CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Predicting Vehicle Usage Using Incremental Machine Learning

How much energy can be saved by following the predictions?

Master's thesis in Computer Science and Engineering

TOBIAS LINDROTH

AXEL SVENSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Predicting Vehicle Usage Using Incremental Machine Learning

How much energy can be saved by following the predictions?

TOBIAS LINDROTH
AXEL SVENSSON

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

TOBIAS LINDROTH
AXEL SVENSSON

Supervisor: Niklas Åkerblom, Department of Computer Science and Engineering
Advisor: Mitra Pourabdollah, Volvo Cars
Examiner: Morteza Haghir Chehreghani, Department of Computer Science and Engineering

Master's Thesis 2022
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 2022

TOBIAS LINDROTH
AXEL SVENSSON
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Today, there is an ongoing transition to more sustainable transportation, and an essential part of this transition is the switch from combustion engine vehicles to battery electric vehicles (BEVs). BEVs have many advantages from a sustainable perspective, but issues such as limited driving range and long recharge times slows down the transition from combustion engines. One way to mitigate these issues is by performing battery thermal preconditioning, which increases the energy efficiency of the battery. However, to optimally perform battery thermal preconditioning, the vehicle usage pattern needs to be known, i.e., how and when the vehicle will be used.

This study attempts to predict the departure time and distance of upcoming drives using different incremental machine learning models. Further, the study includes a sensitivity analysis investigating how sensitive the performance of the battery thermal preconditioning is to incorrect predictions.

The problem of predicting departure time and trip distance is approached in two different ways. The first approach only considers the first drive each day, while the second approach considers all drives that directly follows a charging session. The incremental machine learning models are trained and evaluated on historical driving data collected from a fleet of BEVs. Additionally, the prediction models are extended to quantify the uncertainty of their predictions, which can be used as guidance to whether the prediction should be used or dismissed.

The best-performing prediction models yield an aggregated mean absolute error of 2.6 hours when predicting departure time and 12.5 km when predicting trip distance. However, for the considered temperatures and distances, the sensitivity analysis shows that the battery thermal preconditioning requires more precise predictions. The performance of the battery thermal preconditioning is sensitive, as the energy that can be saved from accurate predictions is less than what may be lost by adapting the preconditioning to incorrect predictions.

Keywords: Battery electric vehicles, Incremental machine learning, Uncertainty quantification, Sensitivity analysis, Time-to-leave, Trip distance.

Acknowledgements

We would first of all like to express our gratitude to our supervisors Mitra Pourabdollah and Niklas Åkerblom for their support and assistance throughout the project. We also wish to thank our examiner Morteza Haghir Chehreghani for his guidance and valuable advice.

Further, we wish to extend our gratitude to the people at Volvo Cars. Many thanks to Ole-Fredrik Dunderberg, Jimmy Forsman, Hannes Karlsson and Daniel Claesson for giving us a helping hand in different parts of our project.

Tobias Lindroth and Axel Svensson, Gothenburg, June 2022

Contents

List of Figures

List of Tables

Acronyms

ADAM Adaptive Moment Estimation.
ADWIN Adaptive Windowing.

BEV Battery Electric Vehicle.

FFNN Feed-Forward Neural Network.

KNN K-Nearest Neighbours.

LSTM Long Short-Term Memory.

MAE Mean Absolute Error.
MPIW Mean Prediction Interval Width.
MSE Mean Squared Error.

PHEV Plug-In Hybrid Vehicle.
PICP Prediction Interval Coverage Probability.

QARF Quantile Adaptive Random Forest.
QKNN Quantile K-Nearest Neighbours.
QR Quantile Regression.

RMSE Root Mean Squared Error.

SGD Stochastic Gradient Descent.
SoC State of Charge.

TD Trip Distance.
TTL Time To Leave.

VIF Variance Inflation Factor.

1

Introduction

There is currently an ongoing transition to more sustainable transportation. Recently, there have been several initiatives that aim to reduce emissions from vehicles and further accelerate the transition to zero-emission vehicles. The European Commission has, for example, proposed a 55 % reduction of emission from cars by 2030 and zero emission from new cars by 2035 [1]. Furthermore, several governments, cities, and automotive manufacturers have recently committed to work towards all sales of new vehicles being zero emission in the leading markets by 2035 and globally by 2040 [2]. In California, an executive order has been issued requiring every new sale of passenger vehicles to be zero emission by 2035 [3].

An essential part of switching to a more sustainable transportation system is battery electric vehicles (BEVs). BEVs are pure electric vehicles, as opposed to plug-in hybrid electric vehicles (PHEVs), and make up two-thirds of the rapidly growing global electric car stock [4]. However, even if BEVs have many advantages from a sustainable perspective, there exist other challenges. Compared to vehicles with a combustion engine, BEVs have a more limited driving range and longer recharge times. These disadvantages are exacerbated in cold and hot climates, but can in these conditions be mitigated by, for example, optimizing cabin preconditioning before departure, battery thermal preconditioning before fast charging, or battery thermal preconditioning before departure. Furthermore, the charging scheduling could be optimized as the time of day affects the cost and carbon dioxide footprint of the electricity production. To perform all these different kinds of optimizations, it is, however, needed to first learn about the vehicle usage pattern, i.e. how and when the vehicle will be used the next time. Some of this information, such as departure time and driving distance, can be given by the driver. Although, it might not be convenient for the driver to fill in information about the next drive, and it might not be accurate. Therefore, a better way to acquire the needed information may be to predict when and how the vehicle will be used the next time, using historical data of the vehicle usage together with machine learning.

Consequently, this study explores how accurately different machine learning models can predict vehicle usage patterns. Moreover, it investigates how sensitive the performance of battery thermal preconditioning before departure is to incorrect predictions. This study deals with data that arrives over time, and [5] argues that a specific type of machine learning called incremental learning fits well with this set-

ting. Incremental learning continuously updates the models as new data arrive over time and can function in environments with limited data storage and computational power. These characteristics could allow us to implement the prediction models on the BEVs directly, avoiding the need to transfer sensitive user data over the network.

1.1 Background

In certain climate conditions, the driving range of a BEV can decrease drastically since the energy of the battery might be needed for more tasks than just moving the vehicle forward [6]. Heating the cabin is an example of such a task. In a car with a combustion engine, waste heat from the engine can be used to heat the cabin when it is cold. In a BEV, the engine's efficiency is much higher, and less excessive heat is emitted. Therefore, energy has to be drawn from the battery to heat the cabin. Furthermore, the battery in a BEV only operates in a specific temperature range and the energy needed to cool or heat the battery is also drawn from the battery itself. Besides auxiliary climate loads, extreme climate conditions can also increase the internal resistance of the battery. For instance, in cold climate conditions, the auxiliary climate loads and extra increased internal resistance might decrease the possible driving range by over 50 % [7]. A possible way to mitigate the effect climate conditions have on the driving range of a BEV is by performing battery thermal preconditioning before the drive.

Today, the driver can initiate the thermal preconditioning of the car manually. When the car is charging, the driver can set a timer for climatization, a process in which both the cabin and battery are heated. Preconditioning the battery is beneficial for the vehicle's performance, energy consumption and charging in cold climates [8, 9]. Simulations from Volvo Cars indicates that the decrease in energy consumption achieved by preconditioning the battery is more prominent in long drives compared to short drives, see Figure 1.1. If the trip duration is short, then it is not as energy efficient to precondition the battery beforehand, as the grid energy used to heat the battery is somewhat wasted. If it had been known in advance that the next trip would be short, the battery preconditioning could have been skipped, saving both energy and battery life [8]. It is, therefore, helpful to be able to predict how and when the driver will use the car the next time in order to improve energy efficiency.

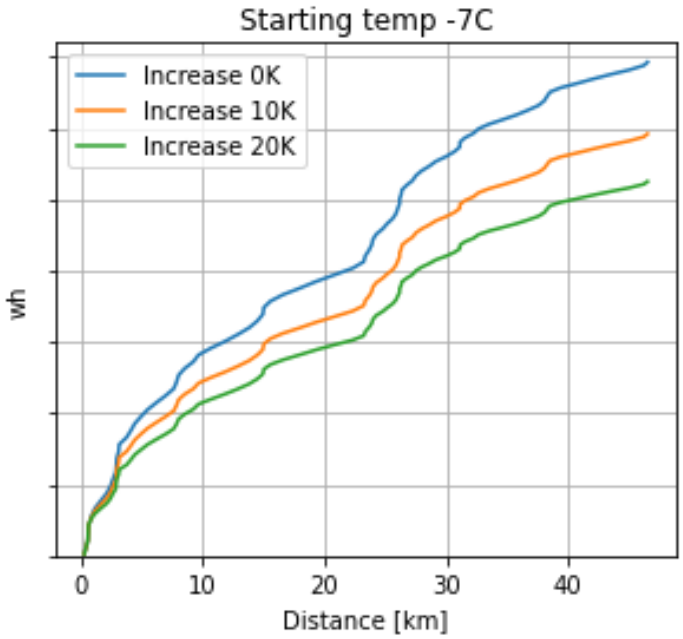


Figure 1.1: An example of how the total energy consumption can increases over time depending on the starting temperature of the battery. Note that the difference in energy usage is small when driving short distances and larger when driving longer distances.

On the other hand, if the predicted driving distance and departure time are wrong, this may also lead to unwanted consequences, such as dissipation of grid energy or insufficient preconditioning of the battery before a long drive. Therefore, another essential part of this problem is investigating how inaccurate predictions affect the battery thermal preconditioning. Such an analysis can provide a guideline for what prediction confidence is needed for the prediction to have a positive impact, and thus when the prediction should be used and when it should be dismissed.

1.2 Related Work

The problem considers predicting the departure time and distance of the next drive, and investigating how the energy consumption of this drive is affected by adapting the battery thermal preconditioning process to these predictions. This section introduces related articles and papers where similar forecasts have been performed but for various other contexts. To the best of our knowledge, no research has been made on predicting vehicle usage using incremental models or in the context of improving energy efficiency when performing battery thermal preconditioning.

1.2.1 Departure Time Prediction

Intuitively, the forecast for individual BEVs should depend highly on the driver, compared to e.g., busses which follow a fixed schedule. Considering an office commute, where flexible working hours are often applied, the arrival time to the office could vary significantly, which implicitly should affect the departure time from the

office as well. This type of correlation is shown in [10], where the authors predict the departure times from workplaces, to optimize smart charging. The authors point out that the average departure time of a BEV has a standard deviation of 141 minutes, showcasing the difficulty of this task. They investigate several different regression models on historical charging data and observe that sophisticated algorithms like XGBoost and artificial neural networks perform better than linear models, receiving a mean absolute error of 82 minutes. In [11] the authors predict the starting time and end time of an upcoming trip based on the start time, end time, and distance of the most previous trip. They receive a root mean square error (RMSE) of around 2 hours for both the start and end time. The authors further argue that simpler models such as KNN and decision trees outperform more complex neural-network-based models, even though the more complex models are better at extracting the travel behavior. Other approaches to departure forecasting predict time intervals instead of a specific time. For example, [12] forecasts a time interval of 15, resulting in an RMSE of roughly 3 hours.

The authors of [13] investigate the variation of departure times and attempt to model the first daily departure time as a probability distribution. They state that this is impossible after applying statistical tests measuring the similarity of well-known probability distributions.

1.2.2 Trip Distance Prediction

Trip distance is an important factor when deciding whether battery thermal preconditioning will be beneficial or not. Panahi et al. [14] predict the driving distance using artificial neural networks, resulting in a mean absolute deviation error of around 9%. Baghali et al. [11] implement a range of machine learning models, such as K-Nearest Neighbours (KNN), decision trees, random forest, and a range of neural networks, for predicting the daily travel and charging demand of electric vehicles. The novelty in their work is that they considered charging at more places than at home. They show that even the less complex models generate reasonable results, with an RMSE of around 19-23 km. However, they point out that to find daily temporal patterns, more complex models are needed.

As the authors of in 1.2.1 state, different individuals may have different driving patterns. The authors in [15] cluster PEV owners according to patterns in their driving behaviors. They train two Long Short-Term Memory (LSTM) recurrent neural networks on each cluster to forecast arrival time and travel distance and investigate the financial impacts of charging demand using these predictions. The authors show a significant improvement in forecasting the travel distance when the driving behavior is tied to a specific travel behavior pattern.

1.3 Aim

This study has two main goals. Firstly, investigate how well incremental machine learning models can predict an upcoming trip's departure time and distance. Sec-

ondly, examine how the energy consumption of a drive is affected when the battery thermal preconditioning is adapted to predictions of departure time and driving distance. To further concretize what this thesis aims to answer, the following research questions are defined:

- How accurate can incremental learning models predict the departure time and distance of an upcoming drive?
- How sensitive is the performance of the battery thermal preconditioning to incorrect predictions?

The first research question is answered by comparing several incremental machine learning models in terms of error rate. The latter question is investigated using dynamical models of the battery preconditioning process and energy consumption during a drive, and can indicate whether the incremental models are accurate enough for adapting the battery thermal preconditioning to their predictions. Several approaches for similar forecasts have been made, but none have used incremental learning models or analyzed how the predictions affect the battery thermal preconditioning.

1.4 Limitations

In this study, we focus on generating accurate predictions and investigating how the energy consumption of a drive is affected when an existing battery thermal preconditioning strategy is adapted to the predictions. We do not attempt to implement or optimize a battery thermal preconditioning strategy in this work, rather, we simulate an already existing strategy developed by Volvo Cars. Further, the investigation of how the energy consumption is affected by adapting this strategy to incorrect predictions only considers driving distances up to 47 km and ambient temperatures between -7 and 5C. The reason for this is that simulated data from Volvo Cars is used, and this data only contains the mentioned range of driving distances and ambient temperatures. Consequently, this study only focuses on heating the battery, not cooling the battery.

The datasets used in the project only contain data from Volvo cars driven by Volvo Cars employees. The potential conclusions in this report may therefore not hold for other car types than cars from Volvo and may not hold for users that do not drive with similar schedules. Moreover, the data was partly collected during the COVID-19 pandemic. The vehicle usage captured in the data may therefore not be representative of the vehicle usage before, or after, the pandemic, implying that some of the results of this study may not hold over time, although the prediction methods are still valid.

As mentioned in Section 1, using incremental learning has the benefit of potentially being implemented in the cars. However, this will not be considered in this project, as the aim is to investigate whether it is plausible to predict vehicle usage using incremental learning.

Finally, Shahriar et al. [16] show that external factors such as road conditions and local events may affect the driving behavior of a particular trip. However, this will not be considered as this type of information is not present in the datasets.

1.5 Ethical Concerns

The data studied in this project is user data. This data is pseudonymous, and all users have signed a contract that allows Volvo to gather and analyze this data according to the GDPR. However, the application is to predict a particular user's behavior, which can be considered sensitive from a privacy perspective. If an actual implementation of the prediction methods are to be done in the cars, one needs to consider privacy policies, according to GDPR.

Furthermore, there are some risks with inaccurate predictions when considering the user experience. If the car is insufficiently preconditioned, the driving range will decrease, exacerbating the range anxiety and potentially making the experience worse, especially when cabin heating is needed. Secondly, if the car is preconditioned too early, energy is unnecessarily lost to the surrounding environment, generating financial and ecological losses for the user.

2

Theory

This chapter introduces some concepts and algorithms applied in this work. Section 2.1 explains the concept of incremental machine learning. Section 2.2 describes the mechanisms of the machine learning models considered in this study and how they are adapted to an incremental setting. The concept of prediction uncertainty is introduced in Section 2.3 ,and Section 2.4 explains a way of evaluating incremental models and introduces the evaluation metrics used in this study.

2.1 Incremental Learning

The typical way of training a machine learning model is called offline learning. This type of learning requires a complete training dataset, and the models are trained once. If new data arrives in the dataset, it is often required that the model is retrained entirely with the extended dataset. Another way of training a machine learning model is called incremental learning, and this approach does not require a complete training dataset. Geng et al. [17] define incremental learning as a machine learning paradigm, where training observations arrive over time, and the training is performed sequentially as new observations arrive. Typical types of data that arrive over time are, for example, stock market data, signal data, and logging data. The common ground for all these data types is that each observation is marked with a timestamp, implying that each observation may depend on the preceding observation or any temporal factors such as time of day, weekday, or month.

There does not seem to be a clear consensus in the literature on what the definition of incremental learning is, and a paradigm named online learning is used in similar contexts [18, 19, 20, 5]. This is also confirmed by [5], who states that the definitions of the two paradigms seem to intertwine. The main consensus of the definitions is that both paradigms inherently try to learn under the assumption that data arrives over time. A further consensus is that a clear distinction between the paradigms is made in terms of how much memory the paradigm is limited to and whether training is done on, e.g., single observations or on sequences of past observations. However, which approach a paradigm should be referred to is ambiguously defined, and the amount of available data for each paradigm varies greatly.

To make these concepts clear, we refer to incremental learning as the concept of training sequentially on a single observation or a sequence of past observations,

while online learning is restricted to training on a single observation at a time.

One potential challenge with incremental learning is that it does not take the entire training set into account, which may demand a change of hyperparameters over time. Further, the distribution of the data might not be static, which is usually an assumption for most machine learning models [17, 18]. Section 2.1.1 describes the latter problem in more detail and a simple yet effective solution to this.

2.1.1 Change of Patterns over Time

A problem with some time-series data is that the relation between the features and the target variable may alter over time. There may be several reasons for this change, e.g., if one considers vehicle usage, aspects such as season, weather, the birth of a child, or a pandemic, could have huge impacts on the previously observed patterns. This change of pattern is known as a *Concept Drift* [18]. A concept drift will affect the performance of a machine learning model, and the model will need to re-adapt to the new data pattern.

Algorithms able to adapt to these concept drifts are often referred to as *Adaptive Algorithms* [17]. A common algorithm to use is *Adaptive Windowing* (ADWIN), introduced by Bifet et al. [21]. The idea of ADWIN is to investigate a sequence of recent data, which is referred to as a window, and look at whether two parts of the window have significant differences in their average error rate provided by, e.g., a regression model. One may deduce that a concept drift occurs when a change in the average error rate is significant enough, and thus the training should be adapted to the newer observations. In [21], Bifet et al. also introduce a strategy to find a potential concept drift by introducing two thresholds. The first threshold indicates a warning, and the algorithm starts training a new prediction model in the background using the new data. The second threshold indicates a drift detection, and the existing prediction model is replaced by the model trained when the warning appeared. Important to note is that this strategy assumes that the drift happens gradually. In this paper, the strategy was applied using a Naïve Bayes classifier. Moreover, this approach has also been used successfully using tree-based models [22, 23], which will be covered in Section 2.2.4.1.

2.2 Regression Models

Regression refers to the task of predicting a numerical value given some input values [24, p. 99]. In the following section, several regression models and their incremental implementations are presented in order of increasing complexity.

2.2.1 Statistical Time-Series Methods

Initially, it is good to investigate whether simple statistical methods can make predictions that suffice for the task at hand. These might also act as baseline models, which more complex models can be compared with. The simplest model is the his-

torical average, which considers all the past values and takes the average as a new prediction [25].

2.2.2 K-Nearest Neighbors

KNN is an algorithm that predicts the target of a new observation by using the K closest observations in the feature space [26, pp. 14–16]. A prediction $\hat{y}$ can, for example, be calculated by taking the mean of the closest K neighbors:

$$\hat{y} = \frac{1}{K} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} y_i \tag{2.1}$$

where $N_k(\mathbf{x})$ is the set of seen observations closest to the new observation, and y_i is the target value corresponding to observation $\mathbf{x}_i$ in the set $N_k(\mathbf{x})$. A typical way of measuring the distance between observations is to use the Euclidean distance in the feature space [26, pp. 14–16]. Assuming the newly observed value is $\mathbf{v} = [v_0, v_1, ..., v_n]$, then the euclidean distance d is calculated by

$$d = \sqrt{(w_0 - v_0)^2 + (w_1 - v_1)^2 + ...(w_n - v_n)^2} \tag{2.2}$$

where $\mathbf{w} = [w_0, w_1, ..., w_n] \in D$, and D is the set of all previously observed observations.

In the incremental setting, KNN may not have access to all past observations. Instead, one can save a fixed-size buffer of past observations and infer the prediction from this buffer.

2.2.3 Linear Regression

Linear regression is one of the simplest linear models for regression [27, pp. 138–143]. Given an observation $\mathbf{x} = (x_0, x_1, ..., x_d)^T$, one typically predicts a target variable y by creating a linear combination of the observations with a feature weight vector θ. The linear combination is denoted $\hat{y}$ and is calculated as

$$\hat{y} = \theta^T \mathbf{x} \tag{2.3}$$

The feature weight vector is typically calculated using the *least squares approach*, or *maximum likelihood estimation*, defined by

$$\theta = (X^T X)^{-1} X^T y \tag{2.4}$$

where X consists of all seen observations $\mathbf{x}$, meaning that all available training data is used to minimize the error between the linear combination and the true values. Important to note is that when using maximum likelihood estimation to estimate the weights, over-fitting might occur if the distribution of the training data is not representative of the underlying distribution. One solution is to add a regularization term (penalty [27, p. 10]) to Equation 2.4, resulting in

$$\theta = (\lambda \mathbf{I} + X^T X)^{-1} X^T y \tag{2.5}$$

where λ is the regularizing term and $\mathbf{I}$ is the identity matrix [27, pp. 144–146].

2.2.3.1 Incremental Linear Regression

In the incremental setting, data arrives continuously, and therefore it might not be possible to calculate the optimal feature weight vector as described in the previous section. However, by utilizing the fact that data arrive sequentially, one can instead update the weights incrementally as observations arrive [27, pp. 143–144]. A simple, yet extensively used technique for deep learning, is gradient based optimization. One of the most common algorithms for gradient based optimization is stochastic gradient descent (SGD), which is given by

$$\theta^{(i+1)} = \theta^{(i)} - \eta((y - \theta^{(i)T}\mathbf{x})\mathbf{x} + \theta^{(i)}\lambda) \tag{2.6}$$

where i is the ith update of the feature weights, η is the learning rate, λ is the regularizing term, $\theta^{(i)}\lambda$ is the gradient of the regularization term using the L2 norm, y is the current true target value and $\mathbf{x}$ is the current observation.

2.2.4 Random Forest

A popular approach for classification and regression tasks is tree-based ensemble models. The main idea of an ensemble model is to use multiple base learners to perform the regression or the classification and then take the majority vote, or the mean predicted value, from the learners. Random Forest is a tree-based ensemble model that uses classification and regression trees, also known as decision trees, as base learners. The idea behind tree-based models is to have a tree-like structure of nodes, where each node tries to split the feature space into multiple subspaces. The splitting is performed according to a boolean expression on a feature, to differentiate patterns in the data [26, pp. 305–307] by, e.g., minimizing the variance of the subspaces. A prediction is usually made by taking the average value of a specific subspace, which is retrieved by comparing the observed features with the nodes in the tree.

2.2.4.1 Incremental Adaptive Random Forest

The typical tree structures are not incremental by nature and require a static dataset to form a complete tree. However, a special kind of tree, called a Hoeffding tree [28], is specifically created to handle streaming data. A Hoeffding tree is created sequentially, effectively utilizing the earliest observations in the stream to select the splitting feature in the root of the tree and then using the subsequent observations to find the best features for the following nodes in the tree. The novelty of the Hoeffding tree is that it can determine, with a certain confidence, the number of observations needed to select the splitting feature which would also be selected if an infinite amount of observations were available.

In the regression setting, one way of selecting the feature to split at a specific node is the one reducing the variance in the target space the most [29, 30]. For instance, let feature x_a and feature x_b be the most promising features to split, and let $\bar{x}_a^{reduction}$ and $\bar{x}_b^{reduction}$ be the mean reduction in variance for the respective feature over a sequence of seen observations. If $\bar{x}_a^{reduction} - \bar{x}_b^{reduction} >= 0$, one can use the Hoeffding bound

to make sure that with a probability of $1-\delta$, the feature x_a is the best choice given that n observations have been seen. The Hoeffding bound states that the true mean of a variable is at least $\bar{x}_i^{reduction} - \epsilon$, where

$$\epsilon = \sqrt{\frac{R^2 \ln \frac{1}{\delta}}{2n}} \tag{2.7}$$

and δ is determined by the user in how confident the tree should be when determining the splitting feature, R is the range of possible values of $\bar{x}_i$ and n is the number of seen features. By using this, one simply needs to make sure that $\bar{x}_a^{reduction} - \bar{x}_b^{reduction} - \epsilon >= 0$ to determine that x_a is the best splitting feature.

The upside of the Hoeffding tree is that it potentially can handle infinite amounts of data while having a manageable computational cost, and further that it does not require any data to be stored after it has been processed. However, in the context of this project, a downside of the Hoeffding tree is that it assumes that observations in the stream are generated with a static probability distribution, i.e., that the distribution of the observation does not change over time. The data in this study will most likely not have a static probability distribution. Consequently, a method for handling a distributional shift, also called concept drift, is necessary. The authors of [22, 23] cope with this issue by using the ADWIN drift detection algorithm which is explained in Section 2.1.1.

An additional aspect of [22, 23] is that multiple trees are aggregated together, effectively creating an adaptive random forest of incremental trees. Here, one may either use ADWIN to either replace parts of the base learners, or to replace complete trees.

2.2.5 Feed-Forward Neural Network (FFNN)

A Feed-Forward Neural Network (FFNN) is the prime example of a deep machine learning model, and forms the foundation of many other types of neural networks [24, p. 164]. It consists of an input layer, a number of hidden layers and an output layer. Typically, the input layer consists of a feature vector x which is passed to the first hidden layer in the network. In this layer, a non-linear function is applied to an affine transformation of the input, specifically;

$$h_1 = g(W^T x + b) \tag{2.8}$$

where W and b are learnable parameters and $g(z)$ is a non-linear activation function, e.g. Rectified Linear Unit. The output from the first hidden layer, h_1, is passed to the next hidden layer in the network, and the same process is repeated until the output layer is reached. What is performed in the output layer is different for different applications. For instance, in regression, a simple linear transformation suffices. In classification, several activation functions exist for scoring each class, e.g., the sigmoid function for binary classification or the softmax function for multi-class classification. The learning is performed by comparing the predicted outcome with the true outcome via some loss function, and the learnable parameters are updated via the backpropagation algorithm.

As many other machine learning models, FFNN might overfit on the training data. A possible approach to ease the overfitting problem is to apply a technique known as dropout regularization during the training phase. This technique is performed by stochastically selecting neurons to disregard when producing a prediction, according to some probability p. Doing so results in a regularizing effect, almost forcing all of the hidden units to become good predictors individually, despite which other hidden units are available during inference [24, pp. 255–268].

2.2.5.1 Incremental Neural Network

A FFNN is in some sense incremental by nature since it updates its weights progressively using a gradient-based optimization algorithm such as SGD or Adaptive Moment Estimation (ADAM). Thus, one can update the parameters directly when a new observation is made, similar to what is described in 2.2.3.1.

2.3 Prediction Uncertainty

An important aspect of predictive modeling is quantifying how uncertain a model is in its prediction. For instance, if a prediction is used to make an important decision, it may be essential to know how confident the model is. Particularly, by using the uncertainty, one could refrain from making a prediction, which according to [31] is a sensible approach when handling anomalies or outliers in the data. To shortly introduce some terminology used by the machine learning community, it is common to differentiate different kinds of uncertainty. The two most distinct kinds of uncertainties are *aleatoric* and *epistemic* uncertainty [32, pp. 7–9], [31, 33, 34, 35, 36].

Aleatoric uncertainty corresponds to the uncertainty due to stochasticity in the data, such as noise. However, it could also be due to hidden variables that affect the prediction outcome (in the context of our work, a hidden variable could, for example, be information about upcoming meetings at work). On the other hand, epistemic uncertainty corresponds to the uncertainty due to the prediction model being inexperienced in certain regions of the data distribution.

The aleatoric uncertainty is often referred to as an irreducible uncertainty as it is not always possible to affect. In contrast, the epistemic uncertainty can be reduced by gathering more of the data with which the model is inexperienced. The aleatoric and epistemic uncertainties can be used separately or combined when estimating the confidence in the prediction. [32, pp. 127–131][33, 34].

One way to quantify the uncertainty of a prediction is to use prediction intervals. A prediction interval quantifies a potential range for where a future observation will occur with a certain probability. For instance, for a 95 % prediction interval, there is a 95 % probability that the next observation will fall within the prediction interval. Note that there is a direct correlation between the specified probability and the size of the prediction interval. The higher the probability, the broader the interval is likely to become.

A prediction interval may be defined as

$$[\hat{y} - z_{\frac{\tau}{2}}\sigma, \hat{y} + z_{\frac{\tau}{2}}\sigma] \tag{2.9}$$

where $\hat{y}$ is the model prediction, $z_{\frac{\alpha}{2}}$ is the $\frac{\alpha}{2}$ quantile of a standard normal distribution and σ is the prediction standard error [33].

In the following subsection, extensions to some of the mentioned prediction models presented in 2.2 will be introduced, which will be used to calculate a prediction interval for each prediction.

2.3.1 Quantile Regression

In linear regression, minimizing the square loss results in predicting the conditional mean response given an observation $\mathbf{x}$. If the absolute loss is minimized instead, namely $|y - f(x)|$, the conditional median response will be predicted instead [27, p. 48]. The median corresponds to the 50 percentile, and other quantiles may be calculated by using the tilted absolute value function, defined as

$$L_\tau(y, f(x)) = \begin{cases} (\tau - 1)(y - f(x)) & \text{if } y < f(x), \\ \tau(y - f(x)) & \text{if } y \geq f(x) \end{cases} \tag{2.10}$$

where τ is the chosen quantile [37, 38]. Thus, using this loss function when training yields an estimate of the respective quantiles directly, without estimating the standard error. Specifically, the 95% prediction interval can be estimated by specifying that $\tau = 0.025$ and $\tau = 0.975$, meaning that we find the 2.5^{th} and the 97.5^{th} quantiles.

Similarly to the incremental version of linear regression, the weights may be updated sequentially using gradient based optimization with respect to the tilted absolute value function. The update step is defined as:

$$\theta^{(i+1)} = \theta^{(i)} - \eta\nabla(L_\gamma(y, f(x)) + (\theta^{(i)T}\theta^{(i)})\lambda) \tag{2.11}$$

where θ is the model parameters, η is the learning rate and $\nabla(L_\gamma(y, f(x))+(\theta^{(i)T}\theta^{(i)})\lambda)$ is the gradient of the tilted absolute loss with added L2 regularization.

2.3.2 Quantile Adaptive Random Forest

Vasiloudis et al. proposes a quantile regression based approach called OnlineQRF to produce prediction intervals for incremental regression forests [39]. The main idea of their approach is to keep an approximated representation of the observed target values, from which it is possible to extract percentiles at wanted significance levels. To store the approximation in a memory bounded way, the authors use an incremental and mergeable data structure called *KLL sketch*. They let each leaf in the forest store a sketch, which keeps an approximation of the target values using a small memory space. When the forest receives a new training instance, each tree sorts the instance to a leaf, and updates the leaf's sketch with the target of the

Algorithm 1 The learning process of an incremental regression forest extended with OnlineQRF

Require: A training example (X, y)
  for $t \leftarrow Forest$ **do** //For each tree t in the forest
    l = Sort X into t //Traverse the tree to find the leaf
    Update l with (X, y)
    Update the sketch of l with y
  end for

training instance. A simplified pseudo code description of the learning process can be seen in Algorithm 1.

The prediction process makes use of the approximated target values to create a prediction interval. Each tree sorts the input to a leaf, and all the sketches of the reached leafs are merged. From the merged sketch it is possible to select the wanted quantiles. A description in pseudocode can be seen in Algorithm 2.

Algorithm 2 The prediction process of an incremental regression forest extended with OnlineQRF

Require: An unlabeled example X, the desired significance level α
  Initialize empty sketch H
  for $t \leftarrow Forest$ **do** //For each tree t in the forest
    l = Sort X into t //Traverse the tree to find the leaf
    Retrieve the sketch h of l
    Merge h into H
  end for
  return [Quantile α of H, Quantile $1 - \alpha$ of H]

OnlineQRF is a meta-algorithm and therefore it is possible to use any online tree based regression algorithm as the underlying learner, for example Adaptive Random Forest.

2.3.3 Feed-Forward Neural Network Uncertainty Estimation

Using a deep neural network, Zhu and Laptev present an approach for estimating a prediction interval on time series data [33]. Notably, they consider three uncertainties: the epistemic uncertainty, aleatoric uncertainty, and uncertainty due to model misspecification. These uncertainties are aggregated to form a combined measurement of the complete uncertainty, effectively estimating the standard error in Equation 2.9. The epistemic and misspecification uncertainties are jointly estimated, while the aleatoric uncertainty is estimated separately.

The epistemic uncertainty is estimated using a technique called *Monte Carlo dropout* introduced by Gal and Ghahramani [40]. In short, multiple predictions are per-

formed in a deep neural network using the same observation, where each prediction is individually affected by dropout. This means that some of the neurons in each layer are disregarded for each prediction, resulting in varying outputs. From a Bayesian perspective, the outputs can be seen as samples from the posterior predictive distribution. The authors prove that this posterior estimation approximates a Gaussian process, where the variance of the process corresponds to the uncertainty, approximated as

$$\hat{\sigma}_e^2 = \frac{1}{B}\sum_{b=1}^{B}(\hat{y}_b - \bar{y})^2 \tag{2.12}$$

where B is the number of predictions, and $\bar{y}$ is the mean target value.

The uncertainty from model misspecification may arise when the test data originates from a different distribution than the training set [33]. In order to capture this uncertainty, Zhu and Laptev propose to utilize the latent representation (the encoded feature vector) from an autoencoder architecture to measure the distance between the training and test data [33]. Specifically, the latent representation is extracted and used as input for another neural network performing the prediction. During the prediction, Monte Carlo dropout is utilized in both the encoding and prediction and simultaneously captures both the epistemic uncertainty and the model misspecification. The joint uncertainty is denoted $\hat{\sigma}_{e*}^2$.

In [33], the aleatoric uncertainty is estimated by using the residuals sum of squares, defined by

$$\hat{\sigma}_a^2 = \frac{1}{N_v}\sum_{n=1}^{N_v}(y_v - f(x_v))^2 \tag{2.13}$$

where the subscript v indicates that the data are from an independent validation set.

Finally, the total uncertainty is calculated by

$$\hat{\sigma}^2 = \sqrt{\hat{\sigma}_{e*}^2 + \hat{\sigma}_a^2} \tag{2.14}$$

which effectively represents the standard error in Equation 2.9.

2.4 Evaluation

This section describes how the incremental machine learning models are evaluated and what metrics are used.

2.4.1 Progressive Validation

The evaluation of an incremental learning model is different from the regular batch learning evaluation procedure. When evaluating a batch learning model, it is common to split the data into a training and test set, train the model on the training set, and test the model on the held-out test set. This approach assumes that the model is trained only once, and will make all future predictions based on this training.

This evaluation process does not reflect how an incremental model is used in production. An incremental model will keep learning all the time, and it is expected to make predictions at any time. An evaluation procedure called *progressive validation* fits better with how an incremental model is used in production and was introduced in [41]. In this evaluation procedure, the observations arrive in the order they happen. For each observation, the model is first given the unlabelled instance and attempts to make a prediction. Next, the model is given the labeled instance and attempts to learn from it. Progressive validation allows the model to be trained on all the data, and all the data is used as a validation set.

2.4.2 Mean Absolute Error (MAE)

The mean absolute error measures the error between each pair of observation and prediction. It is defined as

$$\frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \tag{2.15}$$

where n is the number of observations, $\hat{y}_i$ is the observation and y_i is the prediction.

2.4.3 Mean Squared Error (MSE)

The mean squared error is similar to the MAE but instead of measuring the average absolute error it averages the squared error. It is defined as

$$\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n} \tag{2.16}$$

where n is the number of observations, $\hat{y}_i$ is the observation and y_i is the prediction.

2.4.4 Root Mean Squared Error (RMSE)

The root mean squared error is a metric used in many of the related works mentioned in Section 1.2. It is simply the root of MSE and is thus defined by

$$\sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}}) \tag{2.17}$$

where n is the number of observations, $\hat{y}_i$ is the observation and y_i is the prediction.

The root mean squared error measures the standard deviation of the prediction errors and has the same unit as the predicted variable.

2.4.5 Evaluation of Prediction Intervals

A common approach to evaluate the quality of a prediction interval is to measure the Prediction Interval Coverage Probability (PICP) [34, 42]. As the name implies, it measures the percentage of observation correctly contained in the prediction interval, which is defined as

$$PICP = \frac{1}{n} \sum_{i=1}^{n} c_i \tag{2.18}$$

where n is total number of observations, while $c_i = 1$ if the prediction is in the prediction interval and 0 otherwise.

Another way to evaluate a prediction interval is to measure the Mean Prediction Interval Width (MPIW), defined as

$$MPIW = \frac{1}{n}\sum_{i=1}^{n}((\hat{y} + z_{\frac{\tau}{2}}\sigma) - (\hat{y} - z_{\frac{\tau}{2}}\sigma)) \tag{2.19}$$

where $\hat{y} + z_{\frac{\tau}{2}}\sigma$ and $\hat{y} - z_{\frac{\tau}{2}}\sigma$ defines the upper and lower bound of the prediction interval respectively.

2.5 Multicollinearity

In regression problems with multiple input features, there might be dependencies among the input features. Multicollinearity exists if any of these dependencies are strong [43]. Multicollinearity can be an issue for some regression models, making it difficult to determine the input features' individual effect on the target variable. The models might therefore learn untrue relationships.

One way of measuring the multicollinearity among features is the Variance Inflation Factor (VIF) [43]. The VIF score of a feature can be greater or equal to 1 and represents how well the other features can explain the feature. For a feature x, the VIF score is defined as:

$$VIF = \frac{1}{1 - R^2} \tag{2.20}$$

where R^2 is a measure of the correlation between x and the other features, calculated by regressing x on the other features. If there is a high correlation, R^2 will be close to 1, and the VIF score will therefore be large.

The lower the score, the less multicollinearity exists. There are different opinions regarding what score multicollinearity starts to be a problem [43]. Some authors consider ten as a reasonable threshold, while others suggest a threshold around four or five. This project will accept features with a VIF score below ten.

3

Methods

This chapter describes the methodology used in this study. Figure 3.1 displays an overview of different parts of the method and how they are connected. This chapter will begin with a description of how the prediction problem is modeled, and the rest of the chapter will describe the different parts of the method in more detail.

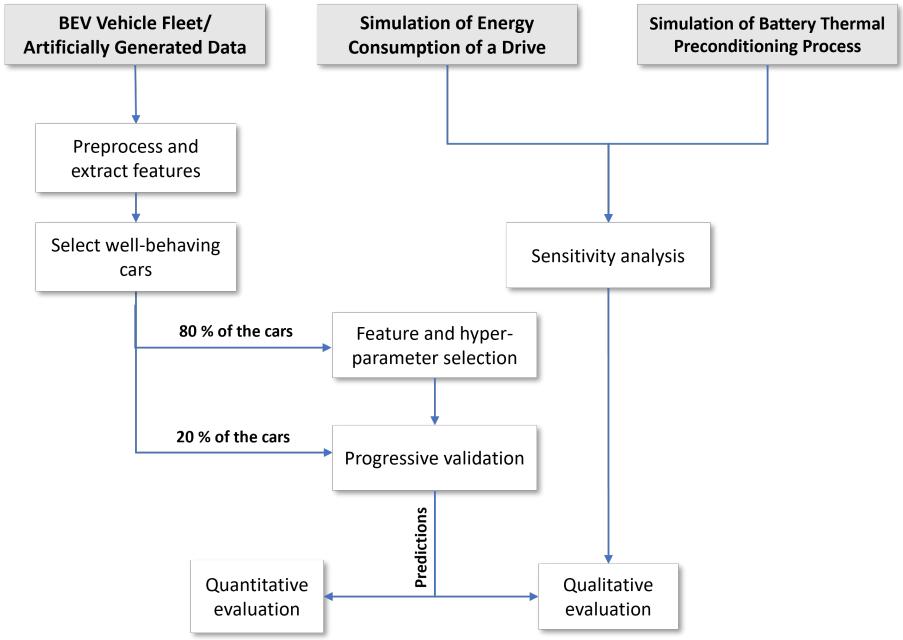


Figure 3.1: An overview of the method of this study.

3.1 Problem Setup

This study approaches the problem of predicting time of leave and trip distance in two different ways. The first approach only attempts to predict the time of leave and trip distance of the first drive each day. A prediction is performed at midnight, and it is assumed that it is known whether there will be a drive or not on a specific day. This approach is referred to as the midnight approach. The second approach is referred to as the charging approach and is slightly more advanced, as all drives which directly follow a charging session are considered. A prediction is performed when a charging session starts, and there is no assumption about on which day the

next drive is.

In both approaches, a prediction should answer the following questions:

- Time To Leave (TTL): What is the time, in minutes, to the start of the next trip from the moment the prediction is performed?
- Trip Distance (TD): What is the trip distance of the next trip?

The two variables TTL and TD are hereinafter referred to as target variables. The predictions of the target variables are based on historical data of previous driving and charging sessions.

3.2 Data Sources and Preprocessing

This study uses two alternative data sources to gather information about driving and charging sessions. The first source consists of data collected from a fleet of BEVs driven by Volvo Cars employees during a year. The second data source consists of artifically generated data, introduced to investigate how well simple and recurring vehicle usage patterns can be predicted. The data gathered from the fleet of cars will be the primary focus, while the generated data is used as supplementary information.

3.2.1 BEV Vehicle Fleet

The data from the BEVs consist of measurements of different attributes such as velocity, acceleration, the state of charge (SoC), and energy consumption. These attributes are measured at a constant rate with high frequency during measurement sessions and give a granular view of the car's properties. From these measurements, we extract summarizations of the driving and charging sessions. These summarizations consist of aggregations of the attributes, such as mean, standard deviation, maximum and minimum value.

3.2.1.1 Preprocessing

To increase the data quality, we apply several filters to the summarizations of driving and charging sessions. These filters are described in the bullet list below and handle inconsistencies such as missing data and abnormally short or long sessions.

- We are removing all driving and charging sessions shorter than 50 seconds. The selection of 50 seconds is arbitrary, though it removes most of the anomalous short trips and charging sessions.
- Two subsequent drive or charging sessions are merged if there are less than 15 minutes between them. This mainly affects the driving sessions, and we argue that this provides a better overview of the vehicle usage and that a 15-minute break does not influence the effect from preconditioning the battery.
- Observations that correspond to a TTL longer than 50 hours are disregarded.

This is because we consider a TTL larger than 50 hours to be too large, as we want to predict the trip that occurs within the near future.

- Cars with less than 50 drives are removed from the data to ensure there exists a minimum of trips for each car from which the machine learning algorithms can attempt to learn.

3.2.1.2 Features

As mentioned in 3.2.1, the predictions of TTL and TD are based on aggregations of different signals. Below, the signals considered for the prediction of the target variables are presented, with a short argument of why we consider them reasonable.

- **Start time of most recent charging session**. This is the feature which, in the charging approach, collaboratively decides the target variable *TTL*, together with the actual time of the next trip. In the charging approach, it is reasonable to assume that the start of a charging session might affect the time to the next drive. For example, if a charging session starts in the evening, it is likely that the next drive will not be until early on the following morning, as most people do not start driving in the middle of the night. In the midnight approach, the start of the most recent charging session might help provide information about the recent vehicle usage.

- **State of charge at the beginning of charging session**. The state of charge might indicate how the vehicle was used and will be used. For example, a low state of charge at the beginning of the charging session might indicate that the charging session will last for a while, and thus there might be a longer TTL compared to when there is a higher state of charge at the beginning of the charging session.

- **Start time, end time, and distance of the previous trip**. These metrics might help understand the recent vehicle usage and this might indicate what the next trip will be.

- **Ambient temperature and sun load during the previous trip**. The current weather could affect vehicle usage. A trip during the weekend is perhaps more likely if it seems to be warm and sunny outside. This data could be provided by a third party, however, in this study, we attempt to use information from the vehicle.

- **Speed and acceleration during the previous trip**. The average, standard deviation, min, and max of the speed and acceleration of the previous trip might be useful for modelling the vehicle usage pattern as it may indicate if it was a city or a highway drive or something in between. The acceleration might further indicate the driving behavior and possibly, if several people are sharing the car, who was driving.

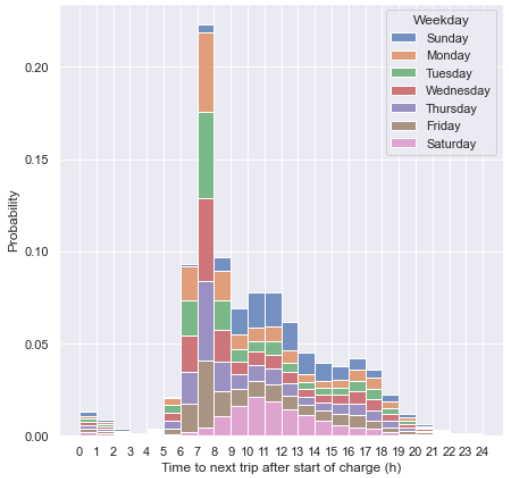
- **State of charge during the previous trip**. The change of state of charge during the last trip may indicate what kind of trip it was and help in predicting

the next trip.

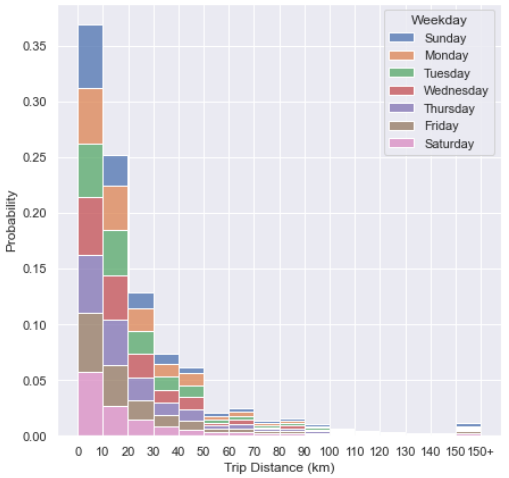
- **Date of when prediction is performed (only for midnight-approach)** It is reasonable to assume that the date of the prediction will affect when the first trip occurs. For example, it is reasonable that the first drive occurs earlier on a Monday than on a Saturday.

3.2.1.3 Analysis of Target Variables

In this section, we attempt to get an understanding of how the target variables TD and TTL differ when using the different approaches mentioned in Section 3.1. Figure 3.2 displays the distributions of the two target variables when using the midnight approach. The right sub-figure represents the TD distribution and indicates that the distribution is skewed towards zero with a long right-sided tail. A majority of the drives are shorter than 20 km, and there seems to be no significant difference depending on which day of the week it is. The left sub-figure displays the TTL distribution, i.e., the distribution over when the first drive of the day occurs. Between 7 a.m and 8 a.m, there is a peak, and the probability then decreases gradually by the hour. In contrast to TD, the day of the week has an evident effect on when the first drive of the day occurs, as the peak occurs later on weekends than during the week.



(a) Histogram estimating the distribution of TTL using the midnight approach. The hue displays the portions of observations corresponding to a specific day of the week for each bin of the histogram.



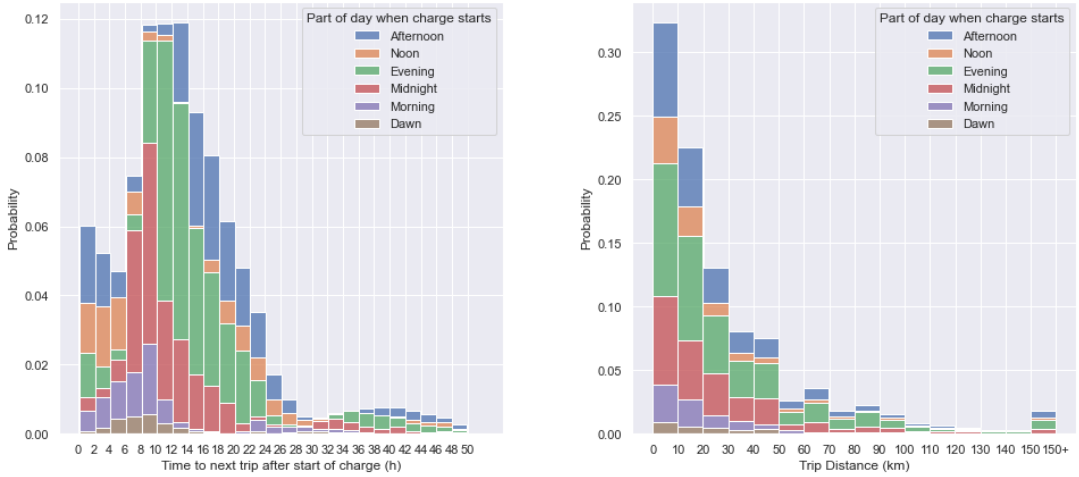
(b) Histogram estimating the distribution of TD using the midnight approach. The hue displays the portions of observations corresponding to a specific day of the week for each bin of the histogram.

Figure 3.2: Histogram over the target variables using the midnight approach.

Figure 3.3 displays the distributions of the target variables when the charging approach is applied. The TD distribution is similar to what it looked like in the midnight approach, and the only difference is that the shorter drives are slightly less frequent. The TTL distribution of the charging approach is quite different from the

midnight approach as it contains a much longer span of values, and it contains more values at the beginning of the range. In other words, the range of plausible values to predict is much larger in the charging approach.

It is important to note that the hue of the distributions has a different meaning in the charging approach. Instead of the day of the weak, the hue symbolizes the part of the day. This is because the day of the week has less impact on TTL than the part of the day that the charge started. The left sub-figure of Figure 3.3 shows that if a charge starts in the morning, the TTL will range from 0-10 hours, while if the charge starts in the evening, it will most likely range between 10-20 hours.



(a) Histogram estimating the distribution of TTL using the charging approach. The hue displays the portions of observation corresponding to a specific part of day for each bin of the histogram.

(b) Histogram estimating the distribution of TD using the charging approach. The hue displays the portions of observation corresponding to a specific part of day for each bin of the histogram.

Figure 3.3: Histogram over the target variables using the charging approach.

It is important to note that the graphs shown in Figures 3.2 and 3.3 are summaries of the joint behavior of all vehicles in the data. When investigating the target variables for single drivers, there are some similarities to the distributions shown above, e.g. that most of the cars perform trips shorter than 20 km, but the range of values is not necessarily as large, and the overall distribution is not necessarily similar to the shape of the distributions in Figures 3.2 and 3.3. As examples of distributions of individual cars, see Figures 3.4 and 3.5 in Section 3.2.1.4. Consequently, it seems beneficial to create individual prediction models per car and adjust them accordingly to one user's behavior.

3.2.1.4 Well-Behaving Cars

The analysis of the target variables in Section 3.2.1.3 indicates that the vehicle usage behavior varies between cars, potentially making the vehicle usage of some cars more difficult to predict compared to others. Inspired by Goebel and Voß [13], we want

to find cars that have a more regular driving and charging behavior, to disregard too noisy distributions. In their work, they predict the first daily departure time of commuter vehicles and develop a selection method for finding "well-behaving" cars. The goal of this selection method is to find cars that regularly travel to work and have similar departure times each day. They do this by selecting cars with a low standard deviation in departure time and number of weekdays a drive occurs.

In this project, we could utilize the same approach of selecting "well-behaving" cars. However, this method does not suit the charging approach, as the drive we try to predict may not necessarily be a commuting drive, nor the first drive of the day. Further, this selection method might favor cars with a similar departure time every day, and miss cars whose departure times vary per day but are the same across weeks. To also capture the weekly patterns, we want to find cars where similar observations of TTL and TD are frequently observed. Therefore, we propose a different selection method, where the clustering tendency is measured, i.e., how well the data can be clustered. A strong clustering tendency indicates that similar observations are frequently observed and that a recurrent vehicle usage pattern exists.

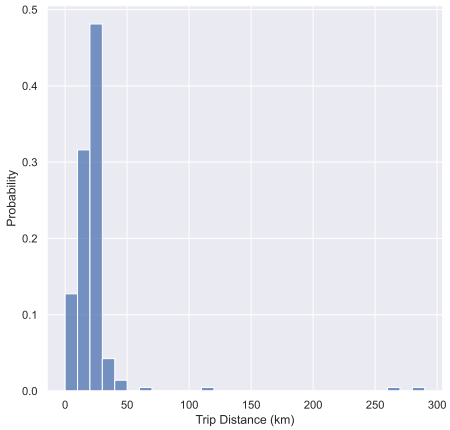
The clustering tendency is measured by a statistical test called the Hopkins test [44]. The test compares the joint distribution of the target variables with a two-dimensional uniform distribution by measuring distances between and within clusters. More specifically, for each sampled point in the joint distribution of target vectors, the Euclidean distance to its nearest neighbour is measured, as well as the distance to the nearest point in a uniform distribution of the same range as the target vectors. Let $X = \{x_0, x_1, ..., x_n\}$ denote the set of observations of the target variables, where n denotes the total number of observations. Further, define m as the number which is sampled from X, where $m << n$ and $Y = \{y_0, y_1, ..., y_m\}$, where Y is a vector of uniformly sampled values in the range of X. Next, define the distance from each point in Y to its nearest neighbour in X as $U = \{u_0, u_1, ..., u_m\}$, and the distance from each point in X to its nearest neighbour in X as $W = \{w_0, w_1, ..., w_m\}$, where the distance may be the Euclidean distance. The cluster tendency, denoted H, is then calculated by

$$H = \frac{\sum_{i=1}^{m} u_i^d}{\sum_{i=1}^{m} u_i^d + \sum_{i=1}^{m} w_i^d} \tag{3.1}$$

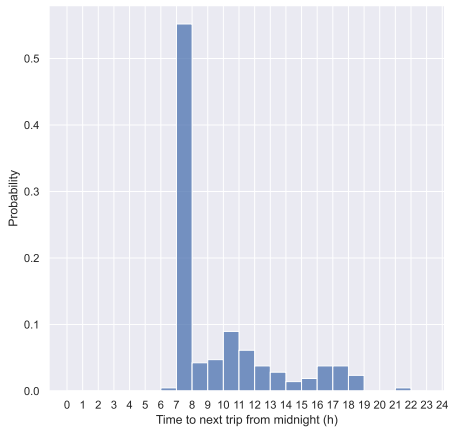
where H takes a value from $[0, 1]$, where 0.5 corresponds to being similar to an uniformly distributed vector while values close to 0 or 1 signals that the target vector is not uniformly distributed.

To exemplify our method, Figure 3.4 displays a car that our selection method deems "well-behaving", while Figure 3.5 shows a non-well-behaving car. In Figure 3.4b, it can be seen that the selection method successfully identifies a car having a frequently observed value of TTL. Further, our method is also able to deem that the car corresponding to Figure 3.5b is non-well-behaving since the distribution of TTL does not have a distinct frequent TTL. Notably, it is enough that one of the target variables is varying for a car to be labeled as non-well-behaving. This can be seen when comparing Figure 3.4a and 3.5a, where the car labeled as non-well-behaving has a more concentrated range of observed values of TD than the car labeled as

well-behaving.

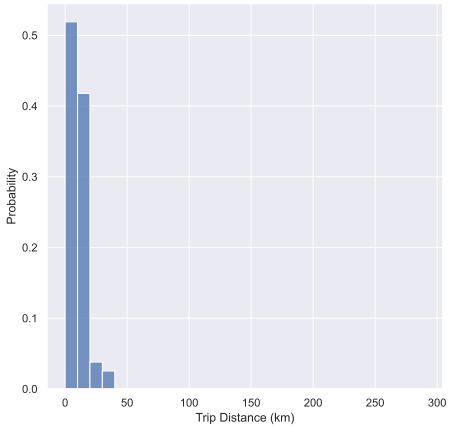


(a) Histogram estimating the distribution of TD for a specific car, deemed as "well-behaving".

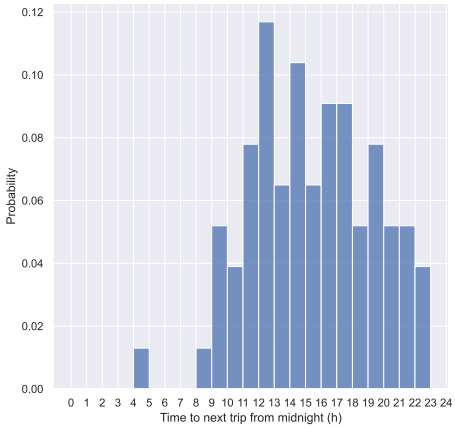


(b) Histogram estimating the distribution of TTL for a specific car, deemed as "well-behaving".

Figure 3.4: Histogram over the target variables of a specific car deemed as well-behaving.



(a) Histogram estimating the distribution of TD for a specific car, deemed as "variable".



(b) Histogram estimating the distribution of TTL for a specific car, deemed as "variable".

Figure 3.5: Histogram over the target variables of a specific car deemed as variable.

Each car is evaluated using the Hopkins statistic, and the 100 cars with the highest cluster tendency, i.e. most different from the two-dimensional uniform distribution, are selected to be used in the prediction process.

3.2.2 Generated Data

It is easy to assume that the vehicle usage of some people should be simple to predict, e.g. people who have very recurring schedules. However, early results in this project indicated that this assumption do not seem to hold. As a sanity check, we therefore introduce a second data source consisting of artificially generated data with simple and recurring vehicle usage, to investigate if that kind of vehicle usage can be predicted with high accuracy.

The data is generated by manually creating the features shown in Table 3.1. Some of the values are arbitrarily selected, while others are based on the mean value of the data from the vehicle fleet.

Table 3.1: The generated features for each behavioral pattern.

Generated Features
Start Time (YYYY-MM-DD hh:mm:ss)
Trip Distance
Average Speed
Average Acceleration
State of Charge
Ambient Temperature
Sun Load

To emulate real behaviors, uniformly distributed noise between the regular commuting trips is added to simulate extra trips that may occur randomly. Further, to investigate how well the machine learning algorithms learn different simple patterns, two distinct types of behavioral patterns are created and described in the following subsections.

3.2.2.1 Basic Pattern

The first pattern corresponds to a basic commuting pattern. Each workday, one trip is performed during the morning and one during the afternoon or evening. A charge session starts after the second trip has ended. All these trips have similar attributes, and the features presented in Table 3.1 are therefore sampled from Gaussian distributions. Specifications on how each feature is generated can be seen in Table 3.2.

Table 3.2: The features that are generated for the basic commuting trips, and the distributions or strategies for the generation.

Generated Feature	Sampling Distribution/Strategy
Start Time$_{\text{morning}}$	$\mathcal{N}(420, 30^2)$ (minutes)
Start Time$_{\text{afternoon}}$	$\mathcal{N}(960, 30^2)$ (minutes)
Trip Distance	$\mathcal{N}(17, 2^2)$ (kilometers)
Average speed	$\mathcal{N}(50, 2^2)$ (kilometers per hour)
Average acceleration	$\mathcal{N}(4, 1^2)$
State of Charge	$100 - \text{SoC}/\text{km} \cdot \text{Travelled Distance}_{\text{Since last charge}}$
Ambient Temperature	Arbitrarily set to match different temperatures for each month and part of day
Sun Load	$f(x; \mu_{data}, \sigma_{data}, 0, \text{inf})$ from Equation 3.2
Charge Start Time	$(\text{Start Time}_{\text{afternoon}}) + \frac{(\text{Trip Distance})}{(\text{Average speed})}$

Due to *Sun Load* being non-negative, a truncated Gaussian was used. Its probability density function is defined as

$$f(x; \mu, \sigma, a, b) = \begin{cases} \frac{\mathcal{N}(x;\mu,\sigma^2)}{\sigma(\Phi(b;\mu,\sigma^2)-\Phi(a;\mu,\sigma^2))} & \text{if a} \leq \text{x} \leq \text{b} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where $\mathcal{N}(x; \mu, \sigma^2)$ is the probability density function of a Gaussian distribution, $\Phi(x; \mu, \sigma^2)$ is the cumulative distribution function of a Gaussian distribution with mean μ and standard deviation σ. The distribution has the usual properties of a Gaussian distribution but in a closed interval.

3.2.2.2 Distinct Pattern

To generate a more complex pattern, we add different types of trips to the basic pattern. Firstly, the sampling strategy for the starts of the drives and charges during Mondays, Tuesdays, Thursdays, and Fridays described in the basic pattern is kept, while the trip distance is slightly longer. On Wednesdays, the starting time for a trip during the morning is later, and the trip in the afternoon is earlier, while all other features are the same as the other commuting trips. The specifications of each generated feature are displayed in Table 3.3.

Table 3.3: The features that are generated for the commuting trips, and the distributions or strategies for the generation. Note how Wednesdays have different starting times compared to the other weekdays, which is to challenge the models in detecting different patterns in departure time depending on which day it is.

Generated Feature	Sampling Distribution/Strategy
Start Time$_{\text{morning}}$	$\mathcal{N}(420, 30^2)$ (minutes)
Start Time$_{\text{afternoon}}$	$\mathcal{N}(960, 30^2)$ (minutes)
Start Time$_{\text{Wednesday morning}}$	$\mathcal{N}(600, 30^2)$ (minutes)
Start Time$_{\text{Wednesday afternoon}}$	$\mathcal{N}(720, 30^2)$ (minutes)
Trip Distance	$\mathcal{N}(30, 2^2)$ (kilometers)
Median speed	$\mathcal{N}(50, 2^2)$ (kilometers per hour)
State of Charge	$100 - \text{SoC}/\text{km} \cdot \text{Travelled Distance}_{\text{Since last charge}}$
Maximum acceleration	$\mathcal{N}(4, 1^2)$
Ambient Temperature	Arbitrarily set to match different temperatures for each month and part of day
Sun Load	$f(x; \mu_{data}, \sigma_{data}, 0, \text{inf})$ from Equation 3.2
Charge Start Time	$(\text{Start Time}_{\text{afternoon}\mid\text{Wednesday afternoon}}) + \frac{(\text{Trip Distance})}{(\text{Average speed})}$

Further, we add longer trips on Friday and Sunday evenings to introduce varying trip distances and trips during weekends that are not due to the noise generation. To avoid overlapping with the regular commuting trips generated on Friday afternoon, the departure times of the longer trips on Fridays are sampled from a truncated Gaussian (described in Equation 3.2) with an offset from the ending of the regular trip, making sure that these trips happen consecutively. Further on, the departure times of the trip on Sundays are sampled from a Gaussian distribution. The detailed descriptions of the generation of each feature are covered in Table 3.4.

Table 3.4: The features that are generated for the longer trips on Fridays and Sundays, and the distributions and strategies for the generation

Generated Feature	Sampling Distribution/Strategy
Start Time$_{\text{Friday}}$	$(\text{Start Time}_{\text{Friday afternoon}}) + f(x; 60, 30, \frac{0-60}{30}, \text{inf})$ (minutes)
Start Time$_{\text{Sunday}}$	$\mathcal{N}(1080, 30^2)$ (minutes)
Trip Distance	$\mathcal{N}(17, 2^2)$ (kilometers)
Average speed	$\mathcal{N}(50, 2^2)$ (kilometers per hour)
State of Charge	$100 - \text{SoC}/\text{km} * \text{Travelled Distance}_{\text{Since last charge}}$
Ambient Temperature	Arbitrarily set to match different temperatures for each month and part of day
Sun Load	$f(x; \mu_{data}, \sigma_{data}, 0, \text{inf})$ from Equation 3.2
Charge Start Time	$(\text{Start Time}_{\text{Friday}\mid\text{Sunday}}) + \frac{(\text{Trip Distance})}{(\text{Average speed})}$

3.2.2.3 Distributions of Target Variables

To exemplify the generation, the data generation processes in Sections 3.2.2.1 and 3.2.2.2 are performed to generate time periods of one year with minimal noise, resulting in two different distributions for the target variables. Considering a drive and a charge as events, the distributions over starting time on the time of day for these events are displayed in Figure 3.6 for each pattern, while the corresponding

trip distances for each driving event are displayed in Figure 3.7. In Figure 3.6, the basic generation pattern has two modes, i.e., peaks in the distribution where values occur most frequently, of starting times for a drive, while there is a single mode for when a charge occurs. The more complex pattern, also has two distinct modes when a drive occurs, while there are three modes for when a charge occurs. In Figure 3.7, there exists a major peak for the basic pattern, while the complex pattern has two modes in its distribution.

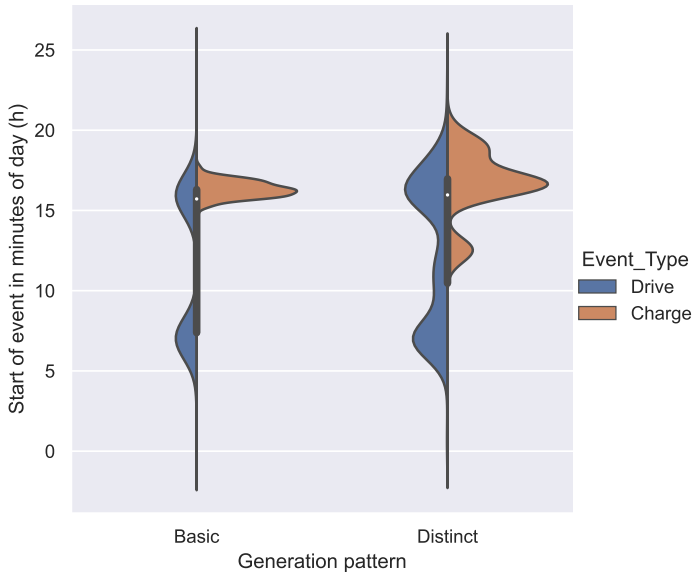


Figure 3.6: Violin plot of the generated start of an event (charge or drive) for the two generation strategies described in Sections 3.2.2.1 and 3.2.2.2. The generation patterns describe which strategy is used. The time of the event is given in the time of a day in hours.

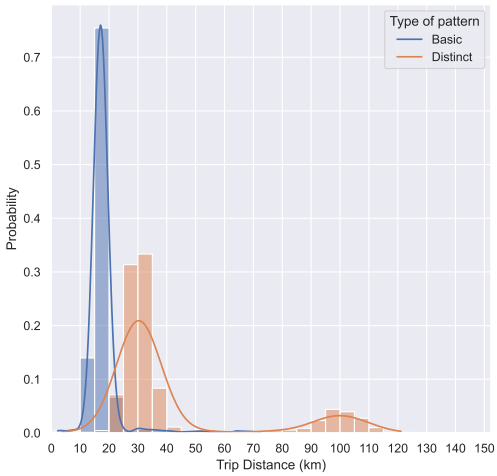
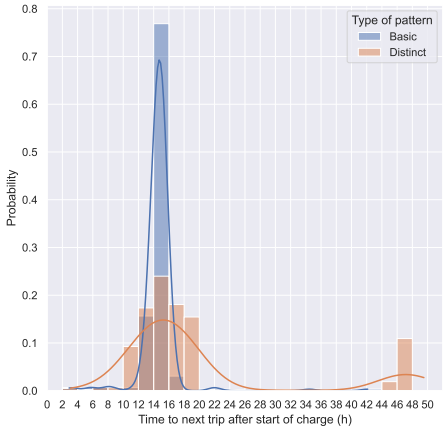
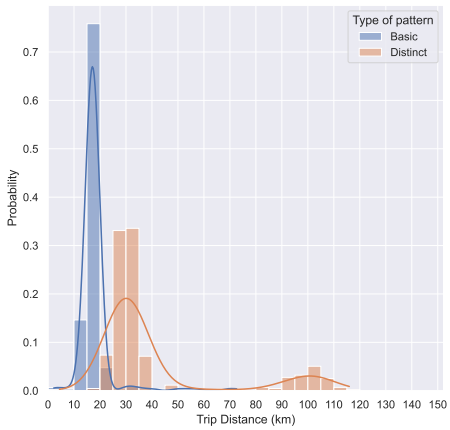


Figure 3.7: Histogram and its corresponding kernel density estimation over the generated trip distance for trips performed over a year with minimal noise for the two generation strategies described in Sections 3.2.2.1 and 3.2.2.2.

The generated data is processed in a similar way as described in Section 3.2.1.1, resulting in two distributions corresponding to TTL and TD, respectively. The two approaches mentioned in Section 3.1 yield slightly different target distributions, where the target distributions for the two generated patterns are displayed in Figure 3.8 for the charging approach, while the respective distributions for the midnight approach are displayed in Figure 3.9.

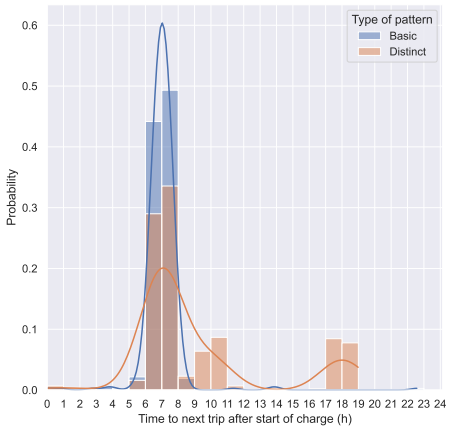


(a) Histogram and its corresponding kernel density estimation over the generated TTL after a charge.

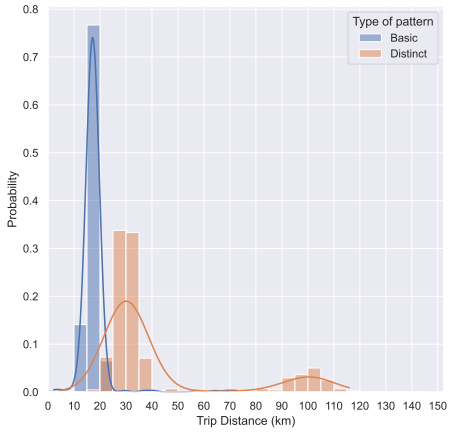


(b) Histogram and its corresponding kernel density estimation over the generated TD after a charge.

Figure 3.8: Histogram and its corresponding kernel density estimation over the generated target variables after a charge. Trips are performed over a year with minimal noise for the two generation strategies described in Sections 3.2.2.1 and 3.2.2.2.



(a) Histogram and its corresponding kernel density estimation over the generated TTL using the midnight approach.



(b) Histogram and its corresponding kernel density estimation over the generated TD using the midnight approach.

Figure 3.9: Histogram and its corresponding kernel density estimation over the generated target variables using the midnight approach. Trips are performed over a year with minimal noise for the two generation strategies described in Sections 3.2.2.1 and 3.2.2.2.

3.2.3 Feature Engineering

Feature engineering is the process of transforming data into features useful for machine learning models by using human knowledge. In this study, we apply feature engineering to extract different features from the datetime fields mentioned in Section 3.2.1.2. For example, the minute of the hour, the hour of the day, the part of the day (morning, noon, afternoon, etc), the day of the month, the day of the week, whether it is a workday or not, similar to [10]. As in [16], cyclic representations of some of the temporal features are added to include information such that 11 PM and 1 AM are closer than 1 AM and 6 AM. This is done by performing the trigonometric transformations

$$f_x = sin(2\pi f/max(f)) \tag{3.3}$$

$$f_y = cos(2\pi f/max(f)) \tag{3.4}$$

where f is the feature to be transformed and f_x and f_y are the two components of the cyclic feature.

One-hot encodings are added for the categorical features describing the day of the week and the part of the day; in other words, a new binary feature is added for each unique value. Further, similar to [16, 10], historical averages are used. For example, the historical average of TTL, the historical average of TD, and more specific

averages, such as the historical average of TTL and TD for previous predictions performed on the same day of the week and the same part of the day. These features are created by, for each car, averaging the previous target variables. For example, the historical average of TTL for a charging session starting in the evening is calculated by taking the average TTL of all earlier observations where the charging session started in the evening. If there were no earlier observations, the historical average is set to 0. As historical averages are based on all previous values, rolling averages are also used to catch more recent changes. These are calculated by averaging the target variables of the last five observations.

All features considered for the predictions can be seen in Table 3.5.

Table 3.5: All the features considered for the predictions

Feature	Description
Midnight Info (Only midnight approach)	
Midnight_Year	Numerical representation of year
Midnight_Month	Numerical representation of month
Midnight_Day_Of_Week	Day of the week, one-hot encoded
Midnight_Day	Numerical representation and Sine and Cosine components of the day
Midnight_Workday	Binary variable indicating whether it is a workday
Charging Info	
Charge_Start_Time_Year (Only charging approach)	Numerical representation of year
Charge_Start_Time_Month (Only charging approach)	Numerical representation of month
Charge_Start_Time_Day_Of_Week	Day of the week, one-hot encoded
Charge_Start_Time_Day	Numerical representation and Sine and Cosine components of the day
Charge_Start_Time_Part_Of_Day	Part of day (e.g. evening). one-hot encoded
Charge_Start_Time_Hour	Numerical representation and Sine and Cosine components of the hour
Charge_Start_Time_Minute	Numerical representation of the minute
Charge_Start_Time_In_Minutes	Numerical representation and Sine and Cosine components of the time of day in minutes
Charge_Start_Time_Workday	Binary variable indicating whether it is a workday
Charge_SoC_Init	The initial state of charge
Previous Trip	
End_Time_Part_Of_Day	Part of day (e.g. evening). one-hot encoded
End_Time_Hour	Numerical representation and Sine and Cosine components of the hour
End_Time_Minute	Numerical representation of the minute
End_Time_In_Minutes	The time of day in minutes
Start_Time_Day_Of_Week	Day of the week, one-hot encoded
Start_Time_Day	Numerical representation and Sine and Cosine components of the day
Start_Time_Part_Of_Day	Part of day (e.g. evening). one-hot encoded
Start_Time_Hour	Numerical representation and Sine and Cosine components of the hour
Start_Time_Minute	Numerical representation of the minute
Start_Time_In_Minutes	The time of day in minutes
Start_Time_Workday	Binary variable indicating whether it was a workday
Speed	The mean, median, standard deviation, min and max of the speed
Acc	The mean, median, standard deviation, min and max of the acceleration
Amb	The inital, end, mean, median, standard deviation, min and max of the ambient temperature
Sun_Load	The mean, standard deviation, min and max of the sun load
SoC	The initial, end and change of state of charge.
Trip_Dist	The trip distance (km)
Duration	The duration of the trip (min)
Averages	
Historical_Avg	The average TTL and TD from all earlier observations
Historical_Avg_Same_Day	The average TTL and TD from all earlier observations with prediction time (midnight or start of charge) at same day of week
Hist_Avg_Same_Day_And_Time (Only charging approach)	The average TTL and TD from all earlier observations with charge starting time at same day and same part of day
Rolling_Avg	The average TTL and TD from the previous five observations
Rolling_Avg_Same_Day	The average TTL and TD from the previous five observations with prediction time (midnight or start of charge) at same day of week
Rolling_Avg_Same_Day_And_Time (Only charging approach)	The average TTL and TD from the previous five observations with charge starting time at same part of day

3.3 Prediction Models

This section describes the different prediction models that we consider in this study. A preprocessing step common for all models, except the baseline, is that all numeric features are scaled before being inputted into the model. The data is scaled so that each numeric feature has zero mean and unit variance. This is, just like the learning process, done incrementally by keeping track of a running mean and running

variance. For Quantile Regression and the Feed-Forward Neural Network, we also scale the target in the same way as this significantly increases the performance of the models.

3.3.1 Baseline

For both prediction problems, TD and TTL, we use a model that always predicts the historical average as a baseline. For each observed target, the model updates the average target value and uses this average in the next prediction. For example, the fifth prediction is the average of the four previously observed targets.

3.3.2 Quantile Regression (QR)

In this study, we use an incremental version of quantile regression from a package called *River* [1], which is a Python package for online machine learning. Three quantile regression models are used, each being a linear regression model trained using a quantile loss as described in Section 2.3.1, where the 5^{th}, 50^{th} and 95^{th} quantile are predicted respectively. The 50^{th} quantile corresponds to the median of the prediction interval and is chosen as the primary prediction, while the others are used to define a 90 % prediction interval. The hyper-parameters considered for tuning are the learning rate and the amount of l2 regularization.

3.3.3 Quantile K-Nearest Neighbours (QKNN)

K-Nearest Neighbors Regression is a model that makes predictions by clustering the previous observations and aggregating the target values of the closest neighbors. The incremental KNN regression algorithm used in this study comes from the *River* [2] package. To quantify the uncertainty, we have expanded the algorithm by estimating the distribution of the target variables corresponding to the K nearest neighbors as a Gaussian distribution [45]. From this distribution, the 50^{th} quantile is used as the prediction, while the 5^{th} and the 95^{th} quantiles are used to estimate the uncertainty. The hyper-parameters considered for tuning are the window size, i.e., how many previous observations should be stored, and the number of neighbors on to base the predictions.

3.3.4 Quantile Adaptive Random Forest (QARF)

Random Forest is a commonly used machine learning model. In this study, an incremental version called *Adaptive Random Forest Regressor* provided in the *River* package [3] is used. We have expanded the model to also estimate prediction intervals by using the quantile regression approach described in 2.3.2. The hyperparameters considered for tuning are the number of trees and whether the leaves should base their predictions on the target mean or use Linear Regression.

[1] `https://riverml.xyz/latest/api/linear-model/LinearRegression/`
[2] `https://riverml.xyz/latest/api/neighbors/KNNRegressor/`
[3] `https://riverml.xyz/latest/api/ensemble/AdaptiveRandomForestRegressor/`

3.3.5 Feed-Forward Neural Network with Uncertainty Quantification (FFNN)

In this thesis, we implement FFNN using the *PyTorch* [4] package. A single observation is used to update the model parameters via the ADAM optimizer. To quantify the uncertainty, we use Monte Carlo dropout to estimate the epistemic uncertainty and the residual sum of squares of the 10 latest predictions to estimate the aleatoric uncertainty. Note that the model misspecification described in Section 2.3.3 is disregarded. However, the epistemic and aleatoric uncertainty are still aggregated as described in Equation 2.14 to quantify the variance of the error. The hyperparameters considered for tuning are the learning rate, the number of hidden layers, the number of neurons in the hidden layers, and the dropout rate.

3.4 Prediction Process

As mentioned in Section 1, this study uses an incremental learning approach, as this fits well with the temporal nature of the data, where new observations become available over time. The incremental learning approach enables continuous learning over time and avoids the need to retrain the model when new samples become available. However, a negative aspect of this approach is the lack of a standardized way of performing feature selection and hyperparameter optimization [46]. The authors of [5] present two approaches for hyperparameter optimization. The first approach is called the offline setting, and refers to using a complete training set to find the hyperparameters, much like the traditional machine learning paradigm, and then use the optimal hyperparameters to validate the models on a test set. The second approach, called the online setting, saves the first 20% (or the first 1000) observations in a buffer to perform hyperparameter tuning on. The second approach does, however, make the assumptions that the selection of these hyperparameters is optimal throughout the rest of the stream and that concept drifts do not occur [46].

In addition to incremental learning, this study also uses personalized models, i.e., there is one model instance per car. This setup makes feature and hyperparameter selection more problematic as it might differ from car to car what features and parameters yield the best result. We simplify the selection of hyperparameters and features for each machine learning model by selecting the parameters and features that yield the lowest average MAE for all well-behaving cars. The selection method is similar to the first approach described in the paragraph above. That is, feature selection and hyperparameter tuning for a model are performed on a subset of the well-behaving cars. The selected features and hyperparameters are then used on the remaining well-behaving cars to validate the models. An overview of the process can be seen in Figure 3.10. It is important to note that this approach does not attempt to select the best features and hyperparameters for each car individually, but rather it attempts to select the features and hyperparameters that work best overall.

[4] `https://pytorch.org/`

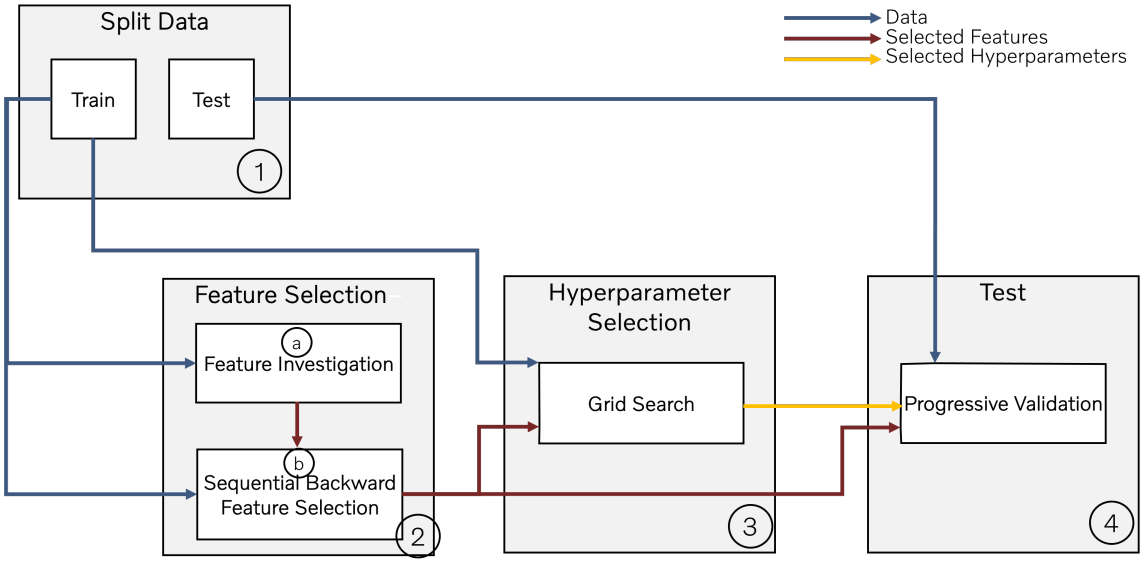


Figure 3.10: The figure shows how the feature selection, hyperparameter selection, and evaluation process are connected. In 1), the data is split into two subsets. In 2) we select what features to use for each model. The feature selection is made by a) performing a feature investigation to see what features seem helpful for the predictions and b): performing a sequential backward feature selection for each model to find a good subset of the selected features in a). In 3), the hyperparameters for each model are selected using grid search. In 4), The models are evaluated on the held-out data using the selected features from 2) and the selected hyperparameters from 3).

3.4.1 Feature Selection

The feature engineering described in Section 3.2.3 generates a large number of features. Reducing the number of features is often desired as it reduces the computational cost, and may improve the performance of the model. The feature selection process of this thesis consists of two steps. First, we perform a general feature selection to reduce the number of features by removing

1. features that do not seem to be helpful when predicting the target variables
2. features that have high correlation with other features

The second step in the feature selection process is to select which subset of features that produces the best result for each model.

3.4.1.1 General Feature Selection

The general feature selection consists of three different feature selection techniques.

The first technique is Pearson's correlation coefficient, which measures the linear correlation between two variables. A coefficient close to -1 or 1 indicates a linear relationship, while a coefficient close to 0 indicates an absence of a linear relationship. As the correlation between a feature and the target variable might be different for different cars, the correlations between the variables are computed for each customer separately and then averaged. Further, as it is only the existence of a linear relationship that is important, rather than the direction, the absolute value of the

coefficient is used.

Next, a forward sequential feature selection is performed to understand how well the different features perform in machine learning models. This greedy algorithm attempts to find the subset of features that yields the best score when input into an existing model. At each iteration, the algorithm selects the feature that improves the MAE the most until a pre-defined number of features has been selected, in our case 10. Batch-learning is used to train the models, while time-series cross-validation, described in [47, Chapter 3.4], is used to validate the models.

As the importance of each feature might be different for each car, we perform the forward sequential feature selection on each car individually. The importance of each feature is then corresponding to the percentage of cars where it is picked among the ten selected features. Further, the usefulness of a feature might not only differ between cars but also between model types. Therefore, the forward sequential feature selection is performed using two different models: Linear Regression and Random Forest.

Based on the scores from the correlation measure and the forward sequential feature selection, some features are removed. As a guideline, we discard any features that do not score equal to or higher than the 70th percentile in any of the tests. However, this is not a strict rule. If there is a coherent group of features from which one or two features are selected to be removed, this may be disregarded. For example, if the cosine part of a feature is removed while the sine part is selected, the cosine part can be included as a selected feature anyway.

The final test in the general feature selection attempts to remove multicollinearity. High multicollinearity among the features might indicate that there are many redundant features. To avoid this, the method described in [48] is used to calculate a subset of the numeric features with low VIF scores. The method is summarized in the following bullet list.

1. Calculate the VIF for all numeric features
2. If there exists features with a VIF greater than 10, remove the feature with the highest VIF.
3. Repeat step 1 and 2 until there are no features with a VIF higher than 10.

The resulting subset of features, after removing features that do not seem to be helpful when predicting the target variables or have high correlation with other features, can be seen in Table 3.6.

Table 3.6: Table displaying the remaining features after the general feature selection

Charging Approach TD	Charging Approach TTL	Midnight Approach TD	Midnight Approach TTL
Charge_Start_Time_Month	Charge_Start_Time_Month	Date_Midnight_Month	Date_Midnight_Month
Charge_Start_Time_Hour_Sin	Charge_Start_Time_Hour	Date_Midnight_Day_Cos	Charge_Start_Time_Day
Charge_Start_Time_Minute_of_Day_Cos	Charge_Start_Time_Hour_Sin	Date_Midnight_Day_Sin	Charge_Start_Time_Minute
Charge_Start_Time_Minute	Charge_Start_Time_Minute_of_Day_Cos	Charge_Start_Time_Day_Sin	Charge_SoC_Init
End_Time_Hour_Sin	Start_Time_Hour_Cos	Charge_Start_Time_Day_Cos	Start_Time_Hour_Cos
Start_Time_Hour_Cos	End_Time_Hour_Sin	Charge_Start_Time_Hour_Sin	End_Time_Hour_Sin
Charge_SoC_Init	Charge_SoC_Init	Charge_Start_Time_Minute_Cos	Acc_Min
Sun_Load_Average	Sun_Load_Min	Charge_SoC_Init	Amb_Temp_Std
Sun_Load_Std	Amb_Temp_Std	Start_Time_Hour_Sin	Amb_Temp_Min
Sun_Load_Min	Acc_Min	Start_Time_Hour_Cos	Sun_Load_Average
Amb_Temp_End	Spd_Avg	Spd_Median	Sun_Load_Std
Amb_Temp_Std	TripDist	Acc_Stdev	Sun_Load_Min
Acc_Stdev	RollingAvgTimeToDriveSameDaySameTime	Amb_Temp_Std	Duration
Spd_Median	RollingAvgTimeToDrive	Amb_Temp_Min	RollingAvgTimeToDriveSameDay
Duration	Charge_Start_Time_Weekday (one-hot encoding)	Sun_Load_Average	Charge_Start_Time_Weekday (one-hot encoding)
RollingAvgTripDist	Start_Time_Weekday (one-hot encoding)	Sun_Load_Std	Start_Time_Weekday (one-hot encoding)
RollingAvgTripDistSameDay	Charge_Start_Time_Daypart (one-hot encoding)	Sun_Load_Min	Charge_Start_Time_Daypart (one-hot encoding)
Charge_Start_Time_Weekday (one-hot encoding)	Start_Time_Daypart (one-hot encoding)	Duration	Start_Time_Daypart (one-hot encoding)
Start_Time_Weekday (one-hot encoding)	End_Time_Daypart (one-hot encoding)	RollingAvgTripDist	End_Time_Daypart (one-hot encoding)
Charge_Start_Time_Daypart (one-hot encoding)	Midnight_Weekday (one-hot encoding)	RollingAvgTripDistSameDay	Midnight_Weekday (one-hot encoding)
Start_Time_Daypart (one-hot encoding)		Charge_Start_Time_Weekday (one-hot encoding)	
End_Time_Daypart (one-hot encoding)		Start_Time_Weekday (one-hot encoding)	
Midnight_Weekday (one-hot encoding)		Charge_Start_Time_Daypart (one-hot encoding)	
		Start_Time_Daypart (one-hot encoding)	
		End_Time_Daypart (one-hot encoding)	
		Midnight_Weekday (one-hot encoding)	

3.4.1.2 Feature Selection per Model

The second step in the feature selection is done by performing backward sequential feature selection on each prediction model. This is a greedy algorithm that attempts to find the subset of features that yields the best score, in our case the lowest aggregated MAE for all cars. Given a model and n features, the algorithm first computes the model's MAE using all n features by performing progressive validation. Next, each feature is sequentially disregarded from the model, one at a time, and the model's MAE is computed for all subsets with $n-1$ features. The feature which by its removal improves the model's MAE the most is removed. This procedure of removing features continues until the model's MAE does not improve by removing any more features.

3.4.2 Hyperparameter Selection

The hyperparameter selection is performed by using *Grid Search*. For each model, a subset of the hyperparameter space is chosen to be evaluated. The Grid Search algorithm then performs an exhaustive search among the possible combinations of hyperparameters. For each combination, the model is evaluated using progressive validation, and the combination which yields the lowest MAE over all cars is selected.

3.5 Evaluation

This section describes how the machine learning models are evaluated. The evaluation is done in two different ways. Section 3.5.1 describes the approach of the quantitative evaluation. Section 3.5.2 outlines the qualitative evaluation and the sensitivity analysis on which the qualitative evaluation is based.

3.5.1 Quantitative Evaluation

The quantitative evaluation of the incremental models is performed by using progressive validation. As explained in Section 2.4.1, this is different from how a batch learning model is usually evaluated, and is more similar to how incremental models would be used in production. Furthermore, as explained in Section 3.4, the evaluation is made on the held-out test set consisting of well-behaving cars not used in the feature selection or the hyperparameter selection.

We use the established intrinsic evaluation metric MAE [49], which is used in plenty of the related works mentioned in Section 1.2. MAE is calculated on an aggregated level for all vehicles, and on an individual level. Computing these metrics for separate cars can help identify and analyze cars on which the models perform worse. Beyond making predictions, the models also attempt to create 90 % prediction intervals, which are evaluated using PICP and MPIW. Moreover, to give the prediction models a chance to learn before evaluation, we do not consider the predictions of the first 20 drives on each car when calculating MAE, PICP, and MPIW.

3.5.2 Qualitative Evaluation

The qualitative evaluation aims to provide information about how the energy consumption of a drive (including possible preconditioning) is affected by following the predictions of the prediction models. The qualitative evaluation is based on a sensitivity analysis, in which we investigate the effect of adapting the battery thermal preconditioning to incorrect predictions of TTL and TD. The sensitivity analysis depends on dynamical models simulating the thermal preconditioning of the battery and the energy consumption during a drive.

An essential aspect of this analysis is to determine if the energy in the car's battery should have an equal value to the energy in the grid. When preheating the battery, energy from the grid is used to make the battery more energy efficient. There are several reasons why the energy in the battery may be considered more important. For example, it increases the range of the vehicle, which is essential for customer satisfaction. In this study, we add a weight to the energy in the battery to indicate this importance. We call this weight α, and the exact value is decided in collaboration with Volvo Cars.

The approach of the sensitivity analysis is to consider one of the metrics at a time, e.g., keeping the TD fixed while measuring the effects of predicting different TTL. The approach is inspired by Zhu et al. [50], who investigate the optimal sizing of a battery-supercapacitor energy storage system for electric vehicles by changing one variable at a time.

A model achieving good performance in the quantitative evaluation is thereafter selected to be evaluated in terms of average weighted energy consumption per drive, based on the result of the sensitivity analysis. In other words, we compare the average energy consumption per drive generated when using different strategies, such as never preconditioning or always preconditioning, with the average energy consumption per drive when the battery thermal preconditioning process is adapted to the predictions of the chosen model.

The following sections describe the dynamical models on which the sensitivity analysis is based, and also how the sensitivity analysis is performed.

3.5.2.1 Energy Consumption Simulation

The energy consumption during a drive is simulated using an existing dynamical model developed at Volvo Cars. The data from the simulation covers the energy consumption, trip distance, and trip duration of a drive in three ambient temperatures. At each ambient temperature, three cases are included, in which the battery is heated to specific temperatures before the drive. The drive consists of two *Worldwide harmonized Light vehicles Test Cycles* (WLTC). A WLTC is a standardized drive cycle that consists of a predefined sequence of different accelerations performed over a specific duration of time.

The simulation gives a fundamental understanding of when it is beneficial to precondition the battery before a drive. However, the different ambient temperatures

and battery temperatures are too coarse-grained to perform the sensitivity analysis accurately. Therefore, a linear interpolation is performed to approximate energy consumption at all temperatures between the provided cases.

3.5.2.2 Battery Thermal Preconditioning Simulation

The simulation we develop for simulating the battery thermal preconditioning is based on an existing battery preconditioning strategy. This strategy starts the heating process at a specific time before the given departure time and heats the battery to a specific target temperature.

Our simulation attempts to mimic this strategy given an ambient temperature, the predicted TTL, and the actual TTL. The simulation adapts the heating process to the predicted TTL and simulates how the energy consumption and battery temperature change at each time step. Thus, it is possible to see what effect an incorrect prediction of the TTL has on the battery starting temperature at the actual TTL, and the amount of energy used at the actual TTL.

In the simulation, the battery's temperature depends on numerous factors: the battery's heat capacity and mass, the thermal power from a high voltage coolant heater, the thermal power from internal resistive losses, and the thermal power lost or gained because of the thermal conductance between the battery and ambient. The following function defines how the battery temperature changes over time.

$$\frac{\delta T_b}{\delta t} = \frac{1}{c_p m_b}(Q^b_{hvch}(t) + Q_{loss}(t) - Q_{amb}(t)) \tag{3.5}$$

where T_b is the battery temperature, c_p is the heat capacity, m_b is the mass, $Q^b_{hvch}(t)$ is the active battery heating, $Q_{loss}(t)$ is the thermal power from internal resistive losses, $Q_{amb}(t)$ is the thermal power lost to or gained from ambient and t is the time step.

The active battery heating using a high voltage coolant heater is defined by

$$Q^b_{hvch}(t) = \eta P^b_{hvch}(t) \tag{3.6}$$

where $P^b_{hvch}(t)$ is the electrical power used to heat the battery and η is a constant defining the efficiency of the heater.

The thermal power from the internal resistive losses is defined by:

$$Q_{loss}(t) = R(SoC(t), T_b(t))I_b^2 \tag{3.7}$$

where $R(SoC(t), T_b(t))$ is the resistance given a state of charge and the current battery temperature, and I_b is the battery current.

The thermal power gained or lost due to the thermal conductance between the battery and ambient is defined as:

$$Q_{amb}(t) = \lambda_a(T_b(t) - T_{amb}(t)) \tag{3.8}$$

where λ_a is a constant defining the heat transfer between the battery and ambient at standstill, $T_b(t)$ is the battery temperature and $T_{amb}(t)$ is the ambient temperature.

3.5.2.3 Distance Prediction Error Sensitivity Analysis

As mentioned in Section 1.1, initial studies at Volvo Cars indicate that the distance and ambient temperature of a drive highly determines the benefit of preconditioning the battery. The benefit seems to grow with longer distances and colder ambient temperatures.

When analyzing how sensitive the battery thermal preconditioning is to incorrect TD predictions, we assume that the battery always reaches a specific target temperature during preconditioning. We then use the energy consumption simulation to decide, for each ambient temperature, what driving distance is needed for preconditioning to be beneficial. When we have a prediction of TD, we can use these thresholds to decide whether or not the battery should be preconditioned. Note that it is only whether the predicted TD is lower or higher than the distance threshold that decides whether preconditioning should be performed. Thus, it is acceptable that a predicted TD is inaccurate as long as it is on the right side of the distance threshold.

Apart from creating the distance thresholds, we also use the energy consumption simulation to investigate how much energy is lost when predicting on the wrong side of the threshold for different distances and ambient temperatures.

3.5.2.4 Time Prediction Error Sensitivity Analysis

Adapting the battery thermal preconditioning process to a prediction of TTL will naturally affect the battery's temperature at the actual TTL. A range of predictions before and after the actual TTL is selected to investigate the effects of TTL prediction errors. These predictions are given as input to the battery preconditioning simulation, described in Section 3.5.2.2, to simulate what the battery's temperature is at the actual TTL. The battery temperatures at the actual TTL are then used as input to the energy consumption model, making it possible to investigate how much the temperature differences affect the energy consumption at different distances. The driving energy consumption results are then added to the energy cost for heating the battery, making it possible to deduce at what prediction errors the preconditioning is still beneficial. Note that the total energy consumption is weighted, see Section 3.5.2.

4

Results

This chapter presents the results of the different parts of the study. In Section 4.1, the findings of the sensitivity analysis are described, that is, how sensitive the battery thermal preconditioning is to incorrect predictions. The selected hyperparameters for each prediction model are reported in Section 4.2. In Section 4.3 the result of the quantitative evaluation is presented, describing how well the prediction models can predict TD and TTL for both the midnight and charge approach. Finally, the outcome of the qualitative evaluation is reported in Section 4.4.

4.1 Sensitivity Analysis

The following two sections present the result of the sensitivity analysis, and attempt to answer how sensitive the performance of the battery preconditioning is to incorrect predictions of TD and TTL.

4.1.1 Trip Distance

The driving distance and the ambient temperate are the two main variables deciding whether it is beneficial to precondition the battery before departure. Thus, if the driving distance and ambient temperature are known, it is possible to decide whether the battery should be preheated or not. Figure 4.1 shows, for different ambient temperatures, the driving distance that is needed for it to be beneficial to precondition the battery temperature when using the current preheating strategy. The distance threshold increases as the ambient temperature increases, and at around -2.5C the needed driving distance grows larger than what the simulated data, described in Section 3.5.2.1, contains.

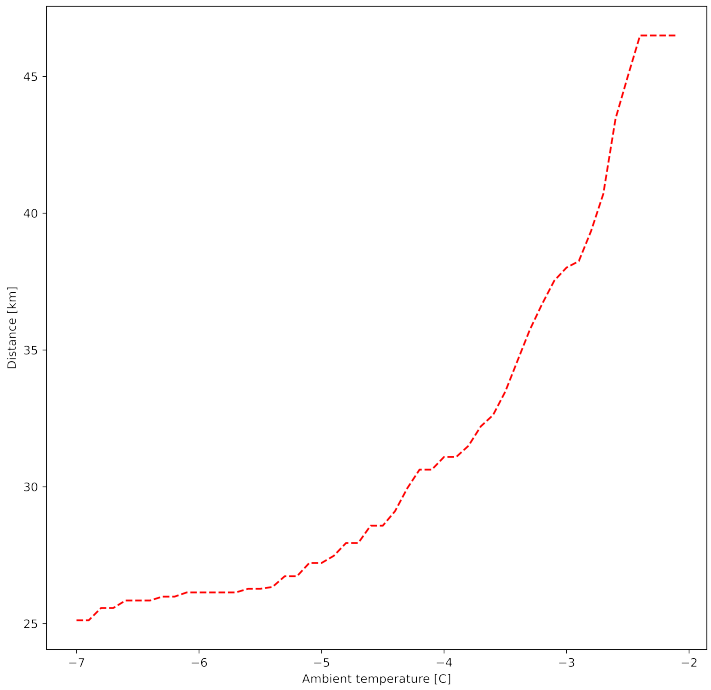


Figure 4.1: The threshold distance at which it becomes beneficial to preheat the battery for different ambient temperatures.

Figure 4.2 presents, for a specific ambient temperature, the energy consumption of different driving distances depending on whether the battery has been preconditioned or not. The red dashed line indicates the distance from which it starts to be beneficial to precondition the battery. It can be seen in the figure that when the driving distance is longer than the threshold, the benefit of preconditioning grows larger with increasing distance. On the other hand, when the driving distance is shorter than the threshold, the energy waste from unnecessarily preconditioning the battery grows larger with decreasing distance. In the figure, it can also be seen that at the considered distances, the possible gain from preconditioning the battery when driving distances longer than the threshold is lower than the energy wasted by preconditioning the battery for drives shorter than the threshold.

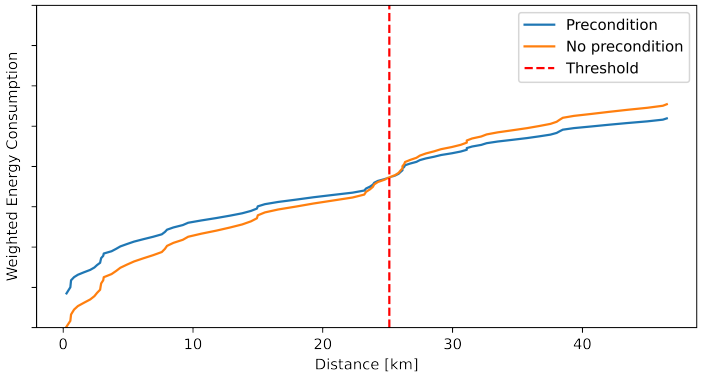


Figure 4.2: Energy consumption over driven distances depending on whether the battery is preconditioned or not.

4.1.2 Time To Leave

Given that it is known that preheating can be beneficial for an upcoming drive, it is crucial that the preheating is performed at the right time. The battery starting temperature might be sub-optimal, and energy might be wasted if the preheating process is adapted to the wrong starting time.

Figure 4.3 presents how the weighted energy consumption from a drive and preheating are affected by ambient temperature, driving distance, and incorrect predictions of TTL. The figure shows that it is not favorable to preheat the battery when driving 30 or 40 km at 0C, even if the departure time is predicted perfectly. The weighted energy consumption is lower if the battery is not preheated, i.e, the dashed green line is always below the filled green line. However, if the ambient temperature is -4C or -7C, it can be advantageous, given that the preheating is performed at the right time. For example, if the driven distance is 40 km and the ambient temperature is -7C, the acceptable prediction interval is around 2.7 hours before and 2 hours after the actual departure time. Thus, if the preheating process is adapted to a time inside this interval, the weighted energy consumption will be lower than if the battery was not preheated. However, if the driving distance is 30 km, the acceptable prediction interval becomes shorter. The figure further indicates that what can be gained by preconditioning the battery according to an accurate prediction of TTL is less than what might be lost if the battery preconditioning is adapted to an incorrect prediction.

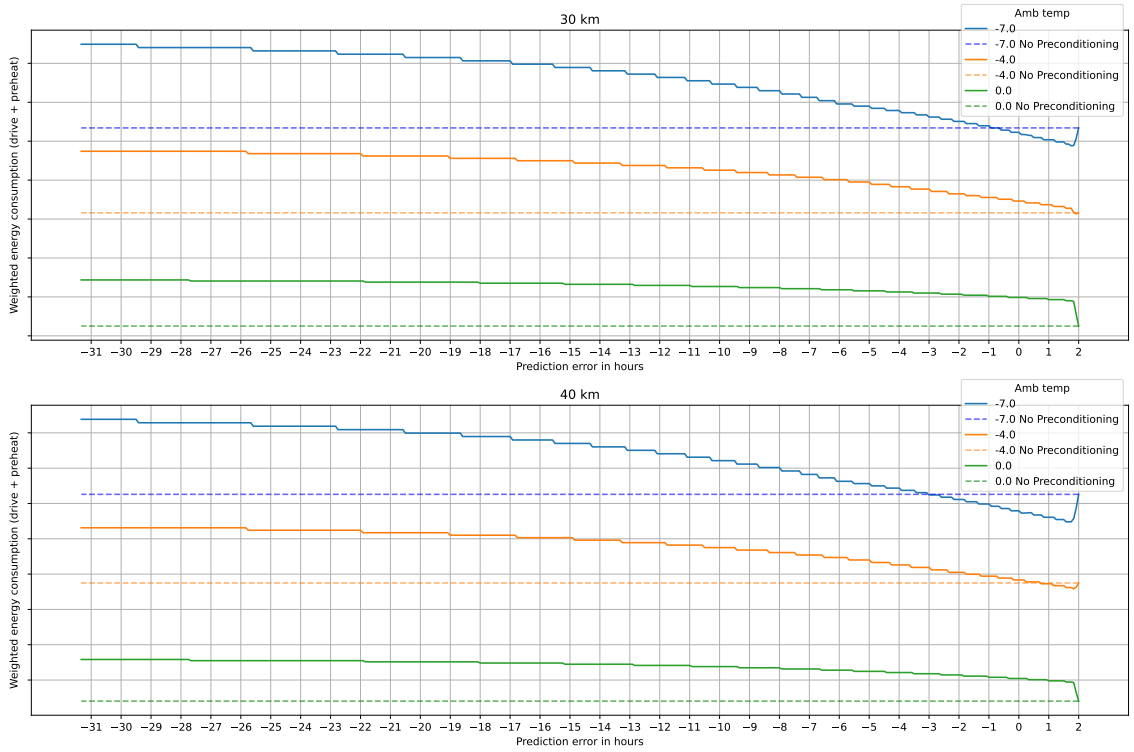


Figure 4.3: How the energy consumption is affected by driving distance, ambient temperature and incorrect predictions of TTL. The top figure shows the weighted energy consumption for a 30 km drive and the lower for a 40 km drive. The dashed lines are displayed for comparison and shows the energy consumed if no preheating is performed. Note that if the prediction error is larger than +2 hours, then the drive will begin before the preheating processing even starts.

4.2 Hyperparameter Selection

Table 4.1 displays the subspace of hyperparameters considered for the prediction models when predicting TD and TTL using either the midnight or charging approach. The selected values are used in the qualitative and qualitative evaluation.

Table 4.1: An overview of the tested and selected hyperparameters for each combination of prediction model, target variable and prediction approach.

Model	Hyperparameter	Selected Value				Options Explored
		TD Charge	TD Midnight	TTL Charge	TTL Midnight	
QR	learning rate	1e-2	1e-1	1e-1	1e-1	[1e-4, 1e-3, 1e-2, 1e-1]
	l2	0	0	0	0	[0, 0.25, 0.5, .75, 1, 2.5, 5]
	# selected features	27 (out of 51)	12 (out of 60)	17 (out of 46)	13 (out of 55)	
QKNN	window size	140	140	120	120	[60, 80, 100, 120, 140]
	neighbors	15	20	12	15	[3, 5, 8, 10, 12, 15 ,20]
	#selected features	45 (out of 51)	51 (out of 60)	37 (out of 46)	33 (out of 55)	
QARF	number of trees	60	30	20	60	[10, 20, 30 ,40 ,50, 60]
	leaf prediction	linear regression	mean	mean	linear regression	[mean, linear regression]
	# selected features	49 (out of 51)	58 (out of 60)	45 (out of 46)	52 (out of 55)	
FFNN	learning rate	1e-4	1e-4	1e-3	1e-3	[1e-5, 1e-4. 1e-3, 1e-2]
	hidden layers	5	5	2	1	[1,2,3,4,5]
	neurons in hidden layer(s)	25	25	200	200	[25,50,100,200]
	dropout	0.2	0.1	0.1	0.1	[0.05,0.1,0.2,0.5]
	# selected features	49 (out of 51)	58 (out of 60)	45 (out of 46)	55 (out of 55)	

4.3 Quantitative Evaluation

The following sections present the performance of the different prediction models when predicting TD and TTL, using either the midnight or charge approach. Further, the result from the evaluation of the prediction uncertainty is displayed.

4.3.1 Trip Distance From Midnight

Table 4.2 presents a summary of how well the models manage to predict TD using the midnight approach. All models achieve a lower MAE than the baseline (Mean), with the QARF-model attaining the lowest error: 12.54 km. QKNN achieves the second-lowest MAE, and both QARF and QKNN predict 50 % of the driving distances within 5 km. The lowest MAE on a single car is 4.72 km, with 76 % of the predictions being closer than 5 km from the actual driving distance.

The FFNN-model almost produces a PICP of 90 %, with an MPIW of 55.4 km. QARF also nearly reaches a PICP of 90 % but with a larger MPIW than FFNN. QR produces a much lower PICP and MPIW than the other models.

Table 4.2: The models' performances when predicting TD from midnight. The table displays both the aggregated result over all the cars and also the result for the car on which the models achieved the best result.

Model	MAE (km)		Percentage of predictions with error less than 5 km		PICP	MPIW
	All Cars	Best Car	All Cars	Best Car	All Cars	All Cars
Mean	15.88	6.10	0.2	0.52	-	-
QR	13.34	5.18	0.44	0.77	0.74	30.3
QKNN	12.57	4.77	0.5	0.76	0.81	49.74
QARF	12.54	4.72	0.5	0.76	0.87	62.5
FFNN	14.76	5.64	0.27	0.55	0.89	55.4

In Figure 4.4 a more in-depth view of the models' performance is given as the MAE on each car is shown.

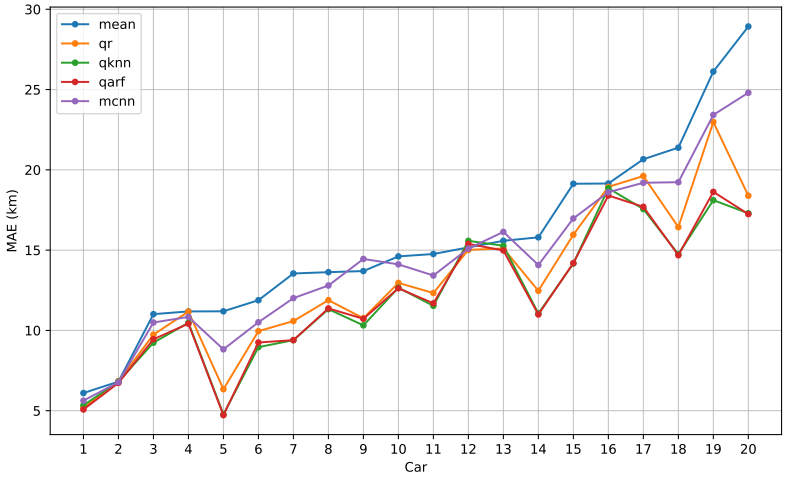
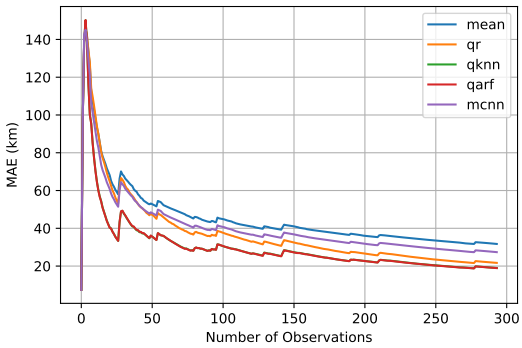
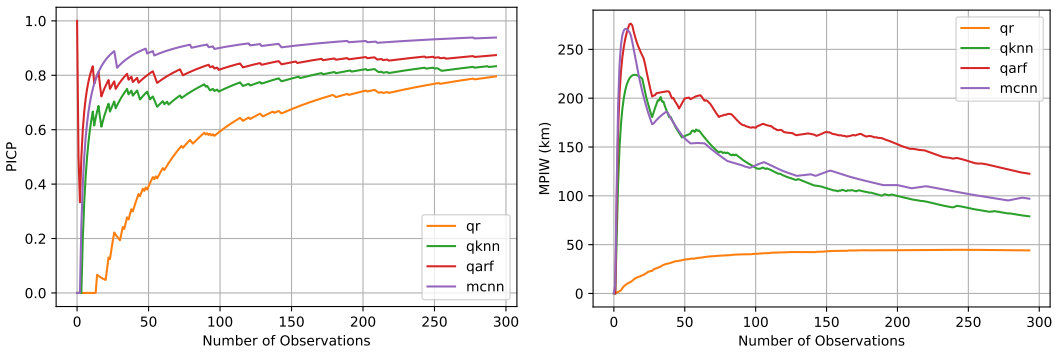


Figure 4.4: MAE per car when predicting TD using the midnight approach. The cars are sorted by the MAE of the mean baseline.

Figure 4.5 shows examples from a single car of how MAE, PICP, and MPIW change when the number of observations increases. It is clear from Figure 4.5a that the models have similar learning curves for this specific car. However, while the learning curves are similar, Figure 4.5b shows that QR is slower in learning an acceptable prediction interval compared to the other models.



(a) MAE



(b) PICP **(c)** MPIW

Figure 4.5: Examples from a single car on how the MAE, PICP, and MPIW change over time when TD is predicted using the midnight approach.

Table 4.3 displays the performance of the prediction models when predicting TD using the midnight approach on the generated data. On the basic pattern, QARF yields a slightly lower MAE than the baseline, while all other models receive a larger MAE. Considering the uncertainty quantification, FFNN yields the most accurate PICP while having a significantly broader prediction interval compared to the other models. On the irregular pattern, QR yields the least MAE of 6.25 kilometers, while QKNN yields a slightly larger MAE. The baseline yields a significantly larger MAE, while FFNN yields an even larger MAE. QARF achieve the most accurate PICP compared to the other models, having a broader interval than QR and QKNN but a narrower interval than FFNN.

Table 4.3: The models' performances when predicting TD from midnight using the generated data. The table displays the result over the two patterns described in Section 3.2.2.

Model	MAE (km)		PICP		MPIW	
	Basic	Irregular	Basic	Irregular	Basic	Irregular
Mean	2.42	19.20	-	-	-	-
QR	2.45	6.25	0.83	0.87	6.38	26.00
QKNN	2.62	6.72	0.78	0.81	6.45	25.39
QARF	2.39	8.26	0.86	0.87	7.86	43.00
FFNN	2.81	22.13	0.89	0.84	14.04	86.01

4.3.2 Trip Distance From Charge

Table 4.4 presents the performance of the prediction models when predicting TD using the charge approach. The MAE of the baseline is 26.57 kilometers, and all models except FFNN beat this error. QKNN achieves the lowest error, 22.89 km, and predicts 35 % of the driving distances within 5 km. FFNN succesfully manages to achieve a PICP of 90 %, and QARF is close with 85 %. Just as in the midnight approach, the MPIW of FFNN is smaller than that of QARF. The QR model's PICP is notably low.

Table 4.4: The prediction models' performances when predicting TD using the charge approach. The table displays both the aggregated result over all the cars and also the result for the car on which the models achieved the best result.

Model	MAE (km)		Percentage of predictions with error less than 5 km		PICP	MPIW
	All Cars	Best Car	All Cars	Best Car	All Cars	All Cars
Mean	26.57	7.17	0.14	0.45	-	-
QR	24.22	7.17	0.23	0.46	0.55	38.2
QKNN	22.89	7.53	0.35	0.54	0.79	75.4
QARF	23.06	7.21	0.35	0.57	0.85	94.1
FFNN	26.81	7.15	0.14	0.42	0.90	89.3

Figure 4.6 gives a more detailed view of the models' performances as the MAE on each car is shown.

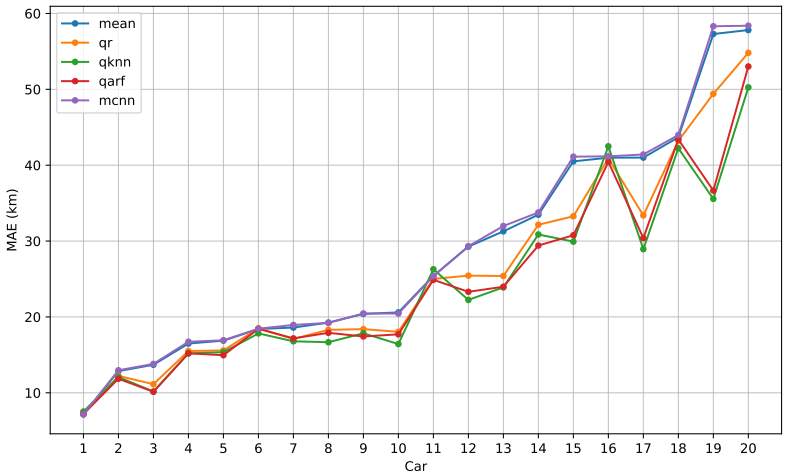
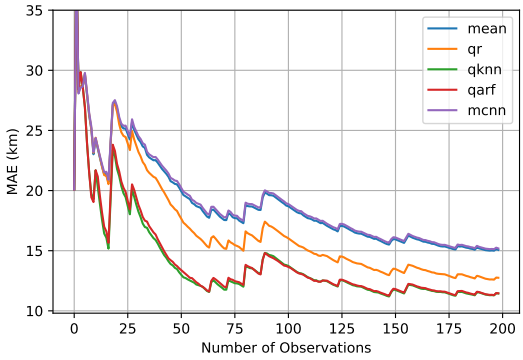
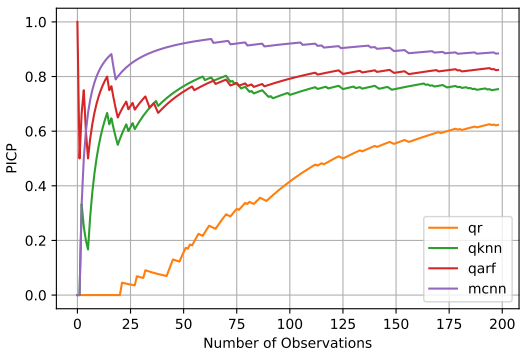


Figure 4.6: The MAE per car when predicting TD using the charge approach. The cars are sorted by the MAE of the baseline (mean).

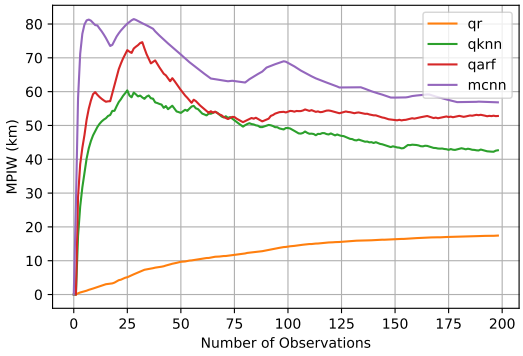
Figure 4.7 displays examples from a single car of how MAE, PICP, and MPIW change when the number of observations increases. Figure 4.7a shows, just as in the example from the midnight approach, that the different models have similar learning curves. Again it is also clear that the QR-model struggles to learn an acceptable prediction interval in the same number of observations as the other models; see Figure 4.7b.



(a) MAE



(b) PICP



(c) MPIW

Figure 4.7: Examples from a single car on how the MAE, PICP, and MPIW change over time when predicting TD using the charging approach.

Table 4.5 displays the performance of the prediction models when predicting TD using the charging approach on the generated data. On the basic pattern, QR and QARF receives similar MAE as the baseline, while QKNN and FFNN receive slightly higher MAE. In contrast, FFNN yields the most accurate PICP, despite having the largest MPIW. On the irregular pattern QKNN receives the lowest MAE while the baseline gets the largest MAE. Further, QARF yields the most accurate PICP, while also having having a more narrow prediction interval than FFNN.

Table 4.5: The models' performances when predicting TD from charge using the generated data. The table displays the result over the two patterns described in Section 3.2.2.

Model	MAE (km)		PICP		MPIW	
	Basic	Irregular	Basic	Irregular	Basic	Irregular
Mean	2.79	18.96	-	-	-	-
QR	2.78	12.23	0.78	0.72	5.54	25.58
QKNN	3.16	6.83	0.79	0.81	9.18	23.08
QARF	2.78	8.47	0.84	0.87	9.51	46.54
FFNN	2.83	16.14	0.89	0.85	14.81	76.72

4.3.3 Time To Leave From Midnight

Table 4.6 presents how well the models manage to predict TTL using the midnight approach. The MAE of the baseline is 3.10 hours, and all models improve this error. The lowest MAE, 2.63 hours, is achieved by QR. However, it is QARF that achieves the best error on a single car with an MAE of 1.12 hours and 76 % of the predictions within an hour of the actual departure time. QKNN predicts 38 % of all departure times within an hour of the actual departure time. No model produces a PICP of 90 %; however, FFNN and QARF are close with 88% and 86% respectively.

Table 4.6: The prediction models' performances when predicting TTL from midnight. The table displays both the aggregated result over all the cars, but also the result for the car on which the models achieved the best result.

Model	MAE (hour)		Percentage of predictions with error less than 1 h		PICP	MPIW
	All Cars	Best Car	All Cars	Best Car	All Cars	All Cars
Mean	3.10	2.32	0.15	0.29	-	-
QR	2.63	1.22	0.34	0.70	0.73	7.8
QKNN	2.67	1.21	0.38	0.73	0.80	9.43
QARF	2.76	1.12	0.34	0.76	0.86	10.35
FFNN	2.91	1.61	0.2	0.46	0.88	11.7

A more exhaustive view of the models' performances can be seen In Figure 4.8, where the MAE for each car is shown.

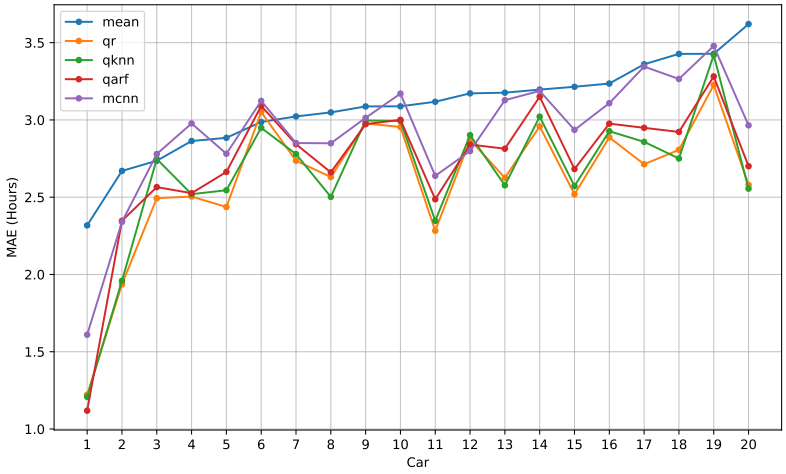
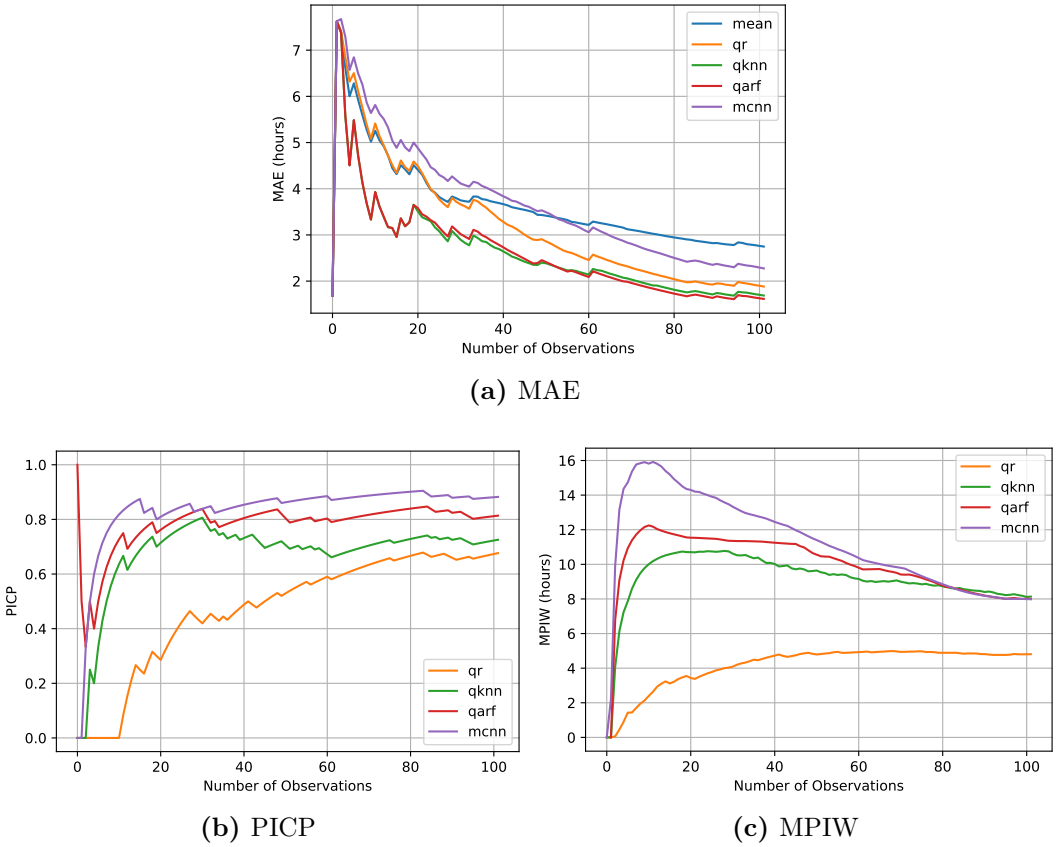


Figure 4.8: MAE per car when predicting TTL using the midnight approach. The cars are sorted by the MAE of the baseline (mean).

Examples from a single car of how MAE, PICP, and MPIW change when the number of observations increases are presented in Figure 4.9. Figure 4.9b shows that the QR model is much slower than the other models at learning an acceptable prediction interval, as it needs many more observations to almost reach the same PICP as the other models.



(a) MAE

(b) PICP

(c) MPIW

Figure 4.9: Examples from a single car on how the MAE, PICP, and MPIW change over time when predicting TTL using the midnight approach.

Table 4.7 displays the performance of the prediction models when predicting TTL using the midnight approach on the generated data. On the basic pattern, all models yield similar errors and outperform the baseline by almost 2 hours. Further, QR and FFNN have the highest PICP on the basic pattern, although QR has a narrower MPIW. QKNN achieves the lowest MAE on the irregular pattern, followed by QR having less than 6 minutes larger MAE. Additionally, all of the models seem to outperform the baseline, and FFNN yields the most accurate PICP.

Table 4.7: The models' performances when predicting TTL from midnight using the generated data. The table displays the result over the two patterns described in Section 3.2.2.

Model	MAE (km)		PICP		MPIW	
	Basic	Irregular	Basic	Irregular	Basic	Irregular
Mean	2.37	3.18	-	-	-	-
QR	0.58	0.89	0.89	0.86	2.29	4.98
QKNN	0.57	0.81	0.82	0.83	1.62	2.68
QARF	0.55	1.24	0.86	0.92	1.68	7.63
FFNN	0.59	1.08	0.89	0.89	3.15	5.12

4.3.4 Time To Leave From Charge

Table 4.8 displays the performance of the prediction models when predicting TTL using the charge approach. All models improve the baseline MAE of 6.33 hours. QKNN achieves the lowest MAE with 5.51 hours and predicts 22 % of all departure times within 1 hour. For the best car, QKNN predicts 52 % of the departure times within an hour and attains an MPIW of 2.17 hours.

Again, FFNN and QARF are close to producing a PICP of 90 %. QR and QKNN produce smaller MPIWs, but they also attain much lower PICP.

Table 4.8: The prediction models' performances when predicting TTL using the charge approach.

Model	MAE (hour)		Percentage of predictions with error less than 1 h		PICP	MPIW
	All Cars	Best Car	All Cars	Best Car		
Mean	6.33	2.82	0.12	0.36	-	-
QR	5.67	2.12	0.18	0.48	0.70	15.1
QKNN	5.51	2.17	0.22	0.52	0.77	18.8
QARF	5.69	2.78	0.19	0.40	0.86	24.0
FFNN	5.90	2.40	0.15	0.45	0.87	22.0

Figure 4.10 shows the MAE of each car in the test set, giving a more extensive view of the models' performances.

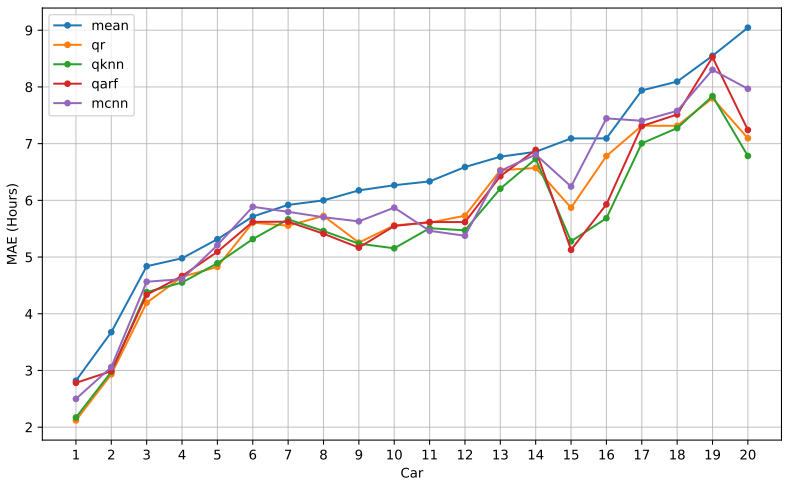
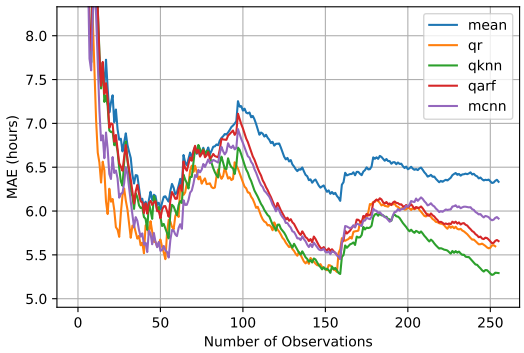
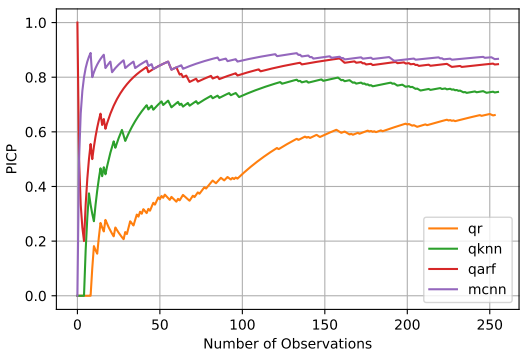


Figure 4.10: The MAE per car when predicting TTL using the charge approach. The cars are sorted by the MAE of the baseline (mean)

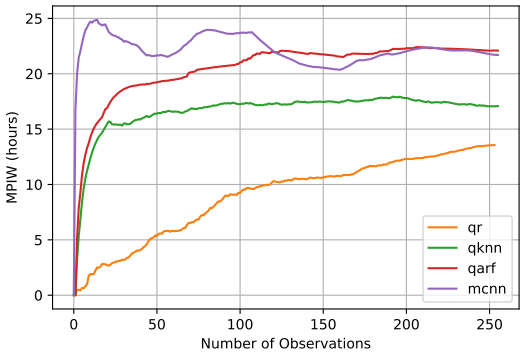
Figure 4.11 presents how the MAE, PICP and MPIW changes for a specific car when the number of observations increases. As in the previous approaches, the models seem to have similar learning curves, see Figure 4.11a. Figure 4.11c shows that the MPIW of QR is increasing much slower than the rest of the models.



(a) MAE



(b) PICP



(c) MPIW

Figure 4.11: Examples from a single car on how the MAE, PICP, and MPIW changes over time when predicting TTL using the midnight approach.

Table 4.9 displays the performance of the prediction models when predicting TTL using the charge approach on the generated data. On the basic pattern, all models, including the baseline, have an MAE of approximately 1 hour. QKNN achieves the lowest MAE on the irregular pattern, and all of the proposed models perform significantly better than the baseline. When quantifying the uncertainty, QR, QARF and FFNN are all close to the targeted PICP, whereas QR seems to yield the narrowest MPIWs when considering both patterns.

Table 4.9: The models' performances when predicting TTL from charge using the generated data. The table displays the result over the two patterns described in Section 3.2.2.

Model	MAE (km)		PICP		MPIW	
	Basic	Irregular	Basic	Irregular	Basic	Irregular
Mean	1.04	8.72	-	-	-	-
QR	0.96	3.52	0.88	0.87	3.12	15.23
QKNN	0.95	1.99	0.76	0.79	2.13	6.21
QARF	1.00	2.94	0.89	0.93	3.10	16.51
FFNN	0.97	2.99	0.91	0.91	5.71	15.83

4.4 Qualitative Evaluation

This section displays the result of adapting the battery thermal preconditioning process to the predictions from the QARF-model. Only the predictions based on the midnight approach will be considered, as this yields significantly lower MAE. Note that in the TD evaluation, we assume that the preheating process is always adapted to the actual departure time. In the TTL evaluation, we only consider drives where preconditioning is beneficial.

4.4.1 Trip Distance

As mentioned in Section 4.1.1, it is the driving distance together with the ambient temperature that decides whether it is beneficial to precondition the battery before departure. Thus, if the driving distance and ambient temperature are known, we can determine whether the battery should be preheated or not. Figure 4.12 presents the average weighted energy consumption for a drive plus possible preheating when using different methods to decide whether or not to preheat before departure. The figure indicates that following the predictions of QARF yields a similar average weighted energy consumption to the strategy of never preconditioning the battery.

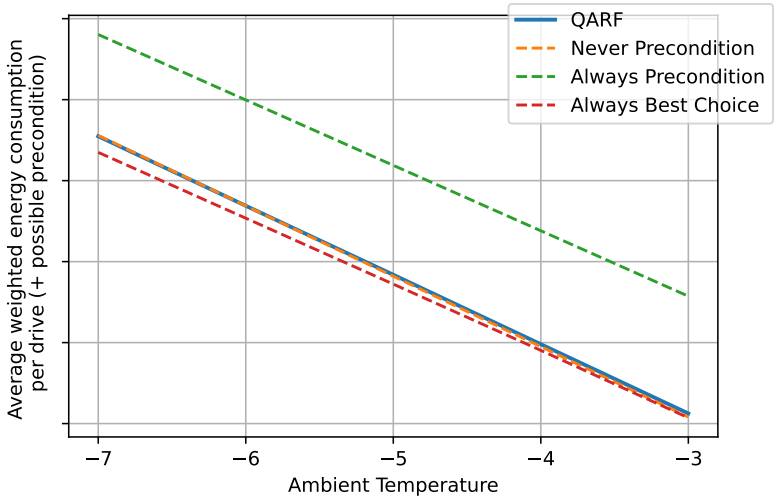


Figure 4.12: The average weighted energy consumption per drive plus possible preheating when using different methods to decide whether or not to preheat the battery.

Figure 4.13 displays a normalized confusion matrix of the QARF-model's predictions on whether it is beneficial to preheat the battery or not. Note that the model does not predict true or false, but rather it predicts a trip distance and uses the distance threshold shown in Figure 4.2 to decide whether the battery should be preconditioned or not. The model successfully identifies the drives on which it is not beneficial to preheat. However, it struggles to find the drives where it is beneficial to preheat.

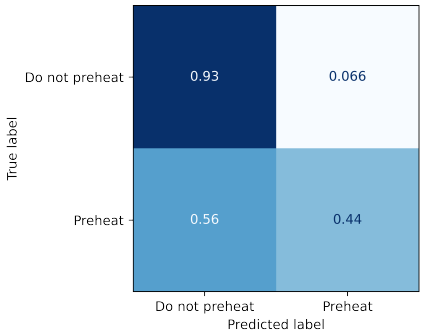


Figure 4.13: Normalized confusion matrix of the QARF-model's predictions on whether the battery should be preheated or not when the ambient temperature is -7C. Of all drives, only 18 % benefited from preheating.

4.4.2 Time To Leave

As explained in Section 4.1.2, given that preheating before a drive is beneficial, it is essential that the preheating is performed at the right time. Otherwise, grid energy might be wasted or the battery's starting temperature might be sub-optimal. This evaluation only considers the drives that may benefit from preheating the battery to a specific goal temperature. Figure 4.14 displays the average weighted energy consumption for a drive plus preheating in various ambient temperatures when using different methods to predict the departure time. Adapting the departure times according to QARF's predictions generates, at all temperatures, a slightly higher average weighted energy consumption than if we never precondition the battery. Predicting the exact departure time does not achieve the lowest average weighted energy consumption because of how the current preconditioning strategy works.

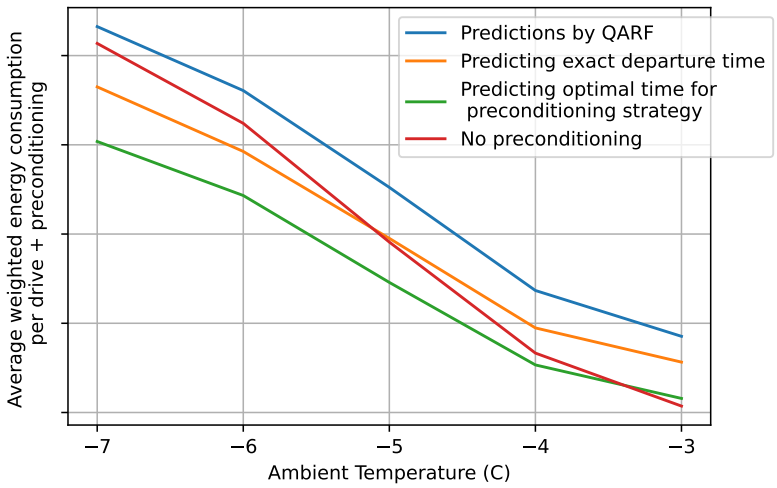
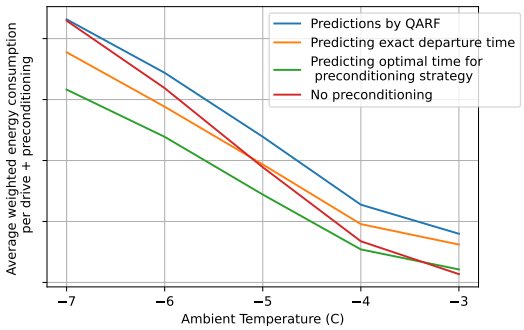


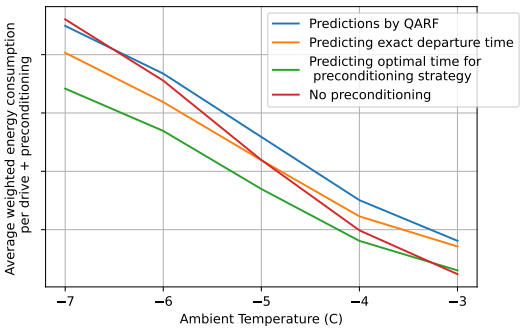
Figure 4.14: The average weighted energy consumption per drive plus preheating when using different methods to predict the departure time.

Figure 4.15 also presents the result of using different approaches for predicting TTL, but only for the drives which the QARF-model considers to be in the top 50 % or,

respectively, 30 % of the drives where the prediction intervals are the narrowest. When only considering the drives in the top 30 % and setting the ambient temperature to -7C, following the predictions of QARF yields a slightly lower average weighted energy consumption than never preconditioning.



(a) Only considering the top 50 % of drives which the QARF-prediction model consider to have the lowest uncertainty.



(b) Only considering the top 30 % of drives which the QRF-prediction model consider to have the lowest uncertainty

Figure 4.15: The average weighted energy consumption per drive and preheating process

5

Discussion

This chapter is divided into three parts: In Section 5.1 we attempt to answer the first research question by summarizing and discussing the findings of the quantitative evaluation. Section 5.2 reports the key findings from the sensitivity analysis and qualitative evaluation, and attempts to answer the second research question. Lastly, the quality of the study is discussed in Section 5.3, focusing on all aspects, from data gathering to evaluations.

5.1 Quantitative Evaluation

The quantitative evaluation measures the prediction models' ability to predict the correct TD and TTL using the midnight and charging approaches, while also measuring their capability to estimate the uncertainty in their predictions. The following sections discuss different aspects of the quantitative evaluation.

5.1.1 Comparison of the Approaches

A straightforward conclusion from comparing the results described in 4.3 is that we achieve significantly better results with the midnight approach than with the charging approach. All prediction models yield lower MAE when the midnight approach is used, and Tables 4.6 and 4.8 show that the error rates of TTL using the charging approach are approximately double compared to the midnight approach. This is also the case for the generated data, seen in Tables 4.7 and 4.9.

The difference is not as significant for the TD predictions, see Tables 4.2 and 4.4. However, the models still achieve better results using the midnight approach. The MAE on all cars is around 10 km less for all models when using the midnight approach and around 2 km less for the best car.

There are many possible reasons why the midnight approach achieves better results. First of all, it only considers the first drive of the day, while the charging approach considers any drive that follows a charging session. Therefore, it is likely that the drives considered using the midnight approach contain a more significant proportion of commuting drives. Furthermore, when using the charging approach to predict TTL, we predict the time between two events, while in the midnight approach, we only predict the time of an event. In the charging approach, we thus have two

varying factors, the start of the charge and the departure time, while the midnight approach only deals with the departure time. A final reason why the charging approach achieves less accurate results is that the possible TTL prediction range is considerably more extensive. There is no such assumption in the charging approach, and the subsequent drive might not be until the next day or the day after.

5.1.2 Comparison of the Models

The quantitative evaluation of the data from the fleet of BEVs indicates coherency across the approaches in how the models predict the target variables. The least accurate model in almost all cases is the baseline. FFNN slightly outperforms the baseline in most cases, but is less accurate than QR, QKNN, and QARF. QR performs well when predicting TTL, yielding the lowest MAE on all cars using the midnight approach and the lowest MAE on a single car using the charging approach. However, QR does not perform as well when predicting TD and is less accurate than QKNN and QARF. Figures 4.4, 4.6, 4.8, and 4.10 show that QKNN and QARF most often are among the top-performing models. QKNN often performs slightly better than QARF in most cases when including all cars in the test set, while QARF often has a lower MAE for the best single car.

Figures 4.5a, 4.7a, 4.9a and 4.11a display how the MAE of a single arbitrarily chosen car changes as more observations arrive. One can note that all models, including the baseline, have similar curves and that the models' MAE seems to be affected in similar ways when observing new data. This suggests that all models are making quite similar predictions. One reason for this may be that no model manages to learn the full vehicle behavior, but rather they all learn some average or the most common drive. This is also what is indicated by Figure 5.1, which displays the actual TTL distribution of a single car together with the different models' predicted distributions.

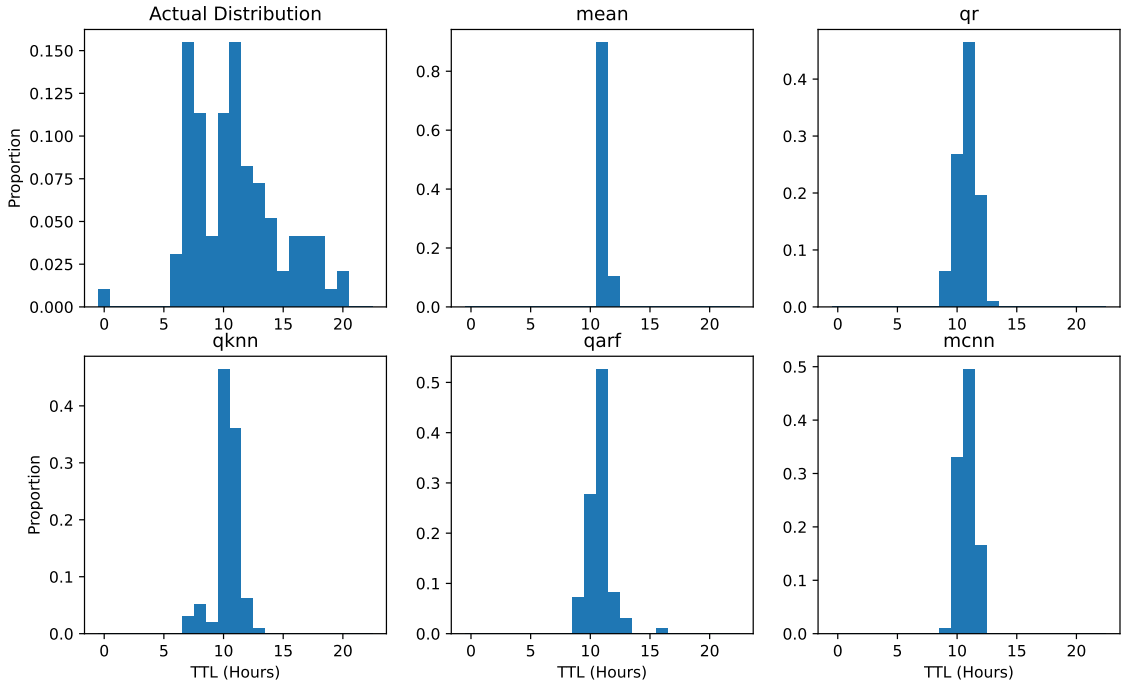


Figure 5.1: Example from a single car of the actual TTL distribution and the predicted TTL distributions.

Considering the predictive performance of the proposed models using the generated patterns, we see a significantly lower MAE in all cases compared to the errors over all cars using the data from the fleet of BEVs. However, when considering the MAE of the best car, the error rate becomes comparable with the results from the irregular pattern. This similarity in error rates could potentially make one think that the irregular pattern is closer to real vehicle usage behavior. But unfortunately, this is not the case. In Figure 5.2, the proposed models predict TTL on the generated data using the irregular pattern. As can be seen, all of the proposed models, except the baseline, can distinguish the different trips present in the data. Comparing these distributions with the predicted distributions in Figure 5.1, it is clear that using the generated data simplifies the problem significantly.

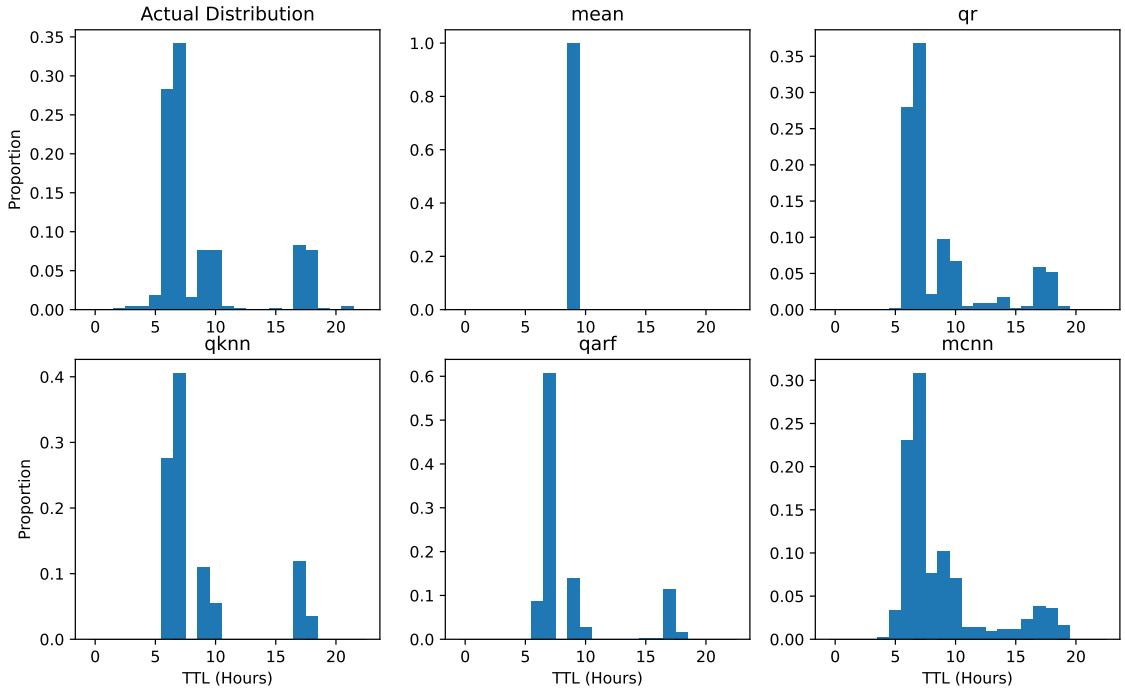


Figure 5.2: Example of the actual TTL distribution and the predicted TTL distributions when using the generated data with the distinct pattern.

5.1.3 Prediction Results in Relation to Related Work

In this section, we compare our results with the results achieved by the related works described in Section 1.2.

In [12], the authors attempt to predict the first daily departure time using several batch-learning models and attain an RMSE from 3.3 to 3.8 hours. Despite only displaying the MAE in Section 4.3, we also measure RMSE, and our QARF yields an RMSE of 3.9 hours aggregated over all cars, and 2.46 hours on the best car when predicting TTL using the midnight approach. In [11], the authors also attempt to predict the first daily departure time using different batch-learning models. The results are quite similar to what is achieved in this project when predicting TTL using the midnight approach, with an MAE of 2.32 to 2.45 hours on weekdays and an MAE of 1.92 to 2.06 hours on weekends.

The two related works mentioned above use an approach similar to our midnight approach. In contrast, the approach in [10] is more similar to our charging approach. The authors attempt to predict TTL from the start of charge using different batch-learning models, but in a much more restrictive setting than ours. A charge session starts when employees at a large company arrive at work, and they try to predict the time the employees will leave work. In this setting, they were able to achieve an MAE of 1.36 to 1.73 hours.

Shifting the focus to TD, [11] also predicts TD for the first daily trip, and achieves an MAE of 20.18 to 23.415 km on weekdays and an MAE of 20.29 to 32.03 km on

weekends. In contrast, our best model achieve an MAE of 12.54 km when using the midnight approach. Furthermore, [11] indicates that neural networks struggle to outperform other regression models.

Collectively among all of the mentioned papers, it seems like neural networks yield poor performance in comparison to less complex models [12, 11, 10, 16]. The overall best-reported models in the related works are often tree-based, such as Random Forest. These two conclusions match the findings of this project.

However, this project is unique compared to the related works, as we use incremental machine learning. Despite the challenges with incremental learning mentioned in Section 2.1, some of the incremental models achieve similar errors as the related works mentioned. Nevertheless, it is difficult to conclude whether incremental learning could substitute the traditional machine learning paradigm, as none of the projects use the same data. To make such a conclusion, a thorough comparison between the incremental and the traditional versions of the models, using the same data, would be needed.

What we can conclude is that predicting vehicle usage is a complex problem. This study and the related works have quite large errors and are far from perfect. However, whether the errors are acceptable is highly situational, and it may be that in some situations, these errors are allowable.

5.1.4 Uncertainty Quantification

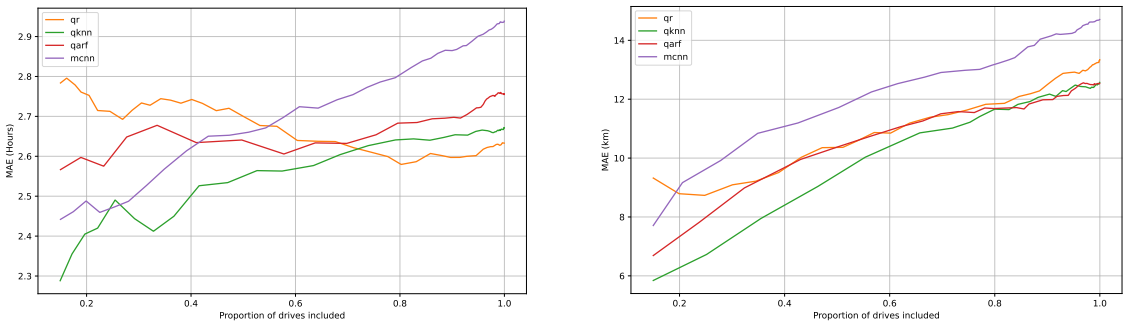
This section analyzes the models' ability to estimate the uncertainty in the predictions. In this study, we quantify the uncertainty by using prediction intervals and we use PICP and MPIW to evaluate the intervals. The prediction intervals should have a PICP equal to the targeted coverage probability while having as low MPIW possible. We target a 90 percent prediction interval, meaning that the PICP should be close to 0.9.

Tables 4.2, 4.4, 4.6 and 4.8 summarize the PICP and MPIW of the prediction intervals that the models produce. The tables show that the prediction intervals generally do not reach the desired coverage probability. However, FFNN and QARF are often close. Most of the time, FFNN produces slightly better results than QARF, as it has a marginally better coverage probability and a slightly less MPIW. QKNN produces lower MPIW than both FFNN and QARF but receives lower PICP. QR produces the most narrow intervals and also the lowest PICP. Figures 4.5b, 4.7b, 4.9b and 4.11b indicate that QR is much slower at learning acceptable intervals than the other models. A possible reason is that QR is training the quantiles separately.

The results indicate that the prediction intervals are not perfectly accurate, as they often do not achieve the desired coverage probability. However, this is somewhat expected. As mentioned in Section 3.5.1, We begin measuring the PICP after the models have observed 20 drives. It is quite unlikely that the first 20 drives represent a car's full vehicle usage during a year, and the models are therefore not expected to directly quantify the uncertainty perfectly. Instead, the models continue to improve

their ability to quantify the uncertainty as more drives are observed.

In Figure 5.3, we measure the impact of the uncertainty quantification in the midnight approach by calculating how the MAE changes when considering different proportions of the most certain predictions, i.e., the predictions with the narrowest prediction intervals. Note that setting the proportion to 1.0 corresponds to the MAE displayed in Tables 4.6, 4.8, 4.2 and 4.4. The figures show that the MAE almost always decreases when the proportion of considered prediction decreases. The exception is QR when predicting TTL, which probably suffers from the poor prediction interval it produces. Interestingly, the decrease in MAE is more significant when predicting TD, which indicates that the uncertainty quantification works better in this case.



(a) TTL **(b)** TD

Figure 5.3: MAE when only considering a specific proportion of predictions with smaller prediction interval. The models considered in these Figures are QR, QKNN, QARF and FFNN using the midnight approach.

5.2 Qualitative Evaluation

The qualitative evaluation is based on a sensitivity analysis, in which we investigate when it is beneficial to perform battery thermal preconditioning, and how inaccurate predictions may affect the preconditioning process. In the following sections, we discuss the results of the sensitivity analysis, and whether the models' predictions are accurate enough to be used for preconditioning the battery before departure.

5.2.1 Benefit of Thermal Preconditioning

Performing battery thermal preconditioning has the positive effect of increasing the battery efficiency. However, it is also costly, as grid energy is needed to heat the battery. When measuring how the energy consumption is affected by the battery thermal preconditioning, we include both the energy consumption of the drive and the energy consumption of heating the battery. However, as mentioned in Section 3.5.2.4, we add a weight α to the energy consumption of the drive. The result of the sensitivity analysis described in Section 4.1 heavily depends on how we choose the weight α, but assuming the currently selected weight, it is clear that battery

thermal preconditioning can help decrease the weighted energy consumption. Figure 4.1 presents, for different ambient temperatures, the driving distance needed for preconditioning to be beneficial.

However, the analysis further shows that reasonably accurate predictions are needed for preconditioning to be beneficial, as inaccurate predictions may significantly affect the weighted energy consumption. In fact, the analysis indicates that what can be gained from an accurate prediction is less than what may be lost by making an inaccurate prediction. Considering predicting TTL, Figure 4.3 indicates that predicting the departure time perfectly only generates a slight decrease in energy consumption compared to the increased energy consumption generated by predicting a 10-hour early departure time. When predicting TD, it is a similar case. The gain from accurately predicting that preconditioning should be performed may be less than what is lost by incorrectly predicting that preconditioning should be performed. Figure 4.2 shows that the gain from preconditioning the battery when driving distances longer than the threshold is lower than the energy wasted by preconditioning the battery for drives shorter than the threshold. Thus, if the predictions are inaccurate, it might be wiser never to precondition the battery.

5.2.2 Machine Learning as Decision-Maker

The results of the qualitative evaluation, described in Section 4.4, shows that the models' predictions are generally not accurate enough to use for thermal battery preconditioning. Considering TD, Figure 4.12 shows that following the model's predictions generates a similar average weighted energy consumption as never preconditioning, whereas always making the right prediction gives a significant improvement.However, following the predictions lead to a lower average weighted energy consumption than always preconditioning.

Despite that the qualitative evaluation of TTL only considers drives that could benefit from preheating, Figure 4.14 shows that following the TTL predictions of the QARF-model generates a worse average weighted energy consumption than never preconditioning. One reason for this might be that the cost of making an incorrect prediction is larger than the gain from making a perfect prediction. Thus, if the predictions are not that accurate, we, on average, might lose more energy by following them than if we never precondition. Figure 4.15 shows that if we only consider the drives where the models' prediction interval is small, then the average weighted energy consumption of following the predictions is slightly less than never preconditioning.

It is important to note that these results are heavily dependent on the considered ambient temperatures and distances, as well as the chosen α. Increasing α will drastically increase the benefit of preconditioning as it decreases the distances needed for preconditioning to be beneficial and widens the acceptable prediction span for TTL. Further, considering colder temperatures and longer distances might also make preconditioning more beneficial and possibly make the predictions more usable.

5.3 Quality of Research

The results of this work are merely a product of the process of producing a prediction. All aspects, from data gathering to hyperparameter selection, affect the prediction. In this section, we highlight and discuss some of the specific choices made in Section 3.

5.3.1 Problem Definition

This project models the prediction problem in two ways: the midnight and charging approaches. The midnight approach yields significantly better results, and in Section 5.1.1 we mentioned several reasons why that is the case. However, the midnight approach is somewhat unrealistic in practice, as we assume that we know whether a drive will occur or not on a specific date. There is no such assumption in the charging approach, making it more applicable in practice. Further, the charging approach allows us to predict more trips than the first drive of the day, as well as trips that do not occur on the same day. The charging approach is also a better fit for the use case of battery thermal preconditioning, as this is a process one most often wants to perform when the car is charging. However, the charging approach yields significantly worse results, possibly indicating that vehicle usage after a charge is not as recurrent as the vehicle usage of the first drive of the day.

5.3.2 Data Gathering

One aspect of the data gathering that could affect the result of this study is that the data has some minor inconsistencies. Occasionally, some driving and charging sessions are missing, and a few times data from whole days are missing. However, we believe the processing procedure mentioned in Section 3.2.1.1 handles the extreme inconsistencies, preventing them from affecting the predictions excessively.

5.3.3 Data Preprocessing

In Section 3.2.1.1, we introduce some filters that we apply to the data to remove anomalies, such as particularly short or long charging and driving sessions. However, the filters are sometimes arbitrarily set, e.g., that two subsequent events are merged if it is less than 15 minutes between them. Arguably, some of these thresholds could be changed while maintaining similar outcomes. However, we do not believe performing a thorough investigation to find optimal thresholds would yield significantly better performance.

Another aspect of the processing is the selection of well-behaving cars, explained in Section 3.2.1.4. The selection attempts to find cars with distinct usage patterns. Notably, we propose a new method of finding well-behaving cars, favoring cars with a strong cluster tendency in the distribution of the target variables. This method seems to eliminate cars having few observations and cars with no regular behavior. However, the method has some drawbacks. Firstly, the method favors cars with few outliers compared to cars with only recurring observations, due to the sampling

process of the uniform distribution. Secondly, even though the joint distribution is measured, it could still be the case that one of the target variables is random, as long as the other variable shows a high clustering tendency. For these reasons, a thorough investigation of this approach is needed to see how well it finds the most well-behaving cars. However, this investigation is not part of this project, as the method still removes the most irregular cars.

5.3.4 Training and Test Split

In Section 3.4, we cover the process of selecting features and hyperparameters, influenced by the off-line approach covered in [5]. In this project, we use the data from a subset of the well-behaving cars to select features and hyperparameters and another subset of well-behaving cars to test the final performance of the models. Importantly, our process assumes that the optimal features and hyperparameters found from the first subset of the well-behaving cars hold for all well-behaving cars.

Further, since we aggregate the results when determining the optimal selection, we are not making the best solution per car. Instead, one could follow the second approach presented in [5], which in our case would imply selecting the optimal features and hyperparameters individually per car, using the first 20% of the observations. However, we refrained from doing so in this project, with the main reason being the limited amount of data per car. Further, we argue that our approach is sufficient enough for finding acceptable features and hyperparameters.

5.3.5 Feature Selection

In this study, we use domain knowledge when deciding what features to consider in the prediction problem. However, as we do not know which the most valuable features are, we include as much information as possible, to help the models find patterns in the data. However, simply introducing multiple features that may be uninformative for the prediction could worsen the predictions, as these features may act as noise for the prediction models. Furthermore, we need to consider multicollinearity since multiple features originate from a single signal, e.g., mean and median of the speed during a driving session, and might thus be heavily correlated. Consequently, we perform a feature selection to avoid these issues.

The procedure of first adding several features and then removing them may seem counterintuitive. Nevertheless, we believe this procedure investigates potential patterns in features, and combinations of features, in an adequate manner. One could view this procedure as a broad search of valuable features for each prediction model.

On a different note, instead of reducing the features as explained in Section 3.4.1, one could consider dimensionality reduction, e.g., Principal Component Analysis (PCA). However, performing PCA may remove the interpretability of the models, making it more complicated to know which features to consider when making similar predictions.

5.3.6 Hyperparameter Selection

The hyperparameter selection, described in Section 3.4.2, compares different hyperparameter configurations to find the one yielding the lowest MAE across all cars. We use grid search to find the optimal configuration, and this approach requires a predefined search space of hyperparameters to test. Grid search is a simple way of selecting hyperparameters; however, it can be relatively inefficient. Furthermore, there is always a risk that the optimal configuration is not in the predefined search space and that a suboptimal configuration is selected instead. There are other approaches to hyperparameter selection that are quicker and might find a more suitable configuration. For example, we could have utilized the more advanced approach of Bayesian optimization to reduce computational time and potentially find a better configuration [51]. However, grid search is a simple method that produces reasonable parameters, and we use it due to the project's limited amount of time.

6

Conclusion and Future Work

In this study, we attempted to predict TD and TTL using incremental learning models of different complexity. The prediction models were evaluated according to their error rate and how the energy consumption from a drive was affected when adapting the battery thermal preconditioning to the models' predictions. Based on this, we attempted to answer the research questions stated in Section 1.3:

- How accurate can incremental learning models predict the departure time and distance of an upcoming drive?
- How sensitive is the performance of the battery thermal preconditioning to incorrect predictions?

To answer how accurate the incremental learning models predict TTL and TD, we considered the two approaches proposed in Section 3.1. Using the midnight approach, the proposed prediction models yield an aggregated MAE of slightly below 3 hours when predicting TTL and 13-15 km when predicting TD, with QARF and QKNN displaying the best predictive performance. Using the charging approach, the proposed prediction models yield an aggregated MAE of slightly below 6 hours when predicting TTL, and 23-27 km when predicting TD. Again, QARF and QKNN display the best performance. The models' ability to quantify uncertainty varies, with FFNN and QARF generally estimating the most accurate prediction intervals. The results indicate the surprising difficulty of predicting vehicle usage. The prediction intervals span tens of kilometers or several hours, and the models often resort to predicting some sort of average or the most common drive.

For the distances, temperatures, and chosen α considered in this study, the sensitivity analysis demonstrates that the performance of the battery thermal preconditioning process is sensitive to incorrect predictions. The energy that might be lost from an incorrect prediction is greater than what can be saved by an accurate prediction. This indicates that performing battery thermal preconditioning according to predictions requires high accuracy to be more beneficial than never preconditioning, and the qualitative evaluation shows that the prediction models considered in this work are not accurate enough.

6.1 Future Work

In this project, we use incremental learning since it fits well with the setting of the problem and potentially allows us to implement the models on the cars. Our prediction models yield similar error rates as the batch-learning models used in related work, but as the projects use different data sources, it is not possible to conclude if incremental models are as accurate. It would be interesting to compare incremental learning with batch learning in terms of MAE and how it affects energy consumption. Further, one could also investigate different approaches to incremental learning, e.g., the second approach proposed in [5].

One aspect of incremental learning that we briefly mention is concept drift. The problem of concept drift is accounted for when using QARF, and potentially QKNN, but whether this is needed is not thoroughly investigated. Therefore, it could be interesting to analyze this in future work. A potential course of action to account for concept drift using QR and FFNN is to use the ADWIN algorithm [21]. Further, one can include a concept drift when generating data to thoroughly investigate how the prediction models cope with concept drift of different strength.

Figure 4.1 shows, for different ambient temperatures, the driving distance needed for battery thermal preconditioning to be beneficial. These distance thresholds are based on the assumption that the battery is heated to a fixed target temperature. However, one could instead use a predicted trip distance to calculate an optimal target temperature. Future work could investigate how using a predicted optimal target temperature would affect the energy consumption, and whether this would affect the sensitiveness of the battery thermal preconditioning.

One last aspect that could be further improved is the predictive performance. In this project, QARF and QKNN yield the lowest error rates. However, as can be seen in Figure 5.1, all of the models sometimes struggle to differentiate an early trip from a later trip, and instead predicts some average departure time. Potential reasons for this behavior could be a lack of patterns in the data, or that the models struggle to find the patterns in the data. From this, there are two clear possibilities for improvements: Firstly, other machine learning models can be investigated. In fact, during the initial phases of this project, a simple Recurrent Neural Network (RNN) was tested. This did, however, display slow learning and poor predictive performance. For this reason, and because related work indicated that neural networks struggle with similar tasks, we decided to leave out the RNN of the project. However, plenty of other models can be investigated, e.g., transformers (self-attention models) which yield high predictive performance in the natural language processing domain.

Secondly, more information could be added by providing more informative features or making certain features more informative. In a completely different time series problem, stock price prediction, it is said that short-term fluctuations of stock prices are considered random and unpredictable according to the *Random Walk Hypothesis* [52]. As an attempt to capture the fluctuations, external dependencies of the stock market have been modeled, such as including financial news as input to machine

learning models [53, 54]. If we relate this to our problem, we could potentially achieve better accuracy in the predictions if external data are included. Shahriar et al. show that road conditions and local events may affect the driving behavior of a particular trip [16]. Additional features, such as GPS data and calendar data, could thus potentially give some indication on which trip is to be performed, assuming that we account for privacy and ethical concerns. Further, taking inspiration from the natural language processing domain, the categorical features (one-hot encodings) introduced in this work could be encoded using embedding layers, typically used for encoding words. This could potentially model the relations between the categorical features, further indicating the type of trip that is to be performed.

Bibliography

[1] European Comission, "Delivering the European Green Deal," [Online]. Available: https://ec.europa.eu/info/strategy/priorities-2019-2024/european-green-deal/delivering-european-green-deal_en (last accessed on 2021-12-10).

[2] Department for Business, Energy & Industrial Strategy and Department for Transport, "COP26 declaration on accelerating the transition to 100% zero emission cars and vans," 2022, [Online]. Available: https://www.gov.uk/government/publications/cop26-declaration-zero-emission-cars-and-vans/cop26-declaration-on-accelerating-the-transition-to-100-zero-emission-cars-and-vans (last accessed on 2022-06-02).

[3] Office of Governor Gavin Newsom, "Governor Newsom Announces California Will Phase Out Gasoline-Powered Cars & Drastically Reduce Demand for Fossil Fuel in California's Fight Against Climate Change," 2020, [Online]. Available: https://www.gov.ca.gov/2020/09/23/governor-newsom-announces-california-will-phase-out-gasoline-powered-cars-drastically-reduce-demand-for-fossil-fuel-in-californias-fight-against-climate-change/ (last accessed on 2022-06-02).

[4] IEA, "Trends and developments in electric vehicle markets," 2021, [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2021/trends-and-developments-in-electric-vehicle-markets (last accessed on 2022-03-02).

[5] V. Losing, B. Hammer, and H. Wersing, "Incremental on-line learning: A review and comparison of state of the art algorithms," *Neurocomputing*, vol. 275, pp. 1261–1274, Jan 2018, doi: 10.1016/j.neucom.2017.06.084.

[6] M. Auer, T. Kuthada, N. Widdecke, and J. Wiedemann, "Increase in range of a battery electric vehicle by means of predictive thermal management," in *15. Internationales Stuttgarter Symposium*. Wiesbaden: Springer Fachmedien Wiesbaden, May 2015, pp. 1495–1508, doi: 10.1007/978-3-658-08844-6_104.

[7] C. Kreutzer, J. Rugh, and J. Tomerlin, "Thermal Load Reduction System Development in a Hyundai Sonata PHEV," in *SAE World Congress*, Detroit, Michigan, 2017, doi: 10.4271/2017-01-0186.

[8] E. Samadani, M. Mastali, S. Farhad, R. A. Fraser, and M. Fowler, "Li-ion battery performance and degradation in electric vehicles under different usage scenarios," *International Journal of Energy Research*, vol. 40, no. 3, pp. 379–392, 2016, doi: 10.1002/er.3378.

[9] J. Lindgren and P. D. Lund, "Effect of extreme temperatures on battery charging and performance of electric vehicles," *Journal of Power Sources*, vol. 328, pp. 37–45, 2016, doi: 10.1016/j.jpowsour.2016.07.038.

[10] O. Frendo, N. Gaertner, and H. Stuckenschmidt, "Improving smart charging prioritization by predicting electric vehicle departure time," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 10, pp. 6646–6653, 2021, doi: 10.1109/TITS.2020.2988648.

[11] S. Baghali, S. Hasan, and Z. Guo, "Analyzing the travel and charging behavior of electric vehicles - a data-driven approach," in *2021 IEEE Kansas Power and Energy Conference (KPEC)*, 2021, pp. 1–5, doi: 10.1109/KPEC51835.2021.9446240.

[12] B. Hilpisch, "Construction of a user-level probabilistic forecast of electric vehicle usage," Semester Thesis, Institute of Cartography and Geoinformation, Swiss Federal Institute of Technology (ETH) Zurich, Zurich, Switzerland, 2020. doi: 10.3929/ethz-b-000460506.

[13] C. Goebel and M. Voß, "Forecasting driving behavior to enable efficient grid integration of plug-in electric vehicles," in *2012 IEEE Online Conference on Green Communications (GreenCom)*, 2012, pp. 74–79, doi: 10.1109/GreenCom.2012.6519619.

[14] D. Panahi, S. Deilami, M. A. S. Masoum, and S. M. Islam, "Forecasting plug-in electric vehicles load profile using artificial neural networks," in *2015 Australasian Universities Power Engineering Conference (AUPEC)*, 2015, pp. 1–6, doi: 10.1109/AUPEC.2015.7324879.

[15] H. Jahangir, S. S. Gougheri, B. Vatandoust, M. A. Golkar, A. Ahmadian, and A. Hajizadeh, "Plug-in electric vehicle behavior modeling in energy market: A novel deep learning-based approach with clustering technique," *IEEE Transactions on Smart Grid*, vol. 11, no. 6, pp. 4738–4748, 2020, doi: 10.1109/TSG.2020.2998072.

[16] S. Shahriar, A. R. Al-Ali, A. H. Osman, S. Dhou, and M. Nijim, "Prediction of ev charging behavior using machine learning," *IEEE Access*, vol. 9, pp. 111 576–111 586, 2021, doi: 10.1109/ACCESS.2021.3103119.

[17] X. Geng and K. Smith-Miles, *Incremental Learning*. Boston, MA: Springer US, 2009, pp. 731–735. [Online]. Available: https://doi.org/10.1007/978-0-387-73003-5_304

[18] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01418129

[19] D. Mukherjee and S. Datta, "Incremental time series algorithms for iot analytics: An example from autoregression," in *Proceedings of the 17th International Conference on Distributed Computing and Networking*, ser. ICDCN

'16. New York, NY, USA: Association for Computing Machinery, 2016, doi: 10.1145/2833312.2849556.

[20] L. Cheng, D. Schuurmans, S. Wang, T. Caelli, and S. Vishwanathan, "implicit online learning with kernels," in *Advances in Neural Information Processing Systems*, vol. 19. MIT Press, 2006. [Online]. Available: https://proceedings.neurips.cc/paper/2006/file/a92c274b8be496fb05d95033552eeddd-Paper.pdf

[21] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM)*, 2007, pp. 443–448. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611972771.42

[22] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, 2017, doi: 10.1007/s10994-017-5642-8.

[23] H. M. Gomes, J. P. Barddal, L. E. Boiko, and A. Bifet, "Adaptive random forests for data stream regression," *ESANN 2018 - Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2018. [Online]. Available: https://www.researchgate.net/publication/329397374_Adaptive_random_forests_for_data_stream_regression

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[25] Y. Chen, K. S. S. Alamin, D. J. Pagliari, S. Vinco, E. Macii, and M. Poncino, "Electric vehicles plug-in duration forecasting using machine learning for battery optimization," *Energies*, vol. 13, 8 2020, doi: 10.3390/en13164208.

[26] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, ser. Springer series in statistics. Springer, 2009. [Online]. Available: https://books.google.se/books?id=eBSgoAEACAAJ

[27] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[28] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceeding of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, doi: 10.1145/347090.347107.

[29] E. Ikonomovska, J. Gama, and S. Džeroski, "Learning model trees from evolving data streams," *Data Mining and Knowledge Discovery*, vol. 23, pp. 128–168, 07 2011, doi: 10.1007/s10618-010-0201-y.

[30] D. Boulegane, A. Bifet, H. Elghazel, and G. Madhusudan, "Streaming time series forecasting using multi-target regression with dynamic ensemble selection," in *2020 IEEE International Conference on Big Data (Big Data)*, 12 2020, pp. 2170–2179, doi: 10.1109/BigData50022.2020.9378264.

[31] N. Tagasovska and D. Lopez-Paz, "Single-model uncertainties for deep learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/73c03186765e199c116224b68adc5fa0-Paper.pdf

[32] Y. Gal. Uncertainty in deep learning. Ph.D. dissertation, Department of Engineering University of Cambridge, 2016. [Online]. Available: https://mlg.eng.cam.ac.uk/yarin/thesis/thesis.pdf

[33] L. Zhu and N. Laptev, "Deep and confident prediction for time series at uber," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2017, pp. 103–110, doi: 10.1109/ICDMW.2017.19.

[34] Y. Lai, Y. Shi, Y. Han, Y. Shao, M. Qi, and B. Li, "Exploring uncertainty in regression neural networks for construction of prediction intervals," *Neurocomputing*, vol. 481, pp. 249–257, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222001102

[35] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik, "Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation," *Computational Statistics & Data Analysis*, vol. 142, p. 106816, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016794731930163X

[36] M. H. Shaker and E. Hüllermeier, "Aleatoric and epistemic uncertainty with random forests," in *Advances in Intelligent Data Analysis XVIII*. Cham: Springer International Publishing, 2020, pp. 444–456, doi: 10.1007/978-3-030-44584-3_35.

[37] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A comprehensive survey of loss functions in machine learning," *Annals of Data Science*, vol. 9, no. 2, pp. 187–212, 2022, doi: 10.1007/s40745-020-00253-5.

[38] R. Koenker and K. F. Hallock, "Quantile regression," *Journal of economic perspectives*, vol. 15, no. 4, pp. 143–156, December 2001, doi: 10.1257/jep.15.4.143.

[39] T. Vasiloudis, G. D. F. Morales, and H. Boström, "Quantifying uncertainty in online regression forests." *Journal of machine learning research*, vol. 20, no. 155, pp. 1–35, 2019. [Online]. Available: http://jmlr.org/papers/v20/19-006.html

[40] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059. [Online]. Available: https://proceedings.mlr.press/v48/gal16.html

[41] A. Blum, A. Kalai, and J. Langford, "Beating the hold-out: Bounds for k-fold and progressive cross-validation," in *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, ser. COLT '99. New

York, NY, USA: Association for Computing Machinery, 1999, p. 203–208, doi: 10.1145/307400.307439.

[42] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Lower upper bound estimation method for construction of neural network-based prediction intervals," *IEEE Transactions on Neural Networks*, vol. 22, no. 3, pp. 337–346, 2011, doi: 10.1109/TNN.2010.2096824.

[43] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*, 5th ed. United States of America: John Wiley & Sons, 2010, pp. 502–503.

[44] G. Cross and A. Jain, "Measurement of clustering tendency**research supported in part by nsf grant ecs-8007106." in *Theory and Application of Digital Control*, A. MAHALANABIS, Ed. Pergamon, 1982, pp. 315–320, doi: 10.1016/B978-0-08-027618-2.50054-1.

[45] S. P. Vasseur and J. L. Aznarte, "Comparing quantile regression methods for probabilistic forecasting of no2 pollution levels," *Scientific Reports*, vol. 11, no. 1, pp. 1–8, 05 2021, doi: 10.1038/s41598-021-90063-3.

[46] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, and J. a. Gama, "Machine learning for streaming data: State of the art, challenges, and opportunities," *ACM SIGKDD Explorations Newsletter*, vol. 21, no. 2, p. 6–22, nov 2019, doi: 10.1145/3373464.3373470.

[47] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed. Melbourne, Australia: OTexts, 2018, accessed on 2022-03-21. [Online]. Available: https://otexts.com/fpp2/

[48] A. Bhandari. What is Multicollinearity? Here's Everything You Need to Know . Analytics Vidhya. [Accessed 2022-03-28]. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/03/what-is-multicollinearity/

[49] L. Zhao, "Event prediction in the big data era: A systematic survey," *ACM Comput. Surv.*, vol. 54, no. 5, may 2021. [Online]. Available: https://doi.org/10.1145/3450287

[50] T. Zhu, R. G. Wills, R. Lot, X. Kong, and X. Yan, "Optimal sizing and sensitivity analysis of a battery-supercapacitor energy storage system for electric vehicles," *Energy*, vol. 221, p. 119851, 2021, doi: 10.1016/j.energy.2021.119851.

[51] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

[52] J. C. V. Horne and G. G. C. Parker, "The random-walk theory: An empirical test," *Financial Analysts Journal*, vol. 23, no. 6, pp. 87–92, 1967, doi: 10.2307/4470248.

[53] J. Liu, Y. Chen, K. Liu, and J. Zhao, "Attention-based event relevance model for stock price movement prediction," in *Knowledge Graph and Semantic Computing. Language, Knowledge, and Intelligence*. Singapore: Springer Singapore, 2017, pp. 37–49, doi: 10.1007/978-981-10-7359-5_5.

[54] M. R. Vargas, C. E. M. dos Anjos, G. L. G. Bichara, and A. G. Evsukoff, "Deep leaming for stock market prediction using technical indicators and financial news articles," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8, doi: 10.1109/IJCNN.2018.8489208.