the disturbance caused by incidents in that category. In their study, the performance of various standalone regression models is compared to models combining clustering and regression. The predictions were made using a number of regression techniques for two distinct data sets. In both cases, the combination of clustering and regression yields significantly better prediction results than any of the standalone regression models.

Another way of combining clustering and regression is through a technique called clusterwise regression, first introduced by (Späth, 1979). The goal of the technique is simultaneous grouping of data points and optimization of regression coefficients for each group. The original research introduces an exchange algorithm for finding the best clusters and coefficients in order to minimize the sum of squares for all clusters. Since then, several studies have been conducted with the goal of implementing new ways of performing such optimization. One way suggested by (DeSarbo & Cron, 1988) includes a conditional mixture, maximum likelihood methodology, whereas another by (Park, Jiang, Klabjan, & Williams, 2017) uses a mathematical programming based approach relying on column generation. One study where a practical application was found for clusterwise regression was conducted by (Luo & Chou, 2006). They successfully use clusterwise regression to group pavements with similar characteristics in order to predict the expected time of required maintenance.

1.4 Ethical considerations of this work

Whenever a method involves classification of human beings, there are ethical aspects to consider. Even when working with anonymous data, one needs to be careful in order to make sure the classification does not pick up on underlying structures and make decisions based on unintended attributes of the people it is classifying. In the case of this project, there are limits as to how the outcome will impact people, but it is still possible to imagine scenarios where people are treated differently because of how they are classified. For example a person who is classified as someone who has a generally low energy consumption is more likely to receive vehicles with lower battery level. This means that the risk of such a person running out of power during an ongoing trip is larger than for someone who is considered less efficient when driving. This means that people who have a more defensive driving style might be penalized for it.

1.5 Thesis outline

The remaining chapters of this thesis are Theory, Results and Discussion. Theory describes the theories behind the methods used in the thesis project. Results contains descriptions of the methods used, as well as the results achieved through the application of each method. Finally, the Discussion chapter presents and discusses conclusions to be drawn from the results, and elaborates on ways to further investigate the thesis topic.

2

Theory

This chapter describes the theories used when developing and evaluating the models for predicting energy consumption. Section 2.1 describes metrics used, Section 2.2 describes regression techniques and Section 2.3 describes clustering techniques.

2.1 Metrics

2.1.1 Pearson's R Correaltion Coefficient

The Pearson's R correlation coefficient describes the strength of the linear relationship between two variables. This means that the Pearson's R coefficient of two variables can be used as an indicator to whether one of the variables works as an independent variable for predicting the other in a regression model. The coefficient is calculated by dividing the covariance of the two variables by the product of their standard deviations:

$$\rho(a, b) = \frac{cov(a, b)}{\sigma_a \sigma_b} \tag{2.1}$$

where a and b are the variables of which the relationship is measured, σ_a and σ_b are their standard deviations, and $cov(a, b)$ is the covariance between them. The covariance between two variables is positive if there is a positive correlation between them and negative if there is a negative correlation. The magnitude of the covariance is not easy to interpret, however, as it has the unit of the two variables multiplied, and can therefore be arbitrarily big. By dividing the covariance by the product of their standard deviations, we get a normalized value. This results in the metric being fixed in the interval $\rho \in [-1, 1]$ where -1 represents a completely linear negative correlation, 1 represents a completely linear positive correlation and 0 represents no correlation. The metric is symmetric for a and b (Baak, Koopman, & Klous, 2018).

2.1.2 R^2

The R^2 statistic describes how much of the variation in the dependent variable y around the mean $\bar{y}$ that is explained by the independent variables (Draper & Smith, 1998a; Sen & Srivastava, 2012a). The statistic is calculated by:

$$SS_{tot} = \sum_i (y_i - \bar{y})^2 \tag{2.2}$$

$$SS_{res} = \sum_i (y_i - \hat{y}_i)^2 \tag{2.3}$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{2.4}$$

where y_i is an observed value of y, and $\hat{y}_i$ is the corresponding predicted value of y. Normally, the metric takes values in the interval $[0, 1]$, where the value corresponds to the percentage of variation in the dependent variable that is explained by the independent variables. If the residual sum of squares SS_{res} is larger than the total sum of squares SS_{tot} however, lower values can occur, and the metric can then take infinitely small values. R^2 can be used as a complement to loss when measuring the fit of a regression model. The benefit of using R^2 is that it can be easier to interpret compared to a raw loss value (Sen & Srivastava, 2012a).

2.2 Linear regression

Linear regression is a tool for modelling the linear relationship between a dependent variable y and one or multiple independent variables $(x_1, ..., x_n)$ (Sen & Srivastava, 2012b). Such a model can then be used to make predictions for new y values given sets of x values which in this study correlates to predicting power consumption from various booking features. Linear regression works by finding the best fitting hyperplane through some training data and then uses this hyperplane to make predictions for new unseen data points. More specifically it finds the best fitting hyperplane:

$$y = \hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_n x_n \tag{2.5}$$

through the n-dimensional points $(x_1, ..., x_n)$ by finding the regression coefficients $(\hat{\beta}_0, ..., \hat{\beta}_n)$ that minimize a specified loss function. There are many different loss functions to choose from and they all include the so called residual term, the difference between predicted and actual dependent variable values for the training data. This guarantees that minimizing the loss function also produces a model that minimizes the residuals for the training data. Our different choices of loss functions and their uses can be found in Section 2.2.3.

Besides the choice of different loss functions, there are also different optimization methods for finding the optimal regression coefficients (Montgomery, Peck, & Vining, 2012). The two optimization methods we will consider are ordinary least squares and stochastic gradient descent.

2.2.1 Ordinary least squares

Ordinary least squares (OLS) is a method that finds the regression coefficients (see Equation 2.5) by using simple calculus and enforcing the use of the squared loss function (Sen & Srivastava, 2012c). The convex nature of squared loss means that there is only a single minimum which can be found analytically using derivatives. To show this more formally we start with the formula for the sum of squared loss:

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 \tag{2.6}$$

where M is the size of the training data set, $\hat{y}_i$ the predicted value for the individual training example i and y_i the observed value of that training example. The reason for summing the squares of all residuals is to find the coefficients that minimize the total squared error over all training data points. If we replace $\hat{y}$ with the full prediction function we get:

$$\sum_{i=1}^{M}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + ... + \hat{\beta}_n x_{in}))^2 \tag{2.7}$$

To find the β values that minimize the loss function we simply derive the loss function with respect to the different $\hat{\beta}$ and set the derivatives to 0:

$$\begin{aligned} \frac{\delta E}{\delta \hat{\beta}_0} &= \sum_{i=1}^{M} -2(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - ... - \hat{\beta}_n x_{in}) = 0 \\ &\vdots \\ \frac{\delta E}{\delta \hat{\beta}_n} &= \sum_{i=1}^{M} -2x_{in}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - ... - \hat{\beta}_n x_{in}) = 0 \end{aligned} \tag{2.8}$$

and solve for the individual $\hat{\beta}$ (Draper & Smith, 1998b). The advantage of using OLS is that it finds the optimal coefficients analytically which means it always find the single global optimum. This makes it a very attractive option for relatively simple scenarios with few features. However, if the number of features is large, finding the optimum can be very costly. If the number of features exceeds the number of training data points, finding a unique global optimum is not even possible as there are more equations than variables and such an underdetermined system has infinitely many solutions. The method also performs badly if the data set is noisy, as large errors squared can have great impact on the model fit, making predictions unreliable (Guo, Hu, & Shi, 2018).

2.2.2 Stochastic gradient descent

Stochastic gradient descent (SGD) is an alternative to OLS that can handle many different loss functions, even those that are non-convex. Instead of finding the optimal regression coefficients for the entire training set at once it iterates the training set randomly, considering one example at a time. Once again using squared loss, the formula for each individual example's loss function becomes:

$$(y - (\hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_n x_n))^2 \tag{2.9}$$

For each example it then just like OLS calculates the derivative of the loss function with respect to the different β:

$$\begin{gathered}\frac{\delta E}{\delta \hat{\beta}_0} = -2(y - \hat{\beta}_0 - \hat{\beta}_1 x_1 - ... - \hat{\beta}_n x_n) = 0 \\ \vdots \\ \frac{\delta E}{\delta \hat{\beta}_n} = -2x_n(y - \hat{\beta}_0 - \hat{\beta}_1 x_1 - ... - \hat{\beta}_n x_n) = 0\end{gathered} \tag{2.10}$$

However instead of finding the points where the derivatives equal zero, it finds the direction each coefficient should move, in order to reach a lower loss value for that individual example. So by inserting the known $(x_1, ..., x_n)$ and y values from random training example r into our derivatives we get the values $(\Delta\beta_1, ..., \Delta\beta_n)$ for which our β coefficients should be updated with:

$$\begin{gathered}\Delta\beta_1 = -2(y - \hat{\beta}_0 - \hat{\beta}_1 x_{r1} - ... - \hat{\beta}_n x_{rn}) \\ \vdots \\ \Delta\beta_n = -2x_n(y - \hat{\beta}_0 - \hat{\beta}_1 x_{r1} - ... - \hat{\beta}_n x_{rn})\end{gathered} \tag{2.11}$$

To reduce the impact every individual training example has on the coefficients these $\Delta\beta$ are then multiplied with a small value η (often called learning rate) before being added to the previous coefficients:

$$\begin{gathered}\beta_1 = \beta_{1_{prev}} + \eta * \Delta\beta_1 \\ \vdots \\ \beta_n = \beta_{n_{prev}} + \eta * \Delta\beta_n\end{gathered} \tag{2.12}$$

This iterative process allows the regression coefficients to slowly converge to values corresponding to a local, and in the case of a convex loss function, also global minimum of the loss function. The advantages of using SGD as opposed to OLS is that the technique is not bound to any particular loss function. It also allows for the inclusion of regularizer terms which can help in many respects that are discussed further in Section 2.2.4 (Theodoridis, 2015).

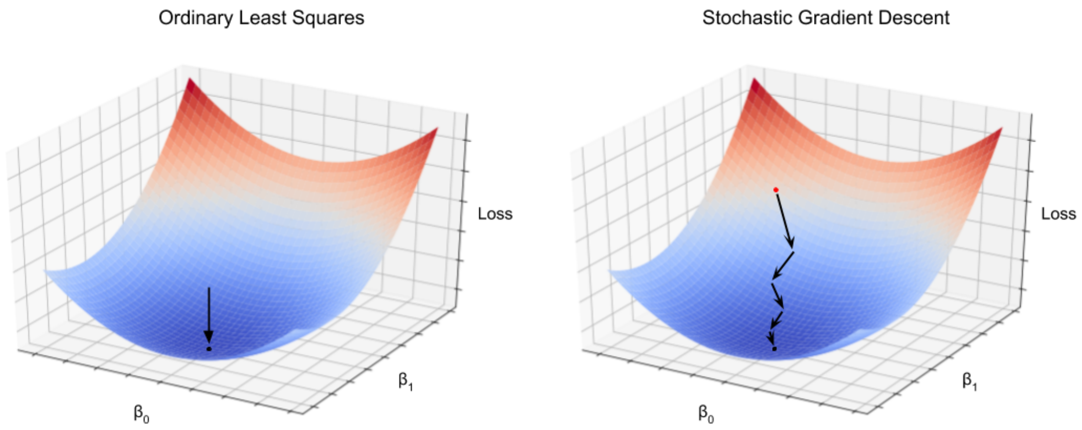

Figure 2.1: Plots showing the optimization process of OLS and SGD respectively for two dimensions of regression coefficients. OLS finds the minima immediatly while SGD takes small incremental steps towards the minima.

2.2.3 Loss functions

Loss functions are the group of functions that are minimized during training in order to make predictions. Different loss functions can produce different prediction models as they penalize residuals in different ways. The two loss functions considered here are Squared loss and Huber loss.

2.2.3.1 Squared loss

Squared loss is a very commonly used loss function which simply returns the squared difference between the actual and predicted value. As previously mentioned in Section 2.2.1, it has the formula:

$$L(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2 \tag{2.13}$$

with $\hat{y}_i$ and y_i being the predicted and actual values for a single observation y_i respectively. The squaring property takes care of possible negative differences and guarantees that the model has 0 loss only if the model perfectly fits all points. An additional property of squared loss is that it penalizes large residuals more than it penalizes small ones making models using squared loss more susceptible to outliers (Guo et al., 2018).

2.2.3.2 Huber loss

Huber loss treats residuals differently depending on their size. Residuals smaller than or equal to a parameter ϵ are squared while residuals larger than ϵ have a value proportional to their absolute value returned. The loss function is described by the following expression:

$$L(y_i, \hat{y}_i) = \begin{cases} (y_i - \hat{y}_i)^2 & \text{for } |y_i - \hat{y}_i| \leq \epsilon \\ 2\epsilon|y_i - \hat{y}_i| - \epsilon^2 & \text{otherwise} \end{cases} \tag{2.14}$$

Using Huber loss makes a model more robust in the sense that it becomes more resistant to outliers, while remaining as penalizing as mean squared error for normal residuals (Hastie, Tibshirani, & Friedman, 2009).

2.2.4 Regularization

A common problem when creating prediction models is making the models overly complex by taking too many variables into account. Such overly complex models risk becoming too closely correlated with the training data. This phenomenon is commonly referred to as overfitting. Regularization is one of the techniques used to combat the effects of overfitting and works by adding an additional term to the loss function. For this project, we consider two different regularization techniques, Ridge and Lasso (Zou & Hastie, 2005). The term added to the loss function in Ridge regression is the sum of all regression coefficients squared:

$$\sum_{i=0}^{n} \beta_i^2 \tag{2.15}$$

The term added in Lasso is similar, but instead of squaring each coefficient it takes the absolute value:

$$\sum_{i=0}^{n} |\beta_i| \tag{2.16}$$

Both of these terms penalize large coefficients which result in a reduced variance and a slight increase in bias. The difference between the two techniques is that while both reduce the size of the coefficients, Lasso is able to reduce coefficients to 0, making it capable of completely ignoring unimportant features (Zou & Hastie, 2005).

2.3 Clustering

2.3.1 K-means

K-means is a method for assigning data points to clusters based on their attributes. The algorithm aims to minimize the summed distance between each data point and the mean of the cluster that point is assigned to. Formally this is defined as:

$$\min \sum_{k=1}^{K} \sum_{x_i \in c_k} ||x_i - \mu_k||^2 \tag{2.17}$$

where K is the specified number of clusters, μ_k is the mean of cluster k, and $x_i \in c_k$ are all points assigned to cluster k. It minimizes this summed distance by iteratively reassigning points to the closest cluster in order to gradually move the means of the clusters. For this to be possible, the number of clusters has to be specified in advance. By minimizing the summed distance, the algorithm assigns each point to the cluster closest to that point (Jain, 2010).

There are two main drawbacks to the K-means algorithm. The first is that it is greedy and is therefore not guaranteed to converge to a global minimum. The second is the fact that the optimal K cannot be found analytically, and so the number of clusters has to be specified in advance. The most common method for dealing with these drawbacks is to run the algorithm several times while varying the initial cluster positions as well as the value of K. The best clustering is then selected based on domain knowledge.

2.3.2 DBSCAN

The density-based spatial clustering of applications with noise, DBSCAN, algorithm is a clustering algorithm which selects clusters by identifying groups of points with high density. The algorithm relies on two parameters in order to identify these groups, Eps and MinPts. Eps is the maximum allowed distance a point q can have to another point p for q to be part of the Eps-neighbourhood of p. The Eps-neighbourhood of a point p is defined as:

$$N_{Eps}(p) = \{q \in D | dist(q,p) \leq Eps\} \tag{2.18}$$

where D is the full set of all points. In the DBSCAN algorithm, all points are labeled as one of three categories, core points, edge points or noise. MinPts describes the minimum size of a point's Eps-neighbourhood for the point to be considered a core point:

$$|N_{Eps}(p) \geq MinPts| \tag{2.19}$$

Points which are part of the Eps-neighbourhood of a core point but that are not core point themselves are labeled as edge points. A cluster consists of a number of neighbouring core points as well as all of their surrounding edge points. Points that are neither core or edge points are considered noise. This technique has two main advantages, the number of clusters does not need to be specified in advance and the clusters can have any shape, concave or convex (Ester, Kriegel, Sander, & Xu, 1996). However, selecting the optimal value for each of the two parameters Eps and $MinPts$ is not trivial. There are basic guidelines for selecting good parameters, but domain knowledge and some trial and error is still required (Schubert, Sander, Ester, Kriegel, & Xu, 2017).

3

Results

The project is structured into five main phases. The phases are data preprocessing, data analysis, regression analysis, cluster based regression and finally evaluation. More in-depth explanations and results of the different phases can be found in the sections below. Section 3.1 goes into detail on how the data is quality controlled and how features are extracted. Section 3.2 describes how the preprocessed data is analysed to identify what factors influence power consumption. Section 3.3 presents stand alone regression models and section 3.4 presents regressions models combined with clustering. Finally, section 3.5 presents the final evaluation of our best performing models using the test set.

3.1 Data preprocessing

The very first step of the project is to retrieve and quality control the data that will later be used for regression and clustering. Due to the different ways the vehicles communicate with the back-end servers at Miveo we receive three different data files with assembled information regarding these bookings. We refer to these three files as the booking contents file, the heartbeat file and the trip segments file. The booking contents file contains a list of the bookings, with information such as the vehicle id, the user id and when the booking took place. The heartbeat file includes information gathered from periodical heartbeat signals that the vehicles send out. This with information such as position, current battery level, etc. Finally, the trip segments file includes information gathered from a second signal that is sent by each vehicle whenever it starts or stops. Tables containing all relevant contents of the data files can be found in Appendices A, B and C.

3.1.1 Quality control of the data set

The initial data set consists of roughly 33000 electric vehicle bookings made between the years of 2016 and 2019. From these 33000 bookings we remove all overnight bookings. As mentioned in Section 1.2, bookings that span over several days have separate requirements compared to regular single day bookings. Vehicles that are booked overnight are expected to be returned in a reasonably charged state and are therefore not as important to make predictions for. By removing overnight bookings we get a smaller data set of more similar bookings that are easier to make predictions for. We also remove bookings made by users with less than three bookings in the data set. This guarantees that each driver category found during clustering will have validation and test data after we split our data into a training, validation and test set (see Section 3.1.4).

Due to the nature of how some of the data is recorded in the vehicles, it is not uncommon for errors to occur. These could be errors such as specific values changing too quickly or suddenly showing 0 without reason. A one hour excerpt from a booking displaying impossible value changes in battery level can be found in Figure 3.1. It is simply not possible for a vehicle to gain or lose as much as 60% of its battery capacity in just a few minutes during normal use. These sorts of fluctuations suggest that there is something wrong with the vehicle or its measuring equipment.

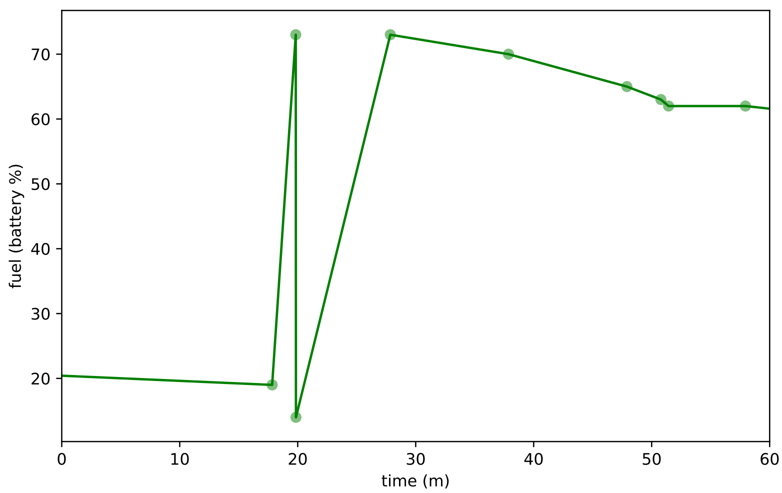

Figure 3.1: A one hour excerpt from a booking displaying impossible value changes in battery level.

In order to automatically find these sort of fluctuations we extract the mean velocity and maximum fuel difference per second for each booking. The mean velocity is calculated for the duration of the booking that the user spent driving and is used to find falsely recorded odometer values. Maximum fuel difference per second is a measure for tracking the largest recorded fuel difference per second in percent between two heartbeat signals during a booking. This is used to find falsely recorded fuel values, like the one shown in Figure 3.1. The two values are calculated for every booking in the data set and all bookings whose measured mean velocity or maximum fuel difference exceed specified thresholds are then removed. As the errors in false recordings typically are large, the thresholds can be set to quite extreme values and still work as intended. This also makes sure that no correct booking entries are accidentally removed. The thresholds used are: |Maximum fuel difference per second (%)| $>$ 1 and Mean velocity (km/h) $>$ 150. We chose these thresholds as they are well above the possible values for an EV under normal use.

3.1.2 The final data set

Once the quality control is finished, we have a data set of 25765 bookings. This means that roughly 7000 bookings are removed by applying our quality control thresholds and by removing bookings based on our project delimitations. A full list of how many bookings are removed during each step can be seen in table 3.1. The rows in the table are computed individually on the initial data set, meaning that the actual intersection of removed bookings is smaller than the sum of bookings fulfilling each criteria.

Group	Number of bookings
Initial data set	32826
Bookings with $\lvert$Fuel difference per second (%)$\rvert$ > 1	611
Bookings with Mean velocity (km/h) > 150	13
Overnight bookings	4289
Bookings made by users with less than three bookings	2548
Remaining data set after quality control	25765

Table 3.1: Summary of the removed bookings.

After cleaning out the false recordings we want to get an overview of how the remaining data set looks like. To do this, we create various plots and tables, portraying for example how many individual users there are in the data or how many bookings have been made in a specific city. Some initial information regarding the data set can be seen in Table 3.2.

Group	Number
Bookings	25765
Business trips	25185
Private trips	580
Pools	89
Cites	38
Users	1774
Vehicle models	10
Unique vehicles	169

Table 3.2: Summary of the complete data set.

We then look into when and where bookings are scheduled. Figure 3.2 below shows how the start times of bookings are divided over hour of day, day of week and month of year. As expected, most bookings take place during the day on weekdays. Figures 3.3 and 3.4 show how many bookings have been made from each pool and city. City refers to the city that a booking originates from and pool the hub where the vehicle has its dedicated parking space in that city. Here we see, for example, that Gothenburg is the city with the largest amount of bookings. Finally, we look into which vehicles are popular by looking at how many bookings have been made with each vehicle model, see Figure 3.5.

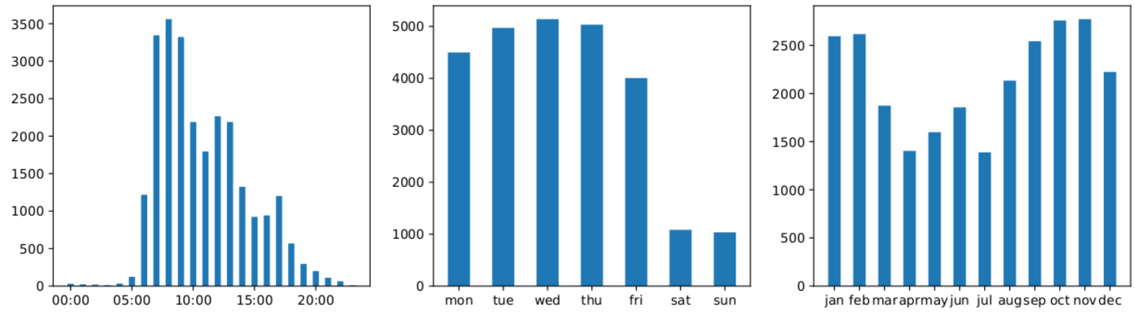

Figure 3.2: Bar charts displaying how many bookings start each hour of day, day of week and month of year.

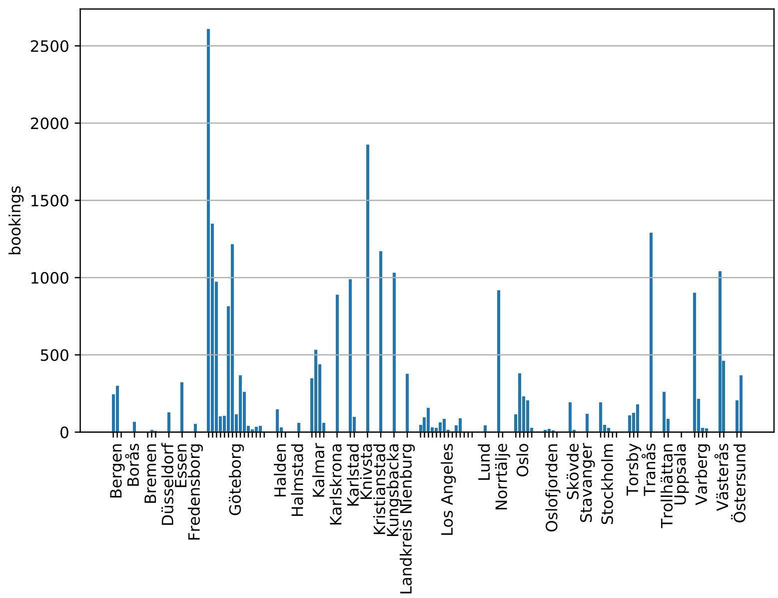

Figure 3.3: Bar chart displaying the number of bookings made from each pool, organized by city.

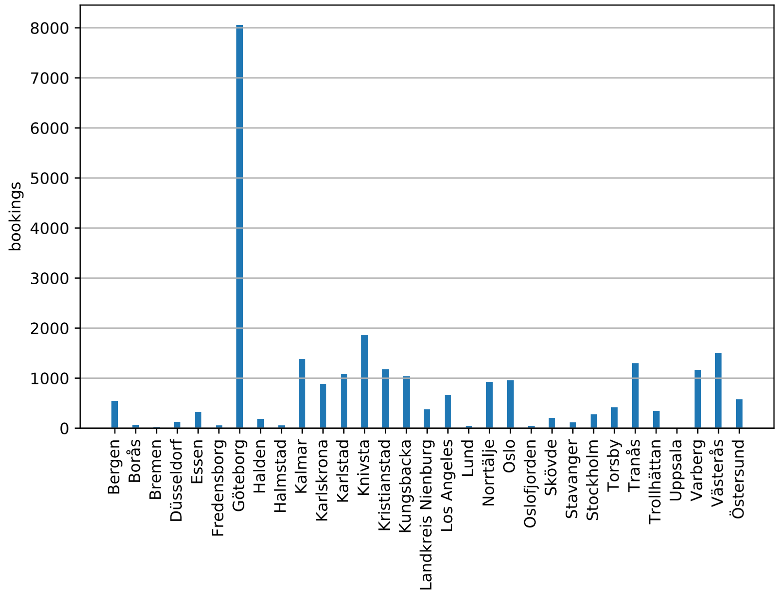

Figure 3.4: Bar chart displaying the number of bookings made from each city.

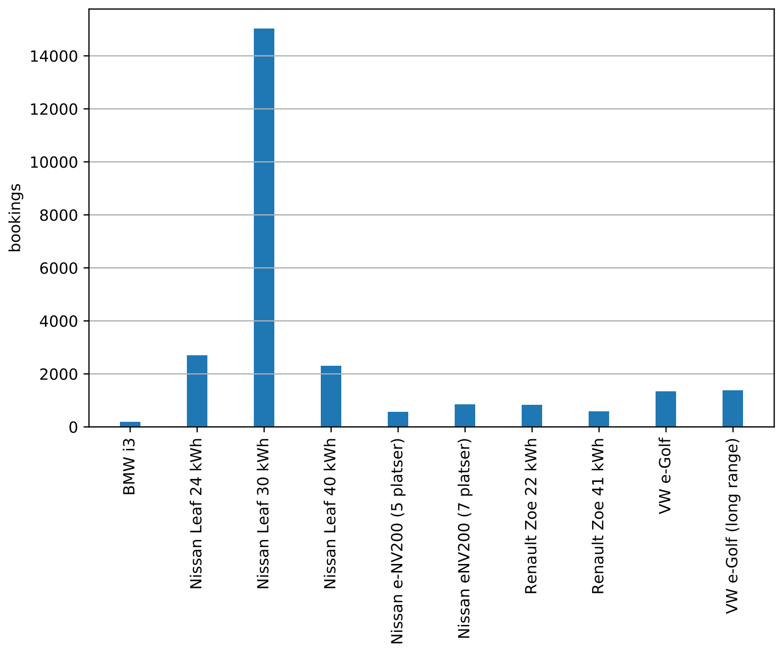

Figure 3.5: Bar chart displaying the number of bookings made with each vehicle model.

3.1.3 Feature extraction

From the quality controlled booking contents file, heartbeat file and trip segments file, it is possible to extract features that may be relevant when trying to predict energy consumption or cluster drivers based on their driving habits. Some of these features are drawn straight from a single data file, such as the users own estimated distance, while others have to be derived by combining information from one or multiple data files. An example of such a feature is the Power consumed which is derived by summing the negative differences in power level from consecutive heartbeat signals during a single booking.

The features are split up into regression and clustering features. The extraction of regression features is described in Subsection 3.1.3.1 and the extraction of clustering features in Subsection 3.1.3.2.

3.1.3.1 Regression features

The regression features are quite heavily limited by the fact that the information they make use of has to be accessible prior to the start of a booking. Once a booking is placed, the model immediately has to be able to make a prediction for how much power is going to be consumed. Many of the regression features are therefore direct translations of the information received with the placement of a new booking. Estimated distance is a clear example of this. A complete list of the basic regression features can be seen in Table 3.3.

There are different arguments for the different selections of basic regression features, however, they are all selected under the assumption that they have a correlation to power consumption. Booking duration is selected in the belief that longer bookings generally mean a higher power consumption. This may seem obvious, however, it is not uncommon for a booking to have large periods of downtime. Estimated distance is potentially a great feature when considering that the correlation between distance traveled and power consumed is proven to be large for EV's (Wu et al., 2015). The issue is the factor of human error. Predicting distance is not easy, in fact, there are cases when you may not even know how far you are going to drive at the time of placing the booking. This reduces the accuracy of the distance predictions and thereby reduce the significance of this feature. The features hour of day, day of week and month of year are all included in an attempt to catch eventual differences in consumption during different time frames. Hour of day to identify rush hours, day of week to separate things such as weekend driving when there might be reduced traffic and lastly month of year to identify seasonal differences such as temperature and other weather related driving conditions.

Finally, we have a few features for estimating the effects that city, pool, and vehicle model has on power consumption. To contextualize this information and make it usable for regression we calculate mean values for the consumption per kilometer and consumption per hour. We do this for all bookings with vehicles in a certain

Feature	Description
Booking duration (h)	Length of a booking.
Estimated distance (km)	User's own estimated distance.
Foresight (h)	Time between the placement of a booking and its occurrence.
Trip type	Business or private trip (0-1).
Hour of day	Hour that a booking was initialized (0-23).
Day of week	Day that a booking took place (0-6).
Month of year	Month that a booking took place (0-11).
City mean consumption (Wh/km)	The mean consumption per kilometer of bookings made within a city
City mean consumption time (Wh/h)	The mean consumption per hour of bookings made within a city
Pool mean consumption (Wh/km)	The mean consumption per kilometer of bookings made within a pool
Pool mean consumption time (Wh/h)	The mean consumption per hour of bookings made within a pool
Vehicle model mean consumption (Wh/km)	The mean consumption per kilometer of bookings made with a specific vehicle model
Vehicle model mean consumption time (Wh/h)	The mean consumption per hour of bookings made with a specific vehicle model

Table 3.3: A list of the features used for regression analysis.

pool or city and for bookings with certain vehicle models. The intuition is that these new features are able to capture some of the specialized driving conditions surrounding a city or pool as well as the impact of driving with a certain vehicle model.

In addition to using the basic regression features, we also introduce polynomial features. Polynomial features are new features that are either the multiplication of two or more basic features or a basic feature raised to some power. The reason for introducing polynomial features is to allow our prediction models to also capture non linear relationships. Optimally, we would want to try all possible combinations of polynomial features using our basic feature list. However, in order to make the comparison of features manageable we limit ourselves to second degree polynomials. Examples of polynomial features that may be effective for making predictions are Booking duration times the consumption per hour features or estimated distance times the consumption per kilometer features.

3.1.3.2 Clustering features

The goal of our clustering features is to divide users into groups with similar driving habits. To achieve this, the chosen features are focused on capturing the individual driving behavior of a user. A complete list of the chosen clustering features can be found in Table 3.4.

Feature	Description
User mean consumed (Wh)	The mean consumption of bookings made by a specific user.
User mean trip velocity (km/h)	The mean velocity of bookings made by a specific user.
User max velocity (km/h)	The highest velocity a user has ever traveled.
User mean consumption (Wh/km)	The mean consumption per kilometer of bookings made by a user.
User mean consumption time (Wh/h)	The mean consumption per hour of bookings made by a user.
Mean driving percentage (%)	The mean percentage of time spent driving in bookings made by a user.

Table 3.4: A list of features used for clustering analysis.

User mean consumed describes how much energy a user consumes on average per booking. This feature indicates whether a user has higher or lower consumption on average. User mean trip velocity and User max velocity are features that intend to discover if a user frequently engages in highway driving, something known to impact power consumption (Wu et al., 2015). User mean consumption and User mean consumption time are similar to the city, pool and vehicle model features used for regression but are instead focused on the individual driver. This in an attempt to capture the driving efficiency of a user, both per kilometer and per hour. The final feature, Mean driving percentage, describes the average percentage of time a user spent driving over all bookings. That is, the average percentage of time they were not parked. The goal of the feature is to distinguish users that have large amounts of downtime during bookings from users that do not.

3.1.4 Data split

After the feature selection is done we split the data set into three subsets, a training set containing 60% of the data points and a validation set and a test set each containing 20% of the data points. The training set is for training our regression models, the validation set for validating their performance during training and the test set for the final evaluation of each model. The test set is thereby removed and kept separate from the rest of the data until the final evaluation of each model. As previously mentioned, the split into the three data sets is done individually for each user. This in order to guarantee that the bookings of each user are properly split according to the 60-20-20 percentages across the three data sets and that there is at least one booking from each user per set. Not splitting the bookings individually for every user could lead to certain users having little to no validation or test data. This may not be a problem for the standalone regression models since they use the entire data set, but for the regression models targeting smaller user groups this could cause problems. In particular, validation and test sets for individual clusters might become smaller than intended and therefore give misleading evaluation results. The splitting of each user's bookings is done randomly.

3.2 Data analysis

In the data analysis phase, we create plots and tables investigating the importance of the various regression features. Only the training set is used to avoid creating a bias for certain features.

In order to understand what features have an impact on the power consumption of a trip, we investigate the linear correlation between each basic feature and the power consumed. We do this using Pearson's correlation coefficient, ρ. As explained in Section 2.1.1, coefficient values close to 1 and -1 correspond to strong positive and negative linear correlations respectively while values close to 0 correspond to no correlation. Because we are interested in linear correlation, regardless of direction, we sort the results by the absolute value of the coefficient. The result can be seen in Table 3.5. We see that two features, booking duration and estimated distance, have a significantly higher correlation to power consumption than the remaining features. Following them, there are three features with a correlation ($|\rho| > 0.1$) and the remaining features having little to no correlation to power consumption ($|\rho| < 0.1$).

Feature	Pearson's R (ρ)
Booking duration (h)	0.446
Estimated distance (km)	0.419
Pool mean consumption time (Wh/h)	0.233
City mean consumption time (Wh/h)	0.187
Hour of day	-0.147
City mean consumption (Wh/km)	0.099
Vehicle type mean consumption time (Wh/h)	0.093
Vehicle type mean consumption (Wh/km)	0.072
Day of week	0.067
Pool mean consumption (Wh/km)	0.063
Foresight (h)	0.025
Trip type	0.017
Month of year	0.013

Table 3.5: Pearson's correlation coefficient describing the linear correlation between each basic regression feature and power consumption. The values are ordered by the strength of the correlation, regardless of direction.

The same thing is done for the polynomial features with Table 3.6 showing the three polynomial features with highest Pearson's R values. There is a notable increase in strength of correlation compared to the basic regression features.

Feature	Pearson's R (ρ)
Booking duration (h) * Pool mean consumption time (Wh/h)	0.608
Booking duration (h) * Estimated distance (km)	0.553
Booking duration (h) * City mean consumption time (Wh/h)	0.543

Table 3.6: Pearson's correlation coefficient describing the linear correlation between three polynomial regression features and power consumption. The selected features are the three features showing the highest correlation, regardless of direction.

In addition to measuring correlation, we select the two features showing the highest ρ values and plot them versus power consumption. This in order to visualize the strength of the correlation. The plots can be seen in Figures 3.6 and 3.7.

To display the effects that pool, city and vehicle model have on power consumption, we plot the mean consumption per kilometer of all bookings made from a pool (Figure 3.8 and Figure 3.9), city (Figure 3.10) or with a certain vehicle model (Figure 3.11). The plots are box plots with the box covering the points closest to the median and including 50% of the data points. The line going through the box is displaying the median value. The whiskers cover the top and bottom most data points but excludes outliers. Points lying 1.5 times the size of the box above the top or below the bottom of the box are considered outliers and are depicted as circles.

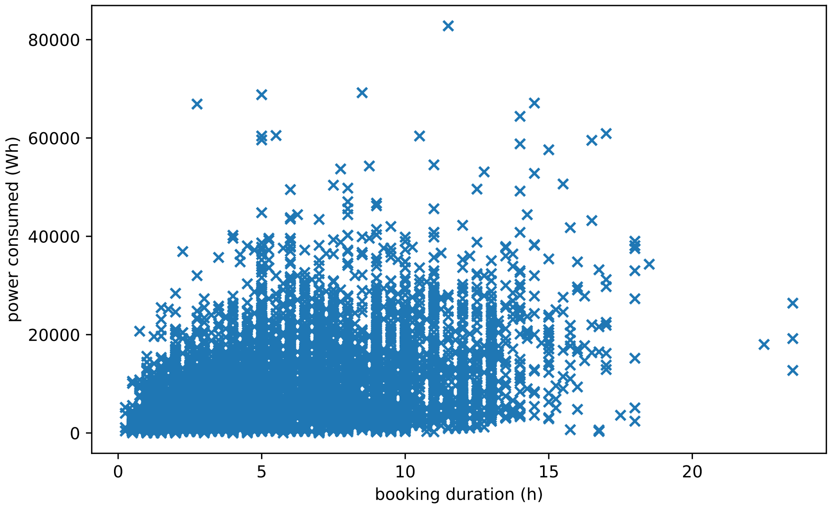

Figure 3.6: A scatter plot of booking duration and power consumed for bookings in the data set. The opacity of each point is set to 10%.

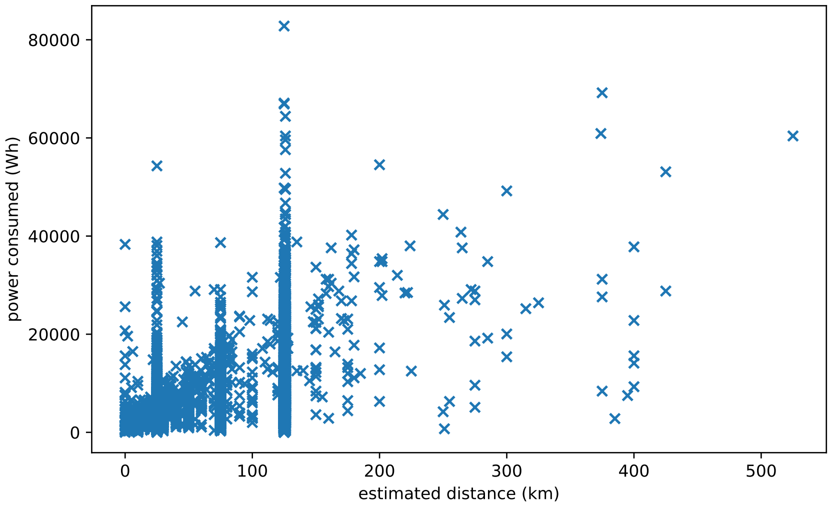

Figure 3.7: A scatter plot of estimated distance and power consumed for bookings in the data set. The opacity of each point is set to 10%.

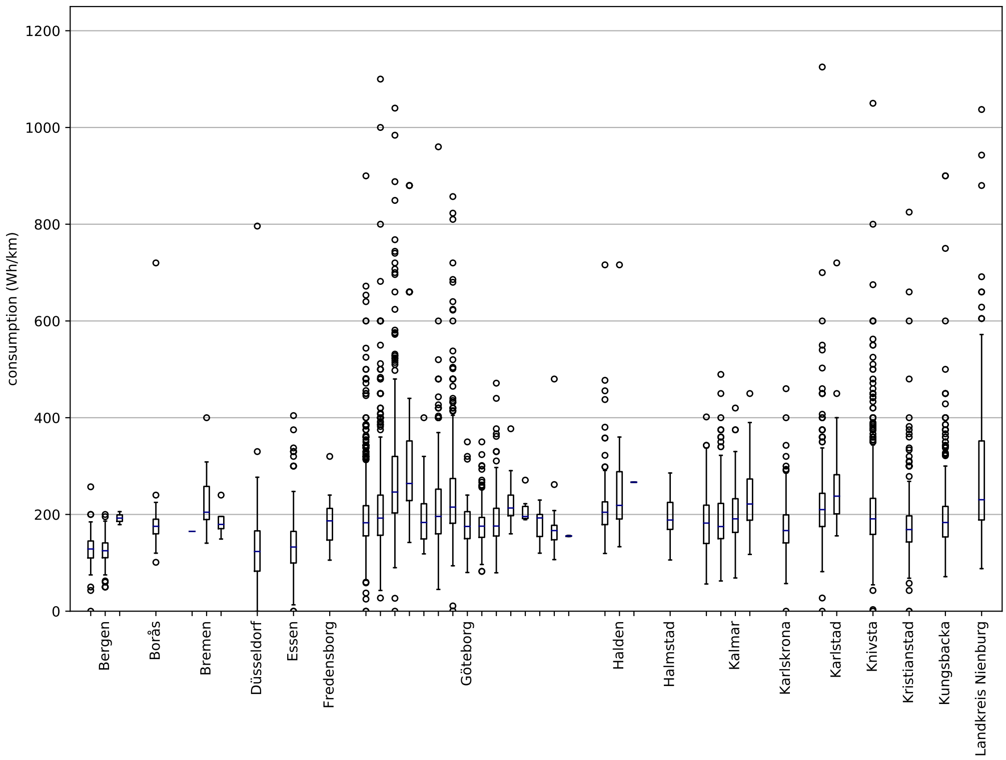

Figure 3.8: A box plot of booking consumption per kilometer for bookings made in the different pools, organized by city name.

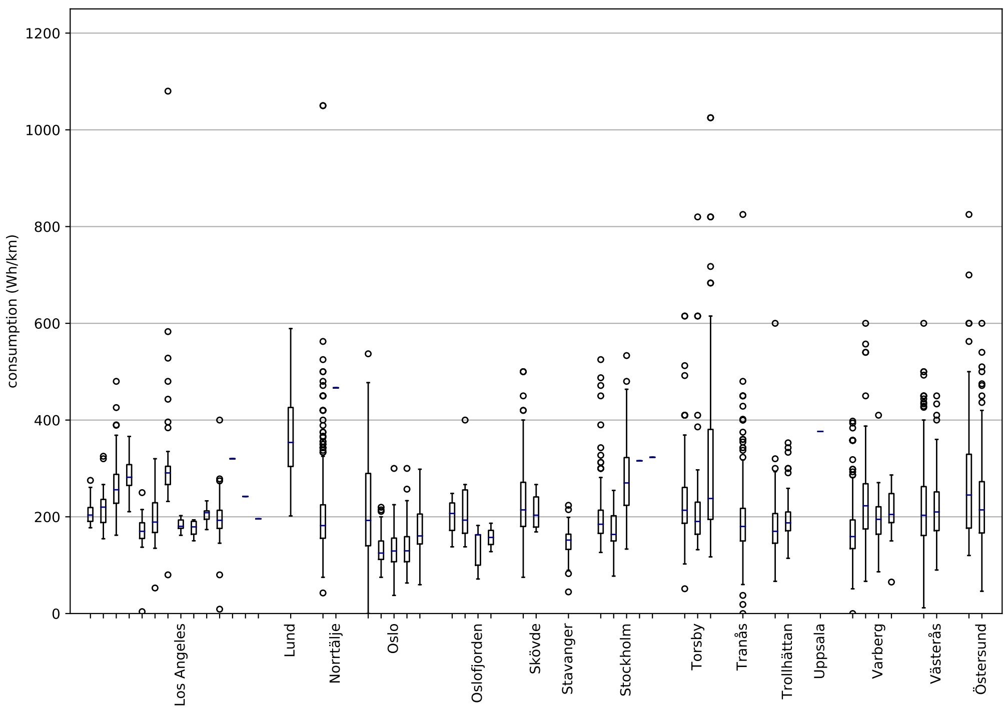

Figure 3.9: A box plot of booking consumption per kilometer for bookings made in the different pools, organized by city name.

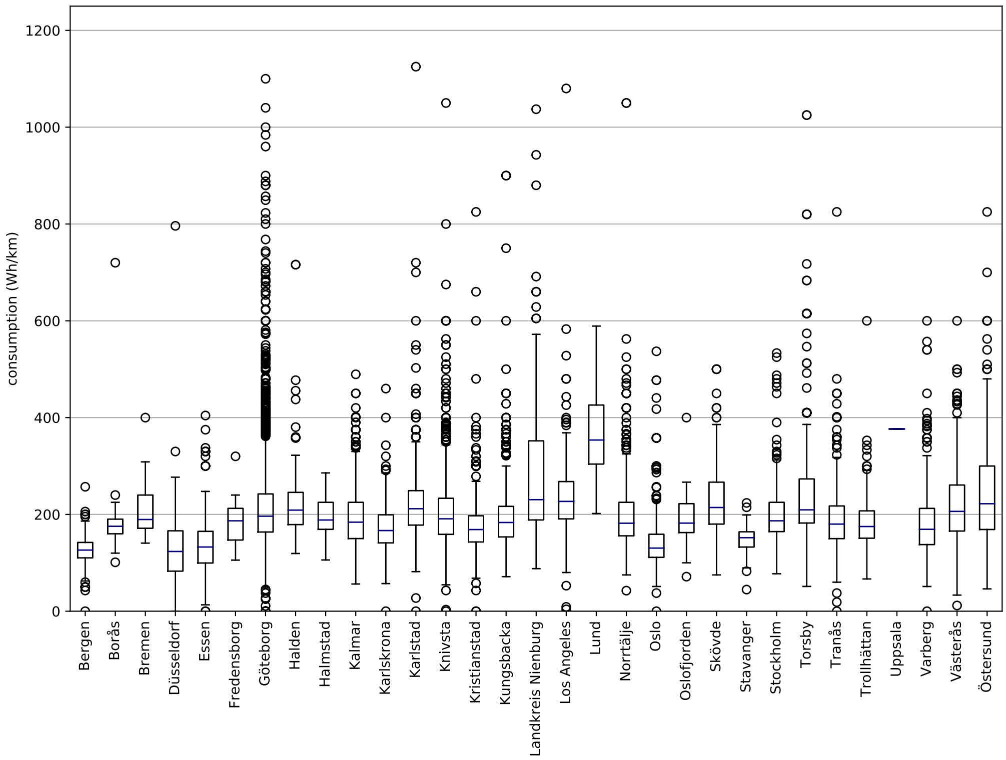

Figure 3.10: A box plot of booking consumption per kilometer for bookings made in the different cities.

3.3 Regression Analysis

In order to verify the relations discovered during the analysis phase, and make predictions for new data points, we use regression analysis. We choose to limit our scope to creating linear regression models due to their popularity, effectiveness, and ease of use (Sen & Srivastava, 2012b). The goal of the regression analysis phase is to find two sets of regression models. The first with a simple baseline model that can be used as a reference point when evaluating the performance of the more advanced models. The second with a fixed number of more advanced regression models that show the best performance compared to the number of features. It is generally easier to work with simpler models, and we know that reducing the number of features often reduces multicollinearity as well as the risk of overfitting (Sen & Srivastava, 2012d; Yan & Su, 2009). To enable comparison between all models, performance will be measured in mean squared error (MSE) for all models, regardless of which loss function was used during training. The MSE of a model is calculated by diving the sum of all squared errors by the total number of predictions, as seen in Equation 3.1:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 \tag{3.1}$$

where M is the number of validation data points.

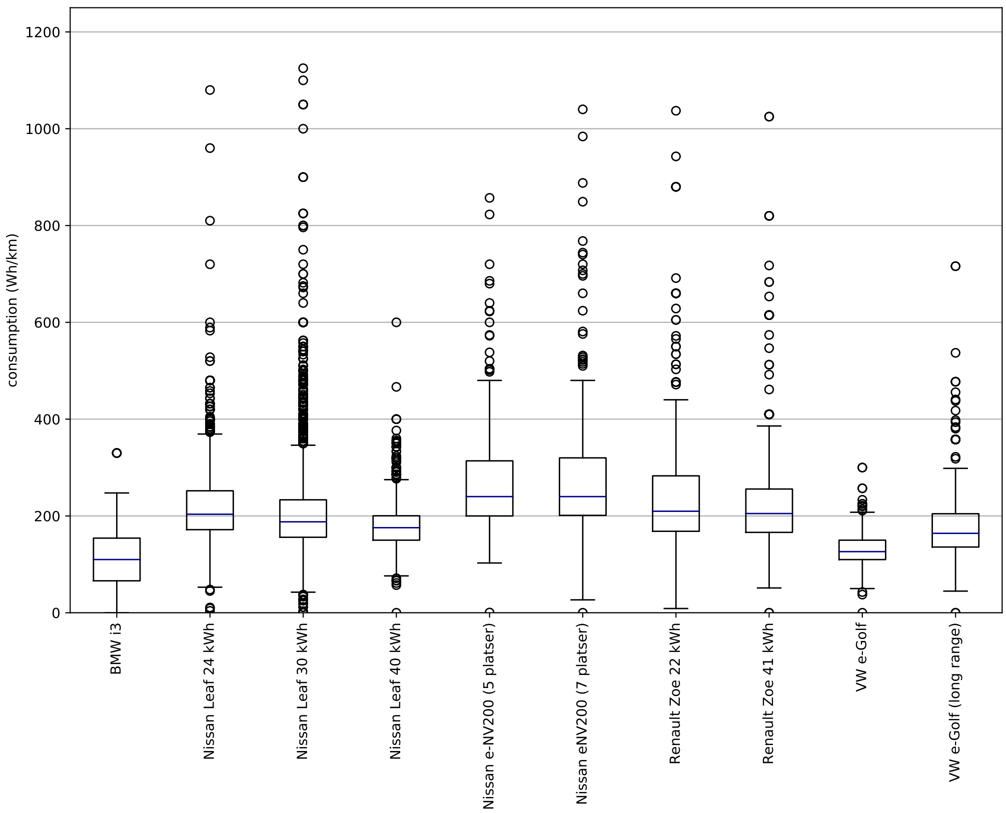

Figure 3.11: A box plot of booking consumption per kilometer for bookings made with different vehicle models.

We use and compare two different optimization methods for our linear regression models, ordinary least squares (OLS) (Section 2.2.1), and stochastic gradient descent (SGD) (Section 2.2.2). For each method, we use exhaustive parameter search to determine what features and hyperparameters yield the best results on the validation data.

3.3.1 Baseline model

The purpose of the baseline model is to act as a point of reference when evaluating the more complex models. To keep it as simple as possible we use OLS, as it is the least complex method in terms of available parameters. For the same reason we also limit ourselves to a single, non polynomial feature. A list of the best performing OLS models with a single, non polynomial feature can be seen in Table 3.7.

Feature	R^2	MSE
Booking duration (h)	0.221	4.303×10^7
Estimated distance (km)	0.166	4.608×10^7
Pool mean consumption time (Wh/h)	0.033	5.344×10^7
Hour of day	0.019	5.419×10^7
Vehicle model mean consumption time (Wh/h)	0.018	5.426×10^7

Table 3.7: MSE and R^2 values for models based on a single feature.

We chose the best performing model to be our baseline model. The complete model with weights can be seen in Equation 3.2:

$$f(x) = 1815.6996 + 1092.0212x \tag{3.2}$$

where x is Booking duration (h).

3.3.2 OLS models

Once we have selected the baseline model we investigate more complex OLS models. For these models, we include both basic and polynomial features. In particular, we consider OLS models with one, two and three features, and select the top five performing models for each category. The resulting models can be seen in Tables 3.8, 3.9 and 3.10.

We see that booking duration multiplied by the pool mean energy consumption per hour is the best individual feature. This is reasonable as it also displays the highest Pearson's R correlation in Section 3.2. In fact, we can see that all top 5 best performing models with a single feature include booking duration in some way, which is the basic feature with the highest Pearson's R correlation. Similarly, when considering models with two features, we see that the second feature of all models includes estimated distance in some way. This is the basic feature with the second

Feature	R^2	MSE
Booking duration (h) * Pool mean consumption time (Wh/h)	0.336	3.670×10^7
Booking duration (h) * Estimated distance (km)	0.316	3.780×10^7
Booking duration (h) * City mean consumption time (Wh/h)	0.269	4.039×10^7
Booking duration (h) * Vehicle model mean consumption time (Wh/h)	0.267	4.049×10^7
Booking duration (h) * Vehicle model mean consumption (Wh/km)	0.230	4.253×10^7

Table 3.8: MSE and R^2 values for the five best performing OLS models based on a single feature.

Feature	R^2	MSE
Booking duration (h) * Pool mean consumption time (Wh/h), Vehicle model mean consumption time (Wh/h) * Estimated distance (km)	0.403	3.297×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Estimated distance (km)	0.394	3.347×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), City mean consumption (Wh/km) * Estimated distance (km)	0.393	3.352×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Vehicle model mean consumption (Wh/km) * Estimated distance (km)	0.392	3.357×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Pool mean consumption (Wh/km) * Estimated distance (km)	0.392	3.358×10^7

Table 3.9: MSE and R^2 values for the five best performing OLS models based on two features.

Feature	R^2	MSE
Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h), Booking duration (h)	0.416	3.226×10^7
Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h), Booking duration (h) * Vehicle model mean consumption (Wh/km)	0.411	3.254×10^7
Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h), Booking duration (h)^2	0.411	3.255×10^7
Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h), Vehicle model mean consumption time (Wh/h) * Estimated distance (km)	0.410	3.257×10^7
Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h), Booking duration (h) * City mean consumption (Wh/km)	0.410	3.259×10^7

Table 3.10: MSE and R^2 values for the five best performing OLS models based on three features.

highest Pearson's R correlation. Adding such a feature decreases the loss of each model by a substantial amount when compared to the models with a single feature. Adding a third feature, however, leads to a quite small decrease in loss in comparison, and the difference in loss becomes even less significant with each increase in number of features. The performance of the best performing OLS models with one to six features can be seen in Table 3.11.

Number of features	R^2	MSE
1	0.336	3.670×10^7
2	0.403	3.297×10^7
3	0.416	3.226×10^7
4	0.420	3.206×10^7
5	0.422	3.193×10^7
6	0.424	3.184×10^7

Table 3.11: A table showing the R^2 and loss values for the best performing OLS models with one to six features.

Considering this decline of increased performance, and the benefit of using more easily interpretable models with fewer features, we choose to bring forward the best performing regression model with two features for the final evaluation.

3.3.3 SGD models

In addition to creating models using OLS, we also create models using SGD. This to try the Huber loss function (Section 2.2.3) and create models with regularizers (Section 2.2.4). Before training the SGD models we normalize the input data. Normalization is done to improve the convergence time of SGD and corresponds to giving each feature in the training set a mean of zero and a variance of one (Orr & Müller, 2003). Apart from normalizing the input, we use the same procedure as for OLS and select the top five performing models with one, two and three features respectively. This is done separately for SGD with Squared loss and SGD with Huber loss. In both cases, the choice of regularizer is included in the exhaustive parameter search, with the options being Lasso, Ridge or none. The best performing SGD models with Squared loss can be seen in Tables 3.12, 3.13 and 3.14, and the best performing SGD models with Huber loss can be seen in Tables 3.15, 3.16 and 3.17.

Feature	Regularizer	R^2	MSE
Booking duration (h) * Pool mean consumption time (Wh/h)	None	0.336	3.670×10^7
Booking duration (h) * Estimated distance (km)	None	0.32	3.758×10^7
Booking duration (h) * City mean consumption time (Wh/h)	None	0.269	4.039×10^7
Booking duration (h) * Vehicle model mean consumption time (Wh/h)	Ridge	0.268	4.046×10^7
Booking duration (h) * Vehicle model mean consumption (Wh/km)	Ridge	0.231	4.249×10^7

Table 3.12: Regularizer, MSE and R^2 values for the five best performing SGD models based on a single feature using squared loss.

Comparing the models using the two different loss functions, we see that the they yield very similar results both in terms of loss and in terms of best performing features. The same is true when comparing the SGD models to the corresponding OLS models. Again, we see very similar loss values and top performing features. We also see the same insignificant decrease in loss when going from two to three or more features.

3.4 Cluster-based regression

In order to make more accurate predictions of power consumption we want to split the users of our data set into groups and then create different prediction models targeting each individual group. We find these user groups using cluster analysis. In particular, we use two clustering methods, K-means and density-based spatial clustering of applications with noise, DBSCAN (see Section 2.3). We use K-means

Features	Regularizer	R^2	MSE
Booking duration (h) * Pool mean consumption time (Wh/h), Vehicle model mean consumption time (Wh/h) * Estimated distance (km)	Ridge	0.404	3.291×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Estimated distance (km)	Ridge	0.395	3.342×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), City mean consumption (Wh/km) * Estimated distance (km)	Ridge	0.395	3.345×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Estimated distance (km) * Pool mean consumption (Wh/km)	Ridge	0.393	3.351×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Estimated distance (km) * Vehicle model mean consumption (Wh/km)	Ridge	0.393	3.352×10^7

Table 3.13: Regularizer, MSE and R^2 values for the five best performing SGD models based on two features using squared loss.

Features	Regularizer	R^2	MSE
Booking duration (h), Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h)	None	0.420	3.205×10^7
Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h), (Booking duration)^2	Lasso	0.414	3.235×10^7
Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h), Estimated distance (km) * Pool mean consumption time (Wh/h)	Ridge	0.411	3.252×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Vehicle model mean consumption time (Wh/h) * Estimated distance (km), Hour of day	Ridge	0.409	3.263×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Day of week, Vehicle model mean consumption time (Wh/h) * Estimated distance (km)	Ridge	0.408	3.268×10^7

Table 3.14: Regularizer, MSE and R^2 values for the five best performing SGD models based on three features using squared loss.

Features	Regularizer	R^2	MSE
Booking duration (h) * Pool mean consumption time (Wh/h)	Ridge	0.337	3.665×10^7
Booking duration (h) * Estimated distance (km)	None	0.32	3.759×10^7
Booking duration (h) * City mean consumption time (Wh/h)	Ridge	0.269	4.039×10^7
Booking duration (h) * Vehicle model mean consumption time (Wh/h)	Ridge	0.267	4.051×10^7
Booking duration (h) * Vehicle model consumption (Wh/km)	None	0.228	4.263×10^7

Table 3.15: Regularizer, MSE and R^2 values for the five best performing SGD models based on a single feature using Huber loss.

Features	Regularizer	R^2	MSE
Booking duration (h) * Pool mean consumption time (Wh/h), Vehicle model mean consumption time (Wh/h) * Estimated distance (km)	Lasso	0.401	3.311×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Estimated distance (km)	Lasso	0.38	3.426×10^7
Booking duration (h) * Estimated distance (km), Vehicle model mean consumption time (Wh/h) * Pool mean consumption time (Wh/h)	None	0.378	3.438×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Estimated distance (km) * Vehicle model consumption (Wh/km)	Ridge	0.375	3.451×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Estimated distance (km) * Pool mean consumption (Wh/km)	Lasso	0.373	3.463×10^7

Table 3.16: Regularizer, MSE and R^2 values for the five best performing SGD models based on two features using Huber loss.

Features	Regularizer	R^2	MSE
Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h), (Booking duration)^2	None	0.418	3.218×10^7
Booking duration (h), Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h)	None	0.407	3.278×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Vehicle model mean consumption time (Wh/h) * Estimated distance (km)	Lasso	0.401	3.311×10^7
Booking duration (h) * City mean consumption (Wh/km), Booking duration (h) * Pool mean consumption time (Wh/h), Vehicle model mean consumption time (Wh/h) * Estimated distance (km)	Lasso	0.4	3.315×10^7
Booking duration (h) * Vehicle model mean consumption time (Wh/h), Booking duration (h) * Pool mean consumption time (Wh/h), Vehicle model mean consumption time (Wh/h) * Estimated distance (km)	Ridge	0.399	3.319×10^7

Table 3.17: Regularizer, MSE and R^2 values for the five best performing SGD models based on three features using Huber loss.

as it is one of the most popular clustering methods due to its efficiency and empirical success (Jain, 2010). It has some drawbacks however, such as requiring the number of clusters to be specified in advance. In order to counteract these drawbacks, we also use DBSCAN.

Both K-means and DBSCAN cluster data sets by measuring distances between data points in the feature space. As the ranges of the various clustering features are of different magnitude, we need to scale the features in order for each feature to become equally influential. To do this, we normalize the input data using z-score standardization giving it a mean of zero and a variance of one. This technique is widely used and studies have found it to work well for clustering (Mohamad & Usman, 2013).

Once we have normalized the data and created our different splits of user groups using K-means and DBSCAN, we use exhaustive search in order to find the best regression model for each group in each split. As described in Section 2.2, the best performing stand alone regression model with two features is selected as the preferred prediction model. To make a fair assessment of whether predictions improve from the use of clustering, we limit our cluster based regression models to two features. This also helps reduce the complexity of finding the best possible regression model for each group. Considering how similarly the OLS and SGD models perform and the fact that OLS runs much faster with just two features, we reduce the complexity further, by limiting our choice of optimization method to OLS. The two subsections below contain the best splits of user groups created with the K-means and DBSCAN methods respectivly.

When comparing the results of our different user splits, we compare the splits to each by what we call combined loss. To get the combined loss for a group of clusters created in a split, we start by summing the individual squared losses of the best performing regression models targeting each cluster. We then divide the total squared loss by the total number of data points in the validation set. This gives us the combined mean squared error of all models targeting clusters created in one split. Apart from the combined loss, we also take into consideration the individual loss of the worst performing clusters in each split and the sizes of individual clusters. The losses of individual clusters are important as it may be undesirable to give poor predictions even for a small cluster of the total population in a real world implementation. The sizes are important as a certain amount of data is required to properly train and evaluate a regression model.

3.4.1 K-means

When using K-means there are two parameters that need to be specified, the clustering features and K, the number of clusters to create. As seen in Table 3.4, we have extracted a total of 6 driver-based clustering features. We can select any combination of these to use when clustering drivers, resulting in a total of 63 unique combinations. For each combination of selected features, we can choose any number for K between 1 and the total number of users, 1774. In summary, the total number of combinations is too large to investigate and we need to limit ourselves. In order to better understand how the number of created clusters impact the combined loss of our cluster specific regression models, we investigate how the loss is affected by increasing the number of clusters. Figure 3.12 shows how the combined loss decreases as the number of clusters increase for one and two clustering features respectively.

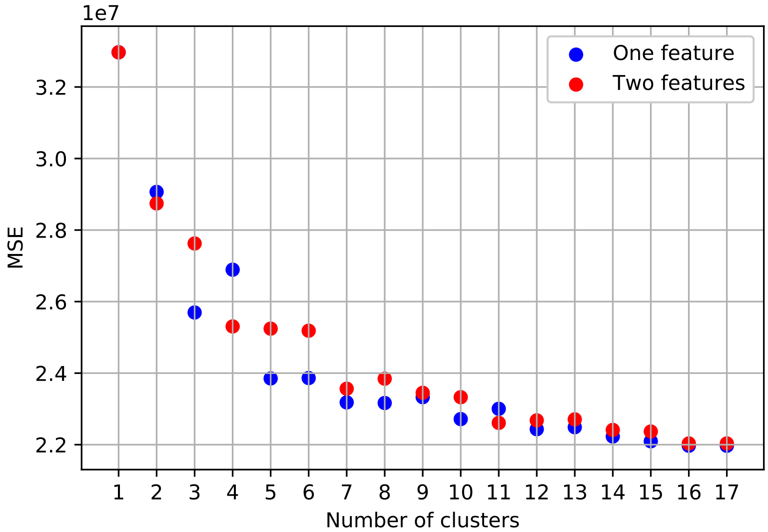

Figure 3.12: Plot showing how the combined loss decreases as the number of clusters increase for the best performing splits with one and two features respectively.

We can see that for both one and two features the loss decreases with the number of clusters. We can also see that the decrease is declining with every increase in number of clusters. Apart from the effects on loss, we must also consider how creating more clusters impact the sizes of the created clusters. Since we use regression analysis individually on each cluster, we do not want clusters that are too small as they run the risk of not having enough training or validation data for their regression models. The more clusters you create in each split, the smaller they become on average meaning that splits with more clusters run a greater risk of creating clusters that are poorly suited for regression (Hox, Moerbeek, & Van de Schoot, 2017). Consequently, we limit our comparison to user splits with between two and seven clusters. Table 3.18 shows the top five performing splits of all combinations of clustering features for two to seven clusters. K represents the number of clusters, combined MSE the combined loss for the regression models targeting the clusters created in a split, max cluster MSE the cluster with the largest individual loss in a split and min cluster training size the cluster with the smallest number of bookings.

By looking at these results, we can see that using the single clustering feature User mean consumption time gives us the best performing split in terms of lowest combined loss. However, as previously mentioned, loss is not the only factor to consider when choosing a split. We can see that the minimum cluster sizes for the top five splits are quite small. In order to guarantee that the regression models targeting each cluster has had enough data points to train on, we also select the

K	Clustering features	Combined	Max cluster MSE	Min cluster training size
7	User mean consumption time (Wh/h)	2.318×10^7	1.033×10^8	15
5	User mean consumption time (Wh/h)	2.385×10^7	7.625×10^7	31
6	User mean consumption time (Wh/h)	2.386×10^7	8.586×10^7	18
7	Mean driving percentage (%), User mean consumption time (Wh/h), User mean trip velocity (km/h)	2.500×10^7	1.034×10^8	26
7	User mean consumption (Wh/km), User mean consumption time (Wh/h), User mean trip velocity (km/h)	2.537×10^7	1.027×10^8	141

Table 3.18: The top five performing cluster splits created with K-means for any combination of clustering features and with a K between two and seven. Min cluster training size represents the minimum number of bookings made by users in one cluster.

fifth best model from Table 3.18 to bring forward for evaluation. This due to it satisfying the condition of having a minimum cluster size greater than 100.

3.4.2 DBSCAN

In addition to creating cluster-based regression models using K-means, we also create models using DBSCAN. Here, in addition to all possible combinations of clustering features, we have to consider two parameters, Eps and MinPts. Previous studies have attempted to find ways of selecting good values for these parameters without having to optimize over all possible values. When considering small numbers of features, it has been suggested to set the value of the MinPts parameter to two times the number of features (Schubert et al., 2017). Eps is considered more difficult to set, so in order to find appropriate combinations of features and values for the Eps parameters, we use exhaustive search. We create one split for each combination of features and for the following values of Eps: 0.05, 0.1, 0.25, 0.5, 0.75, 1.0. The value of MinPts is fixed to two times the number of selected features used in each split. The top five performing splits can be seen in Table 3.19. The column K in the table represents the number of clusters that was created in each split.

Looking at these results, we can see that the best performing splits have much larger values of K compared to K-means. We can also see that the minimum training sizes are much smaller than in the best performing splits found by K-means. This means that just like for K-means, we want to present an alternative split which is the best

K	Clustering features	Combined MSE	Max cluster MSE	Min cluster training size
61	Mean driving percentage (%), User mean consumption (Wh/km)	2.658×10^7	2.371×10^8	6
73	Mean driving percentage (%), User mean consumption time (Wh/h)	2.738×10^7	6.687×10^7	4
20	User mean consumed (Wh), User mean consumption time (Wh/h)	2.828×10^7	7.554×10^7	6
38	User mean consumption time (Wh/h), User mean velocity (km/h)	2.842×10^7	4.478×10^7	11
12	Mean driving percentage (%), User mean consumed (Wh), User mean consumption time (Wh/h)	2.845×10^7	8.025×10^7	7

Table 3.19: The top five performing cluster splits created with DBSCAN for any combination of clustering features and for the Eps values: 0.05, 0.1, 0.25, 0.5, 0.75, 1.0. Min cluster training size represents the minimum number of bookings made by users in one cluster.

performing split with minimum training sizes of 100. The result can be seen in Table 3.20.

K	Clustering features	Combined MSE	Max cluster MSE	Min cluster training size
2	User mean consumed (Wh), User max velocity (km/h), User mean consumption time (Wh/h), User mean velocity (km/h)	2.973×10^7	1.124×10^8	122

Table 3.20: The top performing cluster split created with DBSCAN for any combination of clustering features and for the Eps values: 0.05, 0.1, 0.25, 0.5, 0.75, 1.0. The split satisfies the condition of having more than 100 training data points in the smallest cluster. Min cluster training size represents the minimum number of bookings made by users in one cluster.

3.5 Evaluation

Finally, we compare how our best performing and specially selected models perform on the test set. The stand alone regression models with their respective validation and test set scores are listed in Table 3.21.

We see that our stand alone regression models all show improved performance when

Model	Test MSE	Val MSE
Best performing OLS model (2 features)	2.840×10^7	3.297×10^7
Best performing SGD model Huber loss (2 features)	2.859×10^7	3.311×10^7
Best performing SGD model squared loss (2 features)	2.906×10^7	3.291×10^7
Baseline model	3.801×10^7	4.303×10^7
Miveo's current method	9.236×10^7	8.995×10^7

Table 3.21: MSE values for the our best performing stand alone regression models evaluated on the test set. Miveo's current method and the baseline model is included for reference. Sorted by lowest test set MSE.

Model	**Test MSE**	**Regularizer**
Best performing OLS model (2 features)	2.840×10^7	N/A
Best performing SGD model Huber loss (2 features)	2.859×10^7	Lasso
Best performing SGD model squared loss (2 features)	2.906×10^7	Ridge
Baseline model	3.801×10^7	N/A
Miveo's current method	9.236×10^7	N/A

Table 3.22: TMP. DELETE!

evaluated on the test set. However, just like when evaluated on the validation set, they show very similar performance when compared to each other. We can also see that our stand alone regression models greatly reduce loss when compared to Miveo's current prediction method and the baseline model.

Model	Test MSE	Val MSE
Best performing K-means split	2.394×10^7	2.318×10^7
Best performing DBSCAN split (min cluster training size > 100)	2.756×10^7	2.973×10^7
Best performing OLS model (2 features)	2.840×10^7	3.297×10^7
Best performing DBSCAN split	2.924×10^7	2.658×10^7
Baseline model	3.801×10^7	4.303×10^7
Best performing K-means split (min cluster training size > 100)	6.376×10^7	2.537×10^7
Miveo's current method	9.236×10^7	8.995×10^7

Table 3.23: MSE values for the our best performing and specially selected models when evaluated on the test set. Miveo's current method, the baseline model and the stand alone OLS model is included for reference. Sorted by lowest MSE on the test set.

The clusted-based regression models and their respective validation and test set scores are listed in Table 3.23. For these models, we see a more varied performance. All of our models outperformed both the baseline model and our best performing OLS model when evaluated on the validation set, however, this is no longer the case with the test set. In particular we see that the best performing K-means split with a restriction show vastly different performance on the test set compared to the validation set. For our best performing cluster-based models, we see that the top two models outperform our stand alone OLS model with the same number of features. Our best performing model overall is the K-means split with no restriction on individual training sizes. This model shows a reduction in loss of 14% over the best performing stand alone OLS model with the same amount of features, a reduction in loss of 35% over the baseline model and a reduction in loss of 73% over Miveo's current method for predicting power consumption.

4

Discussion

This chapter includes discussion of the results and methods used in the project. It also includes discussion regarding data quality as well as a section regarding future work.

4.1 Prediction results

We have managed to create stand alone regression models capable of making more accurate predictions both compared to Miveo's current predictions method and compared to our own baseline model. We also see that our cluster-based regression models can outperform our stand alone regression models when both are limited to two regression features. However, we have not yet considered the mean squared errors of individual clusters in our cluster-based regression models. Tables 4.1 and 4.2 show the mean squared errors of the individual clusters of our best performing K-means splits with no restrictions and with cluster training sizes of at least 100 respectively. None of our selected K-means cluster-based models shows an improvement over the stand alone OLS model for every individual cluster. In fact, for the two selected splits shown in the tables, we can see clusters with individual mean squared errors that are close to double of what the stand alone regression model shows. The clustering algorithm has separated the users which are most difficult to make predictions for into one or two clusters, giving those clusters substantially higher mean squared error values. While this highlights which users are more difficult to make predictions for, it makes the model less suited to be used as it is for predicting power consumption for all users in a real life scenario. What you instead would want to do, is to separate the more difficult users and find some other way of predicting power consumption for those specific groups.

4.2 Regression features

The results of this thesis are greatly impacted by our choice of regression features. While we tried to extract as much different information as we could from our three data files, most of them turned out to show very little linear correlation to power

Regression features	Cluster training size	Test MSE
Booking duration (h) * Vehicle model mean consumption time (Wh/h), Booking duration (h) * Estimated distance (km)	3631	2.271×10^7
Booking duration (h) * Vehicle model mean consumption time (Wh/h), (Estimated distance)^2	850	4.043×10^7
Booking duration (h) * Pool mean consumption time (Wh/h), Month of year	15	1.815×10^8
(Booking duration)^2, Estimated distance (km) * Pool mean consumption time (Wh/h)	3667	1.679×10^7
Booking duration (h) * City mean consumption (Wh/km), Vehicle model mean consumption time (Wh/h) * Estimated distance (km)	1995	2.514×10^7
Booking duration (h) * Vehicle model mean consumption time (Wh/h), Trip type	239	1.093×10^8
Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h)	5062	2.107×10^7

Table 4.1: Individual cluster training sizes and test MSEs for the top performing K-means split with no restrictions on cluster training size.

consumption. Once again looking at the Tables 3.5 and 3.6 from Section 3.2, we see that Booking duration and Estimated distance are the basic features which show the highest linear correlation. We also see that when considering polynomial features there are features which show higher linear correlation. However, they are all features which include one or both of Booking duration or Estimated distance. Unsurprisingly, these same two features are included in all the best performing regression models created by using exhaustive search. With the obvious impact these features have on our regression models, we want to investigate their quality. Especially, we investigate Estimated distance since we know that users tend to make inaccurate estimations. A suspected reason for the many inaccurate user estimated distances is the user interface used to estimate distances when placing a booking with Miveo. In Figure 4.1, we can see this interface together with the plot of estimated distance against power consumed for bookings. Looking at the plot, we can see three spikes at 25, 75 and 125km, the distances corresponding to the radio buttons used to estimate distance in the interface. We can also see from the interface image that the default value when a user wants to travel further than 125km is 126km and that if a user wants to be even more specific, she can input an address. To highlight the issues with this user interface, we investigate what number of bookings are estimated using the preset radio buttons and how often these estimates result in an inaccurate estimation. Table 4.3 shows the percentage of estimations made with the various radio buttons and how many percent of them are off by 25, 50 and 75 percent. We see that a total of about 87 percent of all estimations were in fact made using

Regression features	Cluster training size	Test MSE
Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h)	5353	2.593×10^7
Booking duration (h) * Vehicle model mean consumption time (Wh/h), Booking duration (h) * Estimated distance (km)	141	1.778×10^8
Booking duration (h) * Vehicle model mean consumption time (Wh/h), Vehicle model mean consumption time (Wh/h) * Estimated distance (km)	1546	2.894×10^7
Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h)	2378	2.070×10^7
Booking duration (h) * Estimated distance (km), Booking duration (h) * Pool mean consumption time (Wh/h)	593	1.687×10^7
Booking duration (h) * Vehicle model mean consumption time (Wh/h), (Estimated distance)^2	1278	4.856×10^8
Booking duration (h) * City mean consumption time (Wh/h), Booking duration (h) * Estimated distance (km)	4170	1.901×10^7

Table 4.2: Individual cluster training sizes and test MSEs for the top performing K-means split with a minimum cluster training size of 100.

the radio buttons. We also see that over 85 percent of all predictions made using these default distances are over 25 percent off, and that over 35 percent of them are over 75 percent off. Changing the user interface to further aid users in making accurate distance estimations could greatly increase the quality of user predictions. Considering the importance of the Estimated distance feature for our regression models, this in turn would greatly improve predictions.

Estimate	Share of estimates	> %25 off	> %50 off	> %75 off
25km	27.3%	79.4%	58.4%	25.8%
75km	18.0%	88.1%	66.4%	33.2%
125km	32.9%	91.3%	77.1%	49.1%
126km	8.6%	76.3%	53.4%	24.9%
25, 75, 125 or 126km	86.8%	85.4%	66.6%	36.1%

Table 4.3: Shows the most frequently estimated distances and how many percent of all bookings have been made with each estimate. The three rightmost columns show how many percent of actual distances traveled were above or below the estimation with 25, 50 and 75 percent respectively.

Besides our best performing features, there are improvement which could be made to our features derived for estimating the effects that city, pool, and vehicle model has on power consumption. While the intention for these features is to capture the individual impact of a pool, city or vehicle model, this is not really the case. For example, if the majority of bookings made with one vehicle model have been made in one city, then the characteristics of that city will have great impact on the mean consumption of that vehicle model. The way these features are calculated also means that bookings, cities and vehicle models do not in any way influence predictions made in other pools, cities or with vehicle models that display similar characteristics. This would be desirable as bookings made in a city with a certain traffic density, for example, could be used to help make better predictions for bookings made in cities with similar traffic density.

4.3 Data quality

Besides the quality of our features, there are some general problems with the data provided by Miveo. As seen in Section 3.1.1, the initial data set contains falsely recorded values due to the techniques used for gathering data from the vehicles. Even though efforts were made to identify and remove falsely recorded values, it is likely that not all were found and that some made it through to the final data set. This is especially likely for false values which are within reasonable limits. In general, falsely recorded values make it harder to find the true correlation between variables and introduce uncertainty into the regression models. Falsely recorded fuel levels are particularly detrimental as fuel level is the dependent variable and cannot be ignored even if the recorded values are not trustworthy.

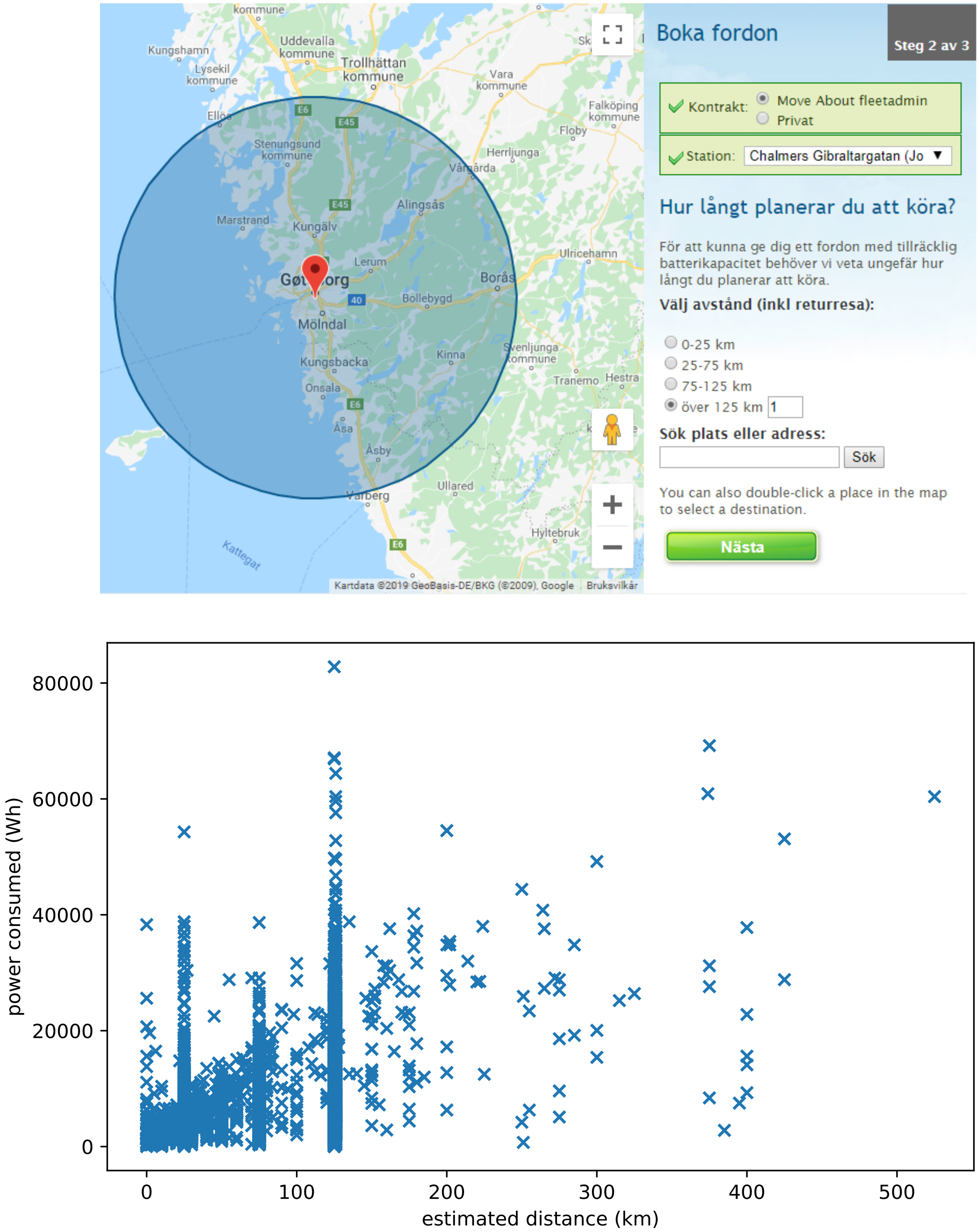

Figure 4.1: Top figure shows the user interface where this information has been entered into the system. Bottom figure shows how certain values are over represented in the estimated distances of drivers.

4.4 Clustering methods

When using clustering to separate our users into smaller groups with more similar driving habits, we see that our two selected clustering methods, K-means and DBSCAN show varying success. While both managed to improve upon the predictions of the standalone OLS model with the same number of regression features, K-means showed significantly better results. If we look at the characteristics of splits created by the two methods, we also see that DBSCAN creates splits with a much larger quantity of clusters compared to K-means. While the K-means method requires the parameter K, the number of clusters, to be specified in advance, DBSCAN requires two parameters, MinPts and Eps. These parameters turn out to be very difficult to optimize for our particular data set. An example of how the clustering changes when the parameters are altered can be seen in Figure 4.2. We see that for small changes in Eps, the user splits go from classifying the entire outer edges of the data set as a single cluster to classifying the entire middle area as a single cluster. We also see that increasing MinPts can reduce the number of clusters, but does so by increasing the size of the largest cluster, which does not make the split more useful. This makes us believe that DBSCAN is not very well suited for data sets similar to ours.

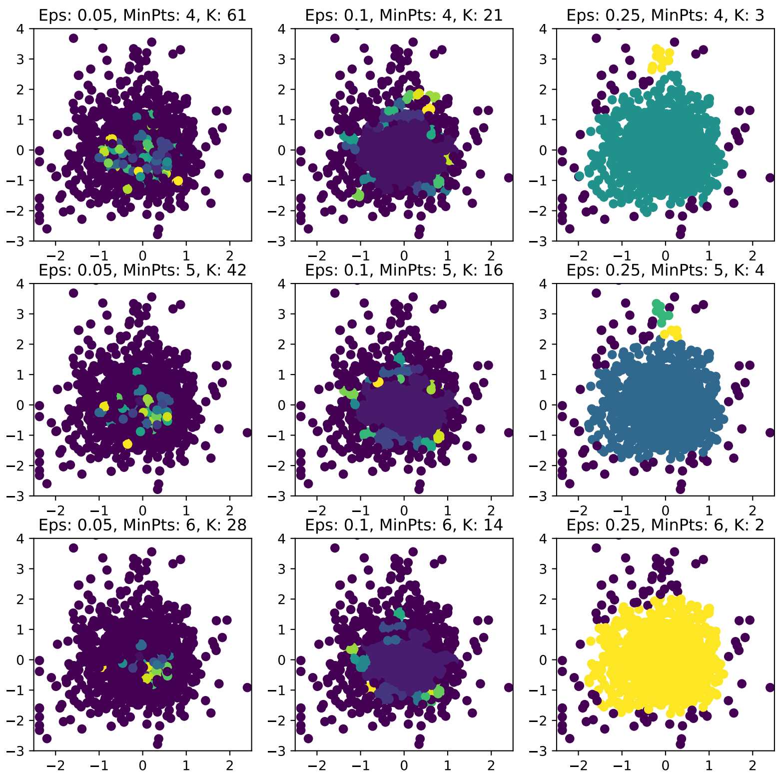

Figure 4.2: Plots showing how the clusters created with DBSCAN change when the Eps and MinPts parameters are altered. K is the number of clusters in each plot.

4.5 Future work

As described in Section 4.2, there are improvements to be made to the way that users estimate their distance. Looking closer at Figure 4.1, we see that there are several ways for a user to manually input a distance other than using the radio buttons. Users can enter longer distances in the small text field, enter an address in the larger text field or double click the map to select a destination. Together, all of these ways contribute to the points in the scatter plot that are not in any of the three lines representing the radio buttons. However, as we see in Table 4.3, the large majority of estimations are still made using the radio buttons. This indicates a lack of incentive for users to actually make accurate estimations. Because of this, we suggest future work to be made in collaboration with Miveo to alter the user interface and somehow introduce an incentive for users to make more accurate estimations.

Another way to improve predictions of power consumption is by extracting more useful features. As previously mentioned, the features estimating consumption based on pool, city and vehicle model are not completely accurate as they are simply the mean consumption of a number of bookings. A potential way to get around these inaccuracies could be to identify the characteristics of each city, pool and vehicle model and what effect they have on the total consumption. This way, what vehicles have been used for bookings in a city will not influence how that city contributes to future estimations. It would also mean that one city displaying some characteristics could help make better predictions for other cities with similar characteristics. Finding these characteristics, however, would require research and domain knowledge. Even though previous studies have shown that certain geographical properties, like road grade, have an impact on power consumption, it is not obvious how such information should be summarized for a pool or a city.

As mentioned in Section 3.4, we limit our cluster based regression models to a maximum of two features. This was done in order to make a fair comparison with the stand alone models. However, this does not mean that the cluster specific models would not benefit from using more than two features. For future studies, it would be interesting to investigate whether the more specialized, cluster specific models see a significant decrease in loss from an increased number of features beyond just two. It would then be necessary to do a similar analysis as we have done for the stand alone models to find out at which number of features the decrease in loss ceases to be significant.

5

Conclusion

In this thesis, we use linear regression to predict the power consumption of electric vehicles based on planned trips without knowledge of specific routes. In order to further improve upon these predictions, we group drivers with similar driving habits using clustering. Individual regression models are then trained for each group, in an attempt to make more specialized predictions.

We show that by using ordinary least squares regression and a few simple features such as the duration of a booking and an estimated distance, it is possible to make more accurate predictions compared to the predictions made by Miveo's current method. We also show that it is possible to achieve an even higher accuracy by first splitting the user base into smaller user groups. This is done by using the clustering method K-means and by introducing clustering features based on users previous booking history. Besides improving the accuracy of our predictions, we see that the K-means method manages to identify users which are more difficult to make predictions for, and place them in separate groups.

While our results show that grouping users together can lead to more accurate predictions of power consumption, they also highlight what could be done to increase this accuracy further. By helping and incentivizing users estimate distance more accurately, the performance of any model taking the estimated distance into account would be improved. It is clear that future work can make further advancements in the area and make electric car sharing services more attractive for a wider audience of customers.

References

Baak, M., Koopman, R., & Klous, S. (2018). A new correlation coefficient between categorical, ordinal and interval variables with pearson characteristics. *arXiv preprint arXiv:1811.11440*.

DeSarbo, W. S., & Cron, W. L. (1988). A maximum likelihood methodology for clusterwise linear regression. *Journal of classification*, *5*(2), 249–282.

Draper, N. R., & Smith, H. (1998a). Fitting a straight line by least squares. In *Applied regression analysis* (Vol. 326, pp. 33–34). John Wiley & Sons, Incorporated.

Draper, N. R., & Smith, H. (1998b). Fitting a straight line by least squares. In *Applied regression analysis* (Vol. 326, pp. 22–23). John Wiley & Sons, Incorporated.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (Vol. 96, pp. 226–231).

Ghosh, B., Asif, M. T., Dauwels, J., Cai, W., Guo, H., & Fastenrath, U. (2016, Nov). Predicting the duration of non-recurring road incidents by cluster-specific models. In *2016 ieee 19th international conference on intelligent transportation systems (ITSC)* (p. 1522-1527). doi: 10.1109/ITSC.2016.7795759

Grubwinkler, S., & Lienkamp, M. (2015). Energy prediction for EVs using support vector regression methods. In *Intelligent systems' 2014* (pp. 769–780). Springer.

Guo, Z.-C., Hu, T., & Shi, L. (2018, may). Gradient descent for robust kernel-based regression. *Inverse Problems*, *34*(6), 065009. Retrieved from `https://doi.org/10.1088%2F1361-6420%2Faabe55` doi: 10.1088/1361-6420/aabe55

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Boosting and additive trees. In *The elements of statistical learning: Data mining, inference, and prediction* (p. 349). New York, NY: Springer New York. Retrieved from `https://doi.org/10.1007/978-0-387-84858-7_10` doi: 10.1007/978-0-387-84858-7_10

Hox, J. J., Moerbeek, M., & Van de Schoot, R. (2017). Estimation and hypothesis testing in multilevel regression. In *Multilevel analysis: Techniques and applications* (p. 46). Routledge.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, *31*(8), 651–666.

Luo, Z., & Chou, E. (2006). Pavement condition prediction using clusterwise regression. *Transportation Research Record: Journal of the Transportation Research*

Board(1974), 70–77.

Mohamad, I. B., & Usman, D. (2013). Standardization and its effects on k-means clustering algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, *6*(17), 3299–3303.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). Polynomial regression models. In *Introduction to linear regression analysis* (Vol. 821). John Wiley & Sons.

Orr, G. B., & Müller, K.-R. (2003). Efficient backprop. In *Neural networks: tricks of the trade* (pp. 16–17). Springer.

Park, Y. W., Jiang, Y., Klabjan, D., & Williams, L. (2017). Algorithms for generalized clusterwise linear regression. *INFORMS Journal on Computing*, *29*(2), 301–317.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)*, *42*(3), 19.

Sen, A., & Srivastava, M. (2012a). Introduction. In *Regression analysis: theory, methods, and applications* (pp. 13–14). Springer Science & Business Media.

Sen, A., & Srivastava, M. (2012b). Introduction. In *Regression analysis: theory, methods, and applications* (p. 1). Springer Science & Business Media.

Sen, A., & Srivastava, M. (2012c). Multiple regression. In *Regression analysis: theory, methods, and applications* (pp. 30–31). Springer Science & Business Media.

Sen, A., & Srivastava, M. (2012d). Variable selection. In *Regression analysis: theory, methods, and applications* (p. 233). Springer Science & Business Media.

Späth, H. (1979). Algorithm 39 clusterwise linear regression. *Computing*, *22*(4), 367–373.

Theodoridis, S. (2015). Stochastic gradient descent: The LMS algorithm and its family. In *Machine learning* (pp. 162–167). Oxford: Academic Press.

Wang, J., Besselink, I., & Nijmeijer, H. (2018). Battery electric vehicle energy consumption prediction for a trip based on route information. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, *232*(11), 1528–1542.

Wu, X., Freese, D., Cabrera, A., & Kitch, W. A. (2015). Electric vehicles' energy consumption measurement and estimation. *Transportation Research Part D: Transport and Environment*, *34*, 52 - 67. Retrieved from `http://www.sciencedirect.com/science/article/pii/S1361920914001485` doi: https://doi.org/10.1016/j.trd.2014.10.007

Yan, X., & Su, X. (2009). Model selection. In *Linear regression analysis: Theory and computing* (p. 157). World Scientific.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(2), 301–320.

A

Booking contents file

A table containing the relevant contents of a booking.

Field	Description
activationCode	Booking id.
startTime	Start time of booking.
endTime	End time of booking.
costCenterId	Department of company who placed booking.
tripType	Company or private trip.
poolId	Pool where vehicle is picked up.
cityId	City where vehicle is picked up.
leaveAtId	Pool where vehicle is dropped off.
leaveAtCityId	City where vehicle is dropped off.
bookerId	The user who placed booking.
driverId	Id of driver.
debitAsVehicleTypeId	Vehicle (e.g. Ford Focus 2018).
totalDistance	Distance driven.
usageStart	Time of vehicle pickup.
usageEnd	Time of vehicle drop off.
bookingPerformedDate	Date when booking was placed.
companyRootId	The issuing company.
estimatedDistance	The user's own estimated distance.
vehicleId	Id of specific vehicle used.
endingType	Describes how the booking ended. Possible values are normal, canceled, not picked up and replaced by taxi.
batteryCapacity	Battery capacity of the vehicle used.

B

Heartbeat file

A table containing the contents of a heartbeat signal.

Field	Description
vehicleId	Specific vehicle used.
activationCode	Booking id.
timestamp	Recorded time of measurement.
powerLevel	Percentage of vehicle power remaining.
distanceToEmpty	Vehicle's own estimation of remaining driving distance at current power level.
charging	Boolean value showing if vehicle is currently charging.
latitude	Current latitude.
longitude	Current longitude.

C

Trip segments file

A table containing the contents of a trip segment.

Field	Description
activationCode	Booking id.
start	Timestamp when vehicle started driving.
stop	Timestamp when vehicle stopped driving.
tripDistance	Distance of trip segment.
odometerStart	Odometer value at start of trip.
odometerEnd	Odometer value at end of trip.