CHALMERS
UNIVERSITY OF TECHNOLOGY

Explainable machine learning for state-of-health prediction using electric vehicle histogram data

Master's thesis in Data Science and AI & Computer Science

ERIC CARLSSON
OSKAR HELLQVIST

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Explainable machine learning for state-of-health prediction using electric vehicle histogram data

ERIC CARLSSON
OSKAR HELLQVIST

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2022

Explainable machine learning for state-of-health prediction using electric vehicle histogram data
ERIC CARLSSON
OSKAR HELLQVIST

Industry supervisor: Herman Johnsson, Volvo Cars, Research & Development department, Analytics & AI
Examiner: Torsten Wik, Department of Electrical Engineering

Master's thesis 2022
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2022

Explainable machine learning for state-of-health prediction using electric vehicle histogram data

ERIC CARLSSON
OSKAR HELLQVIST
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The popularity of Electric Vehicles (EVs) is rapidly growing, owning much to advancements made in battery technology over the recent years. However, a key problem that remains for automotive manufacturers and prospective EV buyers is that batteries age with time and use. A number of different methods have previously been used to predict battery State-Of-Health (SOH), ranging from detailed electrochemical models to semi-empirical parametric models.

With the increased availability of large-scale vehicle data, statistical models have also been applied to the problem. Gathering this data is not without problems however. Privacy concerns and hardware limitations are just two reasons why manufacturers have had to come up with novel methods of data collection and aggregation. One such method that has been commonly used in Volvo cars is to accumulate data over time into histograms. This method has some obvious side effects in terms of data shape and resolution, but it is not known how it affects the underlying data distributions, or statistical models derived therefrom.

This thesis aims to investigate how SOH prediction models are affected by the novel data collection and aggregation method that is commonly used for high-voltage battery signals in Volvo cars. This is done using a multi-step process, in which a dataset consisting of measurements from a large fleet Volvo Plug-in Hybrid Electric Vehicles (PHEVs) is first pre-processed, after which it is used to train statistical machine learning models. The models are then evaluated and examined using feature importance and Shapley Additive Explanations (SHAP). In addition to this, a method for simulating the data collection method used in the vehicles is presented. This makes it possible to create datasets similar to real vehicle data from publicly available time series datasets.

The results indicate that the novel data collection and aggregation method affects the internal behaviours of statistical SOH prediction models. The exact cause of this is not established, but the results indicate that strong feature correlation is a contributing factor. The results also highlights the potential of model-agnostic explainability methods as guiding tools for data engineering and modelling, even when applied to models trained on highly novel datasets.

Keywords: electric vehicles, state-of-health, battery ageing, battery degradation, machine learning, explainability, interpretability, feature importance, shapley values, shapley additive explanations

Acknowledgements

First of all we would like to express our gratitude to our industry supervisor Herman Johnsson, for giving us the opportunity to be a part of the project, and for being a continuous source of inspiration and support. His advice and expertise has been invaluable to us, and to the outcome projects. We would also like to thank our examiner Torsten Wik, who enthusiastically took on our project early on when it was still in its infancy.

Lastly, we would like to thank the remaining members, and fellow master thesis students, of Volvo Cars R&D Analytics and Artificial Intelligence, for welcoming us to the department and supporting our project. Without them, our time working on the project would not have been nearly as enjoyable.

Eric Carlsson & Oskar Hellqvist, Gothenburg, December 2022

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
BEV	Battery Electric Vehicle
ECU	Electronic Control Unit
EV	Electric Vehicle
ICE	Internal Combustion Engine
KDE	Kernel Density Estimation
MAE	Mean Absolute Error
OCV	Open-Circuit Voltage
PDF	Probability Density Function
PHEV	Plug-in Hybrid Electric Vehicle
RMSE	Root Mean Square Error
SHAP	Shapley Additive Explanations
SOC	State-Of-Charge
SOH	State-Of-Health
SPA	Scalable Product Architecture
SSR	Sum of Squared Residuals
XAI	Explainable Artificial Intelligence
XGBoost	Extreme Gradient Boosting

1 | Introduction

The popularity of Electric Vehicles (EVs) has been rapidly growing over the past few years, as they stand to offer a more environmentally friendly alternative, with higher efficiency and lower maintenance costs, compared to their Internal Combustion Engine (ICE) counterparts. By current projections, EVs will continue growing in market share in both the short- and long-term, and many automotive manufacturers are electrifying large parts of their fleets [1].

Batteries, a component that is critical to the function of EVs, remains an expensive part, even if advancements in battery technology and mass manufacturing techniques are working to push the overall prices of EVs down [1]. The longevity of EVs has also been put to discussion, as batteries degrades with time and use. This phenomenon is often referred to as *battery ageing* and happens as a result of various mechanical and electrochemical processes within the battery cells. This effectively limits the lifespan of EV batteries, after which they have to be swapped out through a process that is both costly and time consuming [2].

Previous estimates show that the service life of EV batteries has to reach 10-15 years, in order to make EVs cost competitive compared to ICE vehicles [2]. This means that being able to track and predict the effective health of batteries is an important task for both manufacturers and owners of EVs.

State-Of-Health (SOH) is a metric that is commonly used to measure the degree of battery ageing. Predicting battery SOH presents a major challenge, as the mechanisms that are responsible for it are highly complex and cannot be easily observed or measured when the battery is in use. A variety of methods have been used to tackle the problem previously, such as physical models or empirical data fitting models. With the increased availability of data, various statistical methods have also been used [2].

Many modern vehicles are highly digitized, and contain a large number of interconnected computers known as Electronic Control Units (ECUs). The ECUs can be used to monitor various parts of the vehicle when it is in active use or parked, and make out the backbone of modern vehicle data collection. They are capable of sampling sensory data in real time, and may also perform on-board data processing in some cases.

Battery data in particular is typically accumulated into data structures that can be interpreted as multi-dimensional histograms. Snapshots of these histograms can

then be collected when the vehicle visits a service center. There are both up- and downsides to this data collection technique, as it may for example improve end-user privacy, but also results in lower data resolution and a highly unique data format that is different from most publicly available battery usage datasets. With a very limited number of exceptions, previously published studies into SOH prediction for automotive applications do not consider this phenomena.

Complex machine learning models have proven capable of solving a wide variety of tasks. Not only are they highly adaptable to different problems, they are in many cases capable of outperforming even human intelligence. However, in comparison to humans, such models are typically far worse at explaining how they work and how they arrive at specific decisions [3, Sec. 2.2]. These types of models are often referred to as "black box" models, owning to their highly opaque nature [3, Sec. 3.1].

Black box models can be problematic, as model interpretability is often as important as model performance [3, Sec. 3.1]. Over the years, methods aimed at explaining otherwise opaque models have been developed. Some of these are specific to certain types of models, called model-specific methods, but others do not make any assumption about the inner working of the models, called model-agnostic methods [3, Sec. 3.2].

This thesis aims to create and analyze statistical SOH prediction models using modern model-agnostic explainability methods. In particular, it is of interest to see how such explanations relate to factors that are widely known to affect the rate of battery ageing. The primary dataset that is used in this thesis consists of measurements from over 65 000 Plug-in Hybrid Electric Vehicles (PHEVs), which is by far the largest dataset that has been used for this purpose to the authors' knowledge. An emphasis is made on how the data format, that is largely proprietary to the automotive industry, affects model training, performance and interpretability.

In the following sections, a brief background is given, along with the project scope and a summary of previous related work. Finally, the structure of the rest of the report is explained.

1.1 Background

The Volvo Car Group (Volvo Cars) is a Swedish premium automotive manufacturer that develops and manufactures a variety of passenger cars under the Volvo and Polestar brands. Volvo Cars' main selling points are safety and sustainability, and hence, the company is heavily invested in the rapidly expanding EV market. PHEVs have been a part of the Volvo Cars product line for a long time, and the company recently released their first pure Battery Electric Vehicle (BEV). Volvo Cars aims to become fully electric, and only produce EVs by 2030 [4].

Battery SOH prediction is a problem with a range of potential business applications, and hence, has been worked on previously at Volvo Cars. In the past, SOH prediction was often done using semi-empirical models based on data collected under controlled

conditions. With the increased availability of real-world vehicle data that has been made possible through technological advancements, attempts have also been made to use purely statistical models. The potential benefits of such models is that they can be used without any prior knowledge, and can be fit to actual vehicle data that reflects true operational conditions.

In modern Volvo cars it is possible to sample signals from sensors connected to the ECU network at a high resolution, but for a variety of reasons this data may not always be collected directly. For example, battery signals are typically accumulated into a complex format prior to being stored for later collection. This format can be interpreted as discrete distributions, or histograms, that are accumulated over the lifetime of the vehicle. The histograms are in general multi-dimensional, where each dimension corresponds to its own unit. The histogram bins are set to fixed values, but the sizes of these are in general not fixed. Snapshots of the histograms are collected digitally during service visits. Figure 1.1 shows an overview of the data flow from raw signal measurements to collection and storage.

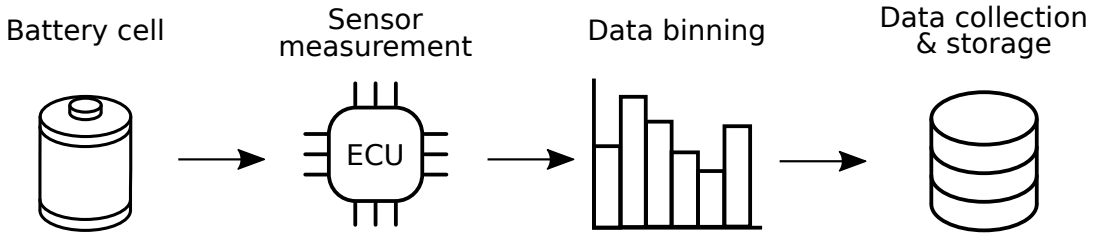


Figure 1.1: Overview of how vehicle battery data is measured, processed and collected. Sensors measure raw signals from the battery cells, which are collected and processed by an ECU. The ECU accumulates the data into histograms over the vehicle's lifetime. Snapshots of the histograms can then be collected during service visits and stored off-board.

Since Volvo cars are typically recommended to visit the workshop once a year, the data collection method has the effect of resulting in relatively few measurements per vehicle. This, together with the growing number of new cars that were added to the fleet over the course of the data collection period, means that most measurements are of young vehicles. It is unclear how the proprietary data format and the imbalanced data distributions affects the prediction models, and so it is of interest to examine what impact it has on the models' behaviour and performance.

1.2 Previous work

Several studies aimed at developing methods for SOH prediction have previously been conducted, encompassing a variety of different methods and approaches [2]. At its core, SOH prediction concerns the electrochemical processes that occur within batteries, and hence, many models are based entirely or partially on physical modelling.

Various types of statistical models have also been applied to the problem, ranging from simpler methods such as Autoregressive Moving Average to more complex methods such as neural networks [2]. It is important to note that the above mentioned studies are typically done using data that has been generated under controlled conditions, where it is possible to reliably measure, collect and store data on fixed intervals. This is not the case when collecting battery data from vehicles, and hence, studying battery ageing in vehicles presents a set of additional challenges.

As an EV manufacturer, Volvo Cars has also invested considerable time and resources into SOH prediction models. In a thesis project from 2020 [5], Chanchaipol and Sirikul showed the feasibility of applying statistical prediction models to data of a similar type using principal component analysis. This approach reduces the complexity of the problem by dimensionality reduction, but has the downside of making model explanations much less interpretable.

Statistical SOH prediction models have also been a key area within the Analytics & Artificial Intelligence department at Volvo Cars previously. Not only has work been done to collect, clean and prepare data, there has also been work done on developing models for predicting future SOH given a recollection of past data. The outcome of this research showed that histogram vehicle data can be used for this purpose, but the models were not sufficiently accurate given the data that was available at the time. This project targets a different, but similar, problem. As a result, this project draws much inspiration from the previous work conducted at Analytics and AI. In particular, since the same dataset is used, much of the data pre-processing is heavily inspired by what was proven to work previously.

Finally, in a publication from 2022 [6], Zhang et al. developed a machine learning framework for online SOH prediction. When trained on real vehicle histogram data, the models achieved performance comparable to that of models trained on lab generated time series data.

1.3 Project scope

This project aims to investigate how statistical SOH prediction models, trained on a novel dataset consisting of measurements from a large fleet of electrical vehicles, can be interpreted using modern interpretability methods. Furthermore, the impact of the data collection and processing methods, that are to the authors' knowledge largely unique to the automotive industry, is examined.

This project is done in collaboration with Volvo Cars, and as a result, all vehicle data that is used was collected from Volvo vehicles. A secondary, publicly available, battery ageing dataset is also used. The number of modelling techniques used in this project is limited, as the primary goal of the project is not to optimize the prediction performance.

To facilitate model analysis, two different types of explainability methods were selected. One of these methods is directly tied to the choice of models, while the other is model-agnostic in general, even if the specific implementation that is used makes

assumptions about the model. Hence, different implementations of the method could be used to reproduce the results with different types of models.

1.4 Thesis structure

The rest of this thesis is structured as following: In Chapter 2, the theoretical background and concepts that are required in order to follow this thesis are presented. In Chapter 4, the experimental approach is presented and the usage of specific methods is discussed. In Chapter 5, the results obtained from the experiments presented in the previous chapter are presented. In Chapter 6, the results are discussed, Finally, in Chapter 7, the results and discussion are summarized, and some ideas concerning potential future work are presented.

2 | Theory

In the following chapter, the theoretic background that is required to follow the rest of the report is presented. In Section 2.1, a brief background on the functionality, properties and ageing behaviour of batteries is provided. In Section 2.2 an introduction to machine learning and the specific methods that are used in this project are presented. In Section 2.3, Explainable AI is described in general, followed by the specific explainability methods that were employed in this project. Finally, in Section 2.4 a method for weighting datasets, called DenseWeight, is presented.

2.1 Batteries

Lithium-ion batteries have been widely used in consumer electronics since the 1990's, and are today commonly found in electrical vehicles owning to their comparatively high energy density and low weight. The main limitation of lithium-batteries when used in such a context relates to their ageing behaviour, which limits their performance with usage and time. This performance degradation affects various properties of the batteries, including their capacity and internal resistance [2].

2.1.1 Battery architecture

Most batteries share the same fundamental components. Each battery cell consist of two electrodes, the anode and the cathode, which are separated by an electrically non-conductive separator material that prevents short circuiting. When the battery is discharged, current is drawn from the anode via a negative current collector, to the cathode via a positive current collector. The electrodes are surrounded by an electrolyte which allows for the transfer of ions in order to keep the internal charge balanced. Some batteries may be recharged by applying a higher, inverse, voltage over the electrodes [7].

The lithium-ion battery get its name from its use of Li^+ ions as charge carriers [7], and refers to not a single type of battery, but a family of cell chemistries that all use lithium-ions as charge carriers. Figure 2.1 shows a diagram of a generic lithium-ion battery during its discharge cycle.

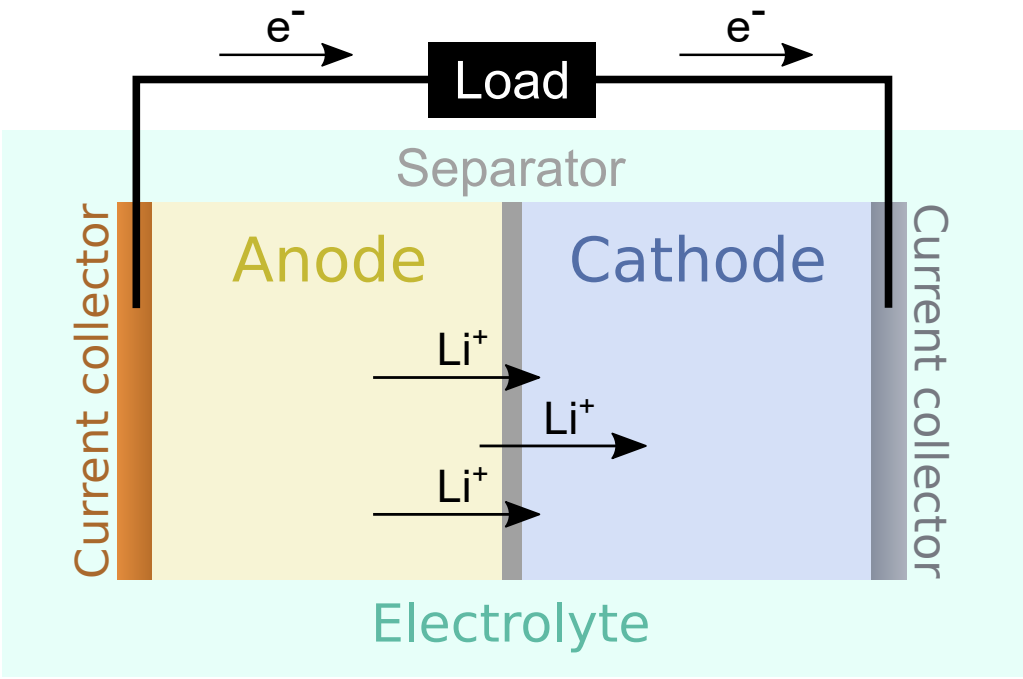


Figure 2.1: Illustration of a lithium-ion battery during its discharge cycle.

The lithium-ion batteries that are used in electric vehicles are made up of many individual battery cells that are connected together. The individual *battery cells* are grouped into *battery modules*, which in turn are stacked to create a full *battery pack*. Figure 2.2 shows how these components relate to each other. The exact cell chemistry and physical construction that is used in a battery pack differs between vendors and models, but they all work in largely the same way.

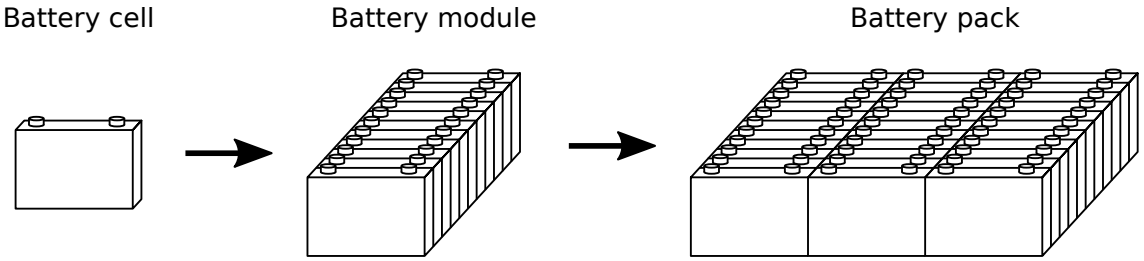


Figure 2.2: Overview of how battery cells, modules and packs relate to each other. Note that battery packs that are used in electric vehicles typically contain many more modules and battery cells than what is shown in the figure.

2.1.2 Battery degradation

Battery cells age with time and use, a behaviour that can typically be observed as an increase in internal impedance, resulting in the fade of deliverable power, and a decrease in nominal cell capacity [2]. A cell's current level of charge is often described by a metric known as State-Of-Charge (SOC), which is defined by

$$SOC(t,\tau) = \frac{Q(t)}{Q_n(\tau)}, \tag{2.1}$$

where $Q(t)$ is its currently available energy, $Q_n(\tau)$ is its nominal (rated) capacity, t is the time relative to the start of the current cycle, and τ is the time relative to the battery's beginning of life.

To describe a cell's nominal capacity fade, another metric called its State-Of-Health (SOH) is often used. SOH is defined by

$$SOH(\tau) = \frac{Q_n(\tau)}{Q_n^{max}}, \tag{2.2}$$

where Q_n^{max} is its maximum (initial) nominal capacity [2].

Cell degradation occurs as a result of various chemical and mechanical degradation mechanisms. The main consequences of these are the loss of usable lithium and electrode material, which results in a loss of charge capacity, and an increase in internal resistance, which results in a loss of available power [2]. The individual degradation mechanisms are fairly well understood, but are typically hard to observe when the battery is in use. Degradation systems are also made more complex as a result of strong coupling between degradation mechanisms [8].

The battery ageing behaviour can be further subdivided into two types of ageing. *Calendar ageing* causes an irreversible loss of capacity, and happens with time, regardless of actual battery usage. The rate of self discharge that occurs during storage is a driving factor for calendar ageing, and is in turn affected greatly by the battery storage conditions. It has been shown that the ambient temperature and the SOC at which the battery is stored has a major impact rate of calendar ageing [2].

Cycle ageing occurs during battery use, as it charges and discharges. As with calendar ageing, ambient temperature can be a major contributing factor to cycle ageing. The depth of discharge, sometimes denoted ΔSOC, denotes the change in SOC over a discharge cycle. An example of how depth of discharge is defined is shown in Figure 2.3. A high depth of discharge has been shown to increase the rate of ageing. Finally, higher charge and discharge voltages, and high current peaks, have been linked to an accelerated rate of degradation [2].

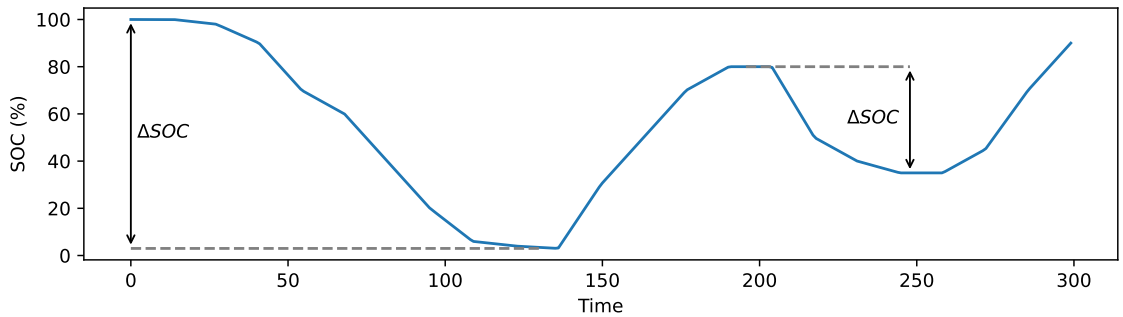


Figure 2.3: Example of how the depth of discharge, ΔSOC, is related to the change in SOC for different battery cycles.

As described in Section 2.1.1, battery packs typically contain many individual battery cells. Since the performance of a battery pack is largely limited by the performance of its worst cell, the SOH for the full battery pack is typically defined as the minimum SOH of the battery cells. In practice, the charge capacity is often estimated by integrating a measured current. This is then used to estimate both the SOC and SOH.

2.2 Machine learning

Machine learning is a field within Artificial Intelligence (AI), which focuses on systems that can learn to perform tasks without being explicitly programmed for them. Machine learning methods are widely used to tackle problems that static programs written by humans cannot solve, either due to sub-optimal performance or by being too time-consuming to create. There exists many different approaches to machine learning, but all are fundamentally a type of learning algorithm, and hence, shares the same framework consisting of a task, some experience and one or more performance measures [9, Sec. 5.1].

The *task* is the underlying problem that the machine learning algorithm is trying to solve. If the goal is to make a car drive, then the task is to *drive*. If the goal is to predict how far aged a battery is, the task is often specified as *predicting its SOH*. Many different types of tasks exist, but regression is of specific interest to the rest of this report [9, Sec. 5.1.1].

Regression is a type of task where the goal is to predict a numerical value. This is done by producing some function $f : \mathbb{R}^n \to \mathbb{R}$, that takes some n-dimensional input and outputs a numerical value. This function can then be used to predict the output corresponding to any set of inputs [9, Sec. 5.1.1].

In order to learn how to perform tasks, machine learning algorithms require some form of *experience* from which they can learn, a process that is typically referred to as *training*. Experience comes in the form of data points, or samples. Each data point consists of a set of feature values that in some way describe or characterize the task, and are typically represented as numerical vectors $\mathbf{x} \in \mathbb{R}^n$. Each vector dimension corresponds to its own, unique, feature [9, Sec. 5.1.3].

Data points are typically grouped into datasets, which are often described using *design matrices*. In a design matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, the rows correspond to data samples and the columns features. Depending on the learning approach, the experience may or may not include additional information besides the data points [9, Sec. 5.1.3].

In a *supervised learning* approach, a vector of labels is given in addition to the design matrix of feature values. The labels correspond to the true values, which is what the model should ideally learn to predict. The exact shape of the label vector can vary depending on the task, but for regression-type problems it is typically given as a simple numerical vector $\mathbf{y} \in \mathbb{R}^m$. In an unsupervised approach, no label vector is given and the model instead has to rely solely on learning patterns from the data points [9, Sec. 5.1.3]. As label data is available for the task examined in this project,

the rest of this chapter is presented under the assumption that a supervised learning approach is used.

To evaluate how good a machine learning model is at solving a given task, one or more *performance measures* are typically used that are usually unique to the task [9, Sec. 5.1.2]. For regression, the most common metrics are based on the difference between the predicted value $\hat{y}$ and the corresponding true label y, $y - \hat{y}$, also called the *residual*. This value is used in a *loss function* (also called a *cost function* or an *error function*) to compute a positive real number that represents how close a prediction is to the corresponding true label. Hence, the goal of a machine learning model is to minimize the loss function value [9, Sec. 5.5]. A number of different loss functions exist, but among the more common ones are the *Root Mean Square Error (RMSE)* and the *Mean Absolute Error (MAE)*. The RMSE is defined as

$$\mathrm{RMSE}(y, \hat{y}) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}, \tag{2.3}$$

where m is the number of data points, y_i is the true value corresponding to data point i and $\hat{y}_i$ is the predicted value for data point i [9, Sec. 5.5.1]. Similarly, the MAE is defined as

$$\mathrm{MAE}(y, \hat{y}) = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|, \tag{2.4}$$

where the variables are defined in the same way as for Equation 2.3 [10].

As mentioned above, the goal of machine learning models is typically to minimize the loss over the dataset that it is trained on. In reality however, it is often more important how well the model can *generalize*, that is, perform on previously unseen data. For this reason, multiple datasets are typically used, such that the model can be trained on one dataset and evaluated on a different dataset to measure how well it can generalize. These two datasets are typically called the *training set* and the *test set* [9, Sec. 5.1.2].

Most machine learning models have settings, or so called *hyperparameters*, that control how the model behaves [9, Sec. 5.3]. These hyperparameters are fixed prior to training, and since they can have a large impact of the model performance and behaviour, some method of selecting the best hyperparameter values is typically used. For this reason, the training set is typically further split into an additional dataset called the *validation set* [9, Sec. 5.3]. This dataset can be used to make decisions about hyperparameter values without affecting the test set.

There are many different approaches to hyperparameter selection, but since there is often an infinite amount of possible hyperparameter values, most methods require some degree of manual input. One of the more direct approaches to hyperparameter selection is to manually select a *search space* containing a set of possible values for each hyperparameter, and then testing all possible value combinations to find the best set of hyperparameter values.

In certain cases it may not be desirable to split the dataset into a fixed train and test (or validation) set, especially if the resulting datasets are small. The main reason for this is that the statistical uncertainty when evaluating models trained on these datasets will be high, making it difficult to determine which model is actually better.

One solution to this problem is to use cross-validation. The most common method of cross validation is k-fold cross-validation, which works by splitting the dataset into k non-overlapping subsets. The model is then trained k times, using one of the partitions as test set and the rest of the partitions combined as the training set each time. This is done such that each of the partitions is used as the test set once. The overall test loss is then estimated by aggregating the test loss computed for each of the k trained models, typically by computing an average [9, Sec. 5.3.1].

2.2.1 Decision trees

A decision tree is a type of machine learning method which works by dividing the input feature space into regions, such that each region corresponds to a specific output value. This is done iteratively through a number of nodes, starting from the root node of a tree structure, hence the name. At each level of the tree, the input feature space is further broken into multiple smaller, non-overlapping, sub-regions. The final nodes, corresponding to the leaves of the tree, are mapped to a specific output value [11, Sec. 9.2].

Figure 2.4 illustrates how a decision tree works. At each node there is a condition which is checked against the input, and the result is used to decide which node to go to next. If the condition evaluates to true, the left child is chosen, and if it evaluates to false, the right child is chosen. In this example, the decision tree is used to compute a prediction for $\mathbf{x} = (x_1, x_2) = (5, 2)$. By following the above-mentioned rules, a path is created starting from the top-most *root node*. The output value is decided by the *leaf node*, which is the final node in the path. The path is highlighted grey in figure. In this example, the prediction becomes $\hat{y} = 5$.

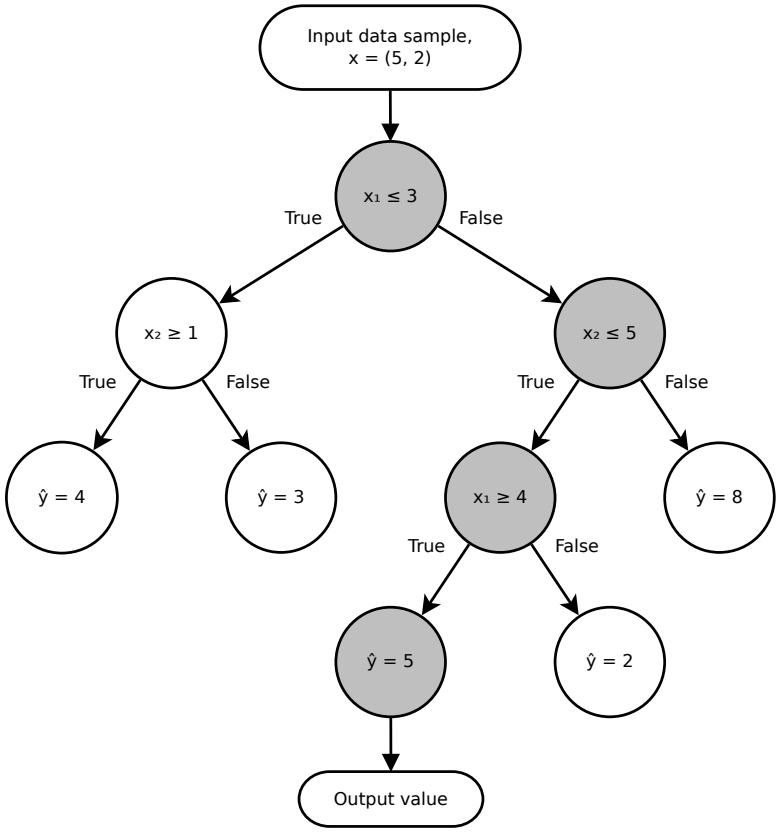


Figure 2.4: Example of how a decision tree works.

The accuracy of decision trees is entirely decided by their structure, as the algorithm for computing a prediction given an input is deterministic. For this reason, a greedy algorithm that tries to optimize the accuracy is used to grow the trees. Decision trees are constructed iteratively, starting from the root node. At each split, two major decisions have to be made: how to pick which variable the split should be performed on, and how to pick the threshold value [11, Sec. 9.2]. This typically done by minimizing some error function, and for regression trees it is common to use the Sum of Squared Residuals (SSR).

$$\mathrm{SSR}(y, \hat{y}) = \sum_{i=1}^{m} (\hat{y}_i - y_i)^2, \tag{2.5}$$

where m is the number of data points, y_i is the true value and $\hat{y}_i$ is the predicted value for data point i. Typically, the optimal threshold value for each feature is first chosen, after which the feature that gives the lowest SSR is chosen for the split [11, Sec. 9.2].

If the tree is allowed to grow infinitely deep, the greedy algorithm will produce a tree in which each training data sample has its own corresponding leaf node. As described in Section 2.2, the actual value of machine learning models lies within their ability to generalize. For this reason, the depth is typically selected as a hyperparameter and fixed prior to training [11, Sec. 9.2].

In addition to the tree depth, a variety of other hyperparameters may also be introduced to control the model behaviour and reduce overfitting. For example, the minimum residual reduction, or the number of samples that have to be considered to perform a split, may also be considered as hyperparameters.

2.2.2 Ensemble methods and forests

Simple models such as decision trees can be useful on their own, but tends to lack the capacity required to solve more complex tasks. In *ensemble methods*, the outputs of several models are combined to produce a single prediction. The idea of such methods is that the combination of multiple simpler models is typically capable of outperforming any one of its individual component models [11, Sec. 16.1].

In an ensemble of decision trees, occasionally called a *forest*, a number of decision trees are stacked horizontally. The input data is broadcast to each decision tree, such that each tree produces its own prediction independently of the other trees. The predictions are then combined using an aggregation function [11, Sec. 15.2]. For regression tasks, the aggregation function is typically selected as the average of all predictions. Figure 2.5 shows an example of a forest model works.

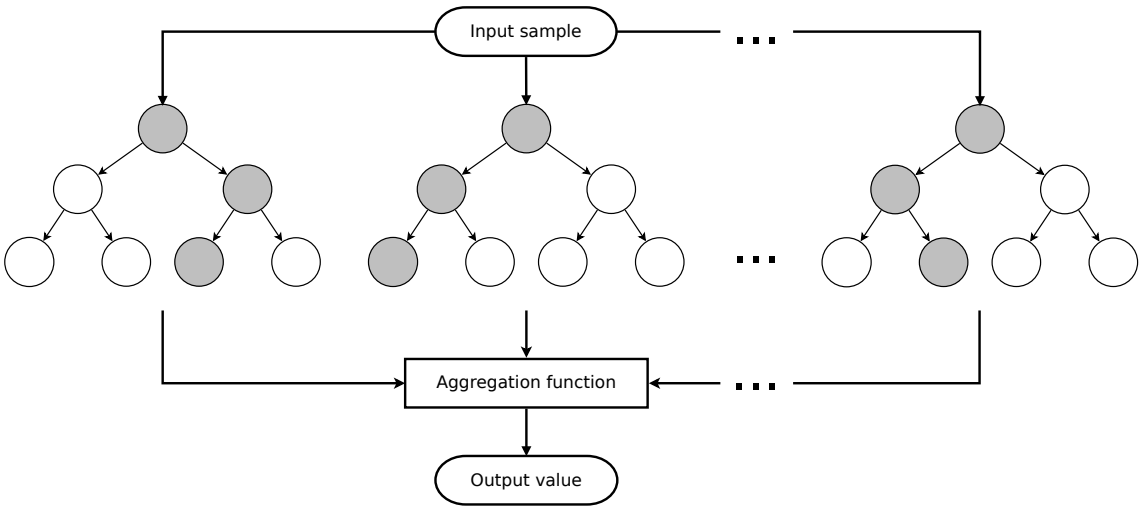


Figure 2.5: Overview of how an ensemble method using decision trees, also called a forest, works.

As when growing individual decision trees, the method of constructing forests has a major impact on the resulting model's performance. A naive approach to constructing forests would be to train each decision tree on the entire dataset. This would not result in a very useful ensemble model however, as all trees would end up being identical. This happens since the decision trees are grown using a greedy, and hence deterministic, algorithm, as described in Section 2.2.1.

Instead, each tree has to be grown from a unique subset of the dataset. In this project, two different models that are both based on forests, but employ different

data selection techniques are used. These are the Random Forest and Gradient Boosting algorithms.

2.2.3 Random Forest

Random Forest is a forest-type model that uses a technique based on *bootstrap aggregation*, or *bagging* to introduce variance among the decision trees. The algorithm works by randomly drawing samples with replacement from the training dataset to create a bootstrap dataset. A decision tree is then trained on the bootstrap dataset, and the procedure is repeated until the desired number of trees are created [11, Chap. 15].

The bootstrap datasets are typically made to be of equal size to the original training dataset, but since samples are drawn with replacement, each bootstrap dataset will contain only a subset of unique samples from the original dataset [11, Chap. 15]. Figure 2.6 shows a visual representation of the Random Forest algorithm.

The Random Forest algorithm solves the issue faced when training a forest of decision trees from a single dataset, described in Section 2.2.2, by introducing a source of randomness when sampling data. This method has been proven to work well, but perhaps there are other, more strategic, ways to approach the problem? In the next section, the Gradient Boosting algorithm is introduced, which aims to do exactly this.

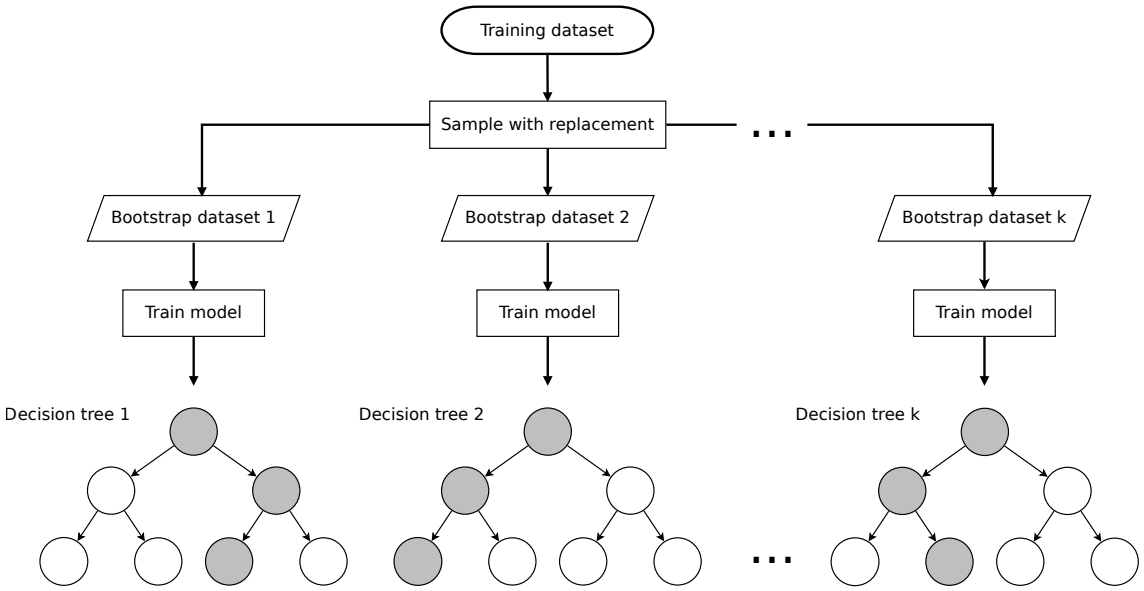


Figure 2.6: Overview of the Random Forest algorithm.

2.2.4 Gradient boosting

Gradient Boosting is another forest-type method, that uses a method called *boosting* to iteratively construct a forest by adding trees to compensate for the previous model's prediction error. The algorithm uses the well-known *Gradient Descent* algorithm to compute the training target, hence its name [12].

Gradient boosting is based on the concept of *Generalized Additive Models*, which is a type of model that is constructed as an additive expansion of unspecified functions, typically called *basis functions*. Additive models have the general form

$$\mathbf{y} \approx f(\mathbf{x}) = \alpha + \sum_{m=1}^{M} f_m(\mathbf{x}), \tag{2.6}$$

where α is a constant and $f_m(\mathbf{x}), m = 1, ..., M$ is a set of basis functions [11, Sec. 9.1].

α can be moved into any one of the basis functions, and let $f_m(\mathbf{x}) = \beta_m b(\mathbf{x}, \gamma_m)$, where β_m is an expansion coefficient and γ_m is the set of the basis function's parameters, for $m = 1, ..., M$. Boosting is a method for fitting such models by estimating β_m and γ_m in a sequential manner. A straight forward method of accomplishing this is with *Forward Stagewise Additive Modelling*, which works by, for each $m = 1, ..., M$, computing

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i, \gamma)), \tag{2.7}$$

where L is a generic loss function. The resulting parameters (β_m, γ_m) will hence fit a new basis function $\beta_m b(\mathbf{x}, \gamma_m)$ that compensates for the error of the previous model, $f_{m-1}(\mathbf{x})$. The next stage expansion is then simply set as

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \beta_m b(\mathbf{x}, \gamma_m), \tag{2.8}$$

and the process is then repeated for all $m = 1, ..., M$ [11, Sec. 10.2].

Solving Equation 2.7 can be a difficult task in general, and hence, some type of numerical optimization methods are typically used instead [11, Sec. 10.9]. Gradient Descent is one such method, which instead of solving Equation 2.7 directly, works by making iterative updates towards the target in the direction of the negative gradient. In practice, this is done by fitting each basis function to the gradient of the inner function in Equation 2.7 with respect to the current expansion,

$$\frac{\partial \sum_{i=1}^{N} L(y_i, f_m(x_i))}{\partial f_m(x_i)} = \frac{\partial L(y_i, f_m(x_i))}{\partial f_m(x_i)} \text{ [12].} \tag{2.9}$$

For regression tasks, the loss function is usually chosen as as the squared-error loss, defined as

$$L(y_i, f_m(x_i)) = \frac{1}{2}(y_i - f_m(x_i))^2, \tag{2.10}$$

which, when inserted in to Equation 2.9, results in

$$\frac{\partial L(y_i, f_m(x_i))}{\partial f_m(x_i)} = y_i - f_m(x_i), \tag{2.11}$$

which is just the residual as defined in Section 2.2 [12].

In practice, this means that for each iteration of the Gradient Boosting algorithm, the residuals of the current model are computed and a new tree is trained on these

residuals. In addition to the regular decision tree hyperparameters, a learning rate $\nu \in (0, 1]$ is also introduced as a regularization technique that scales the contribution from each tree [11, Sec. 10.12]. Figure 2.7 shows a visual representation of the Gradient Boosting algorithm.

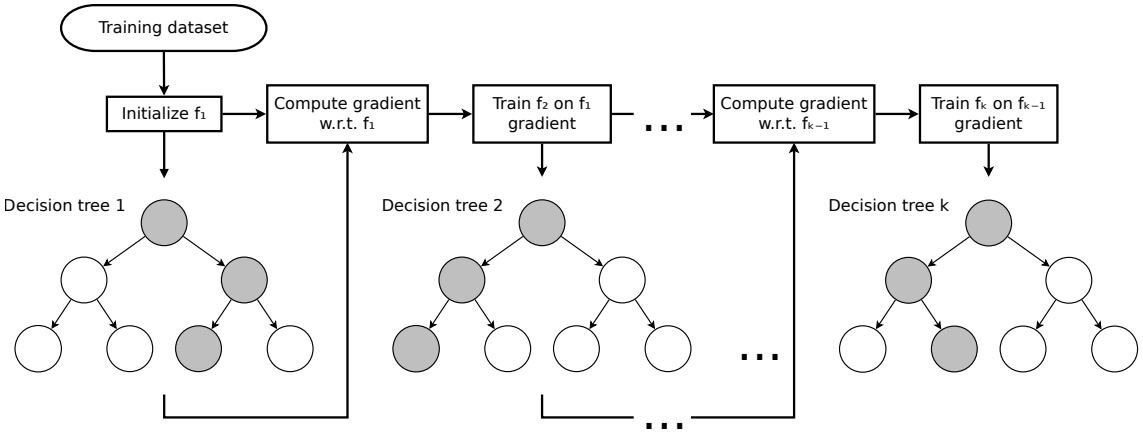


Figure 2.7: Overview of the Gradient Boosting algorithm.

2.3 Explainable AI

Modern AI methods are capable of solving highly complex tasks, and have matured to the point where very little human intervention is required for their design and deployment [13]. In order to achieve this level of performance, AI methods such as machine learning models can have gigantic parametric spaces, consisting of millions, if not billions, of numerical parameters [3, Sec. 2.2,], [13]. This makes it extremely hard, if not impossible, for humans to understand such models in their entirety [3, Sec. 2.2]. These types of machine learning models are called *black-box models*, and are the opposite of *transparent models*, where the mechanism by which the models work can be directly understood [13].

In many real applications, it is not only important to know if a solution proposed by a machine learning model is any good, but also why the specific solution was suggested and what the model's reasoning behind it was [3, Sec. 3.1]. Without model transparency, there is an inherent risk of creating and using decisions that are not justifiable or legitimate [13]. For example, in the case of SOH prediction, it is important to know why a vehicle is given a low predicted SOH. With this information, proper action can be taken to prevent further damage to the battery, and measures could be taken to prevent similar situations for other vehicles in the future.

Explainable Artificial Intelligence (XAI) is a field that aims to provide a set of methods that allow humans to understand the functioning of AI systems [13]. There exists a wide range of terminology that is used differently across the XAI community [13], and so to make thing simple, in this report terms such as *interpretability*, *explainability* and *transparency* will be used interchangeably unless otherwise specified. All of these terms refers to the degree to which a human can understand the cause of a decision made by a machine learning model. A higher interpretability, explainability or transparency means that it is easier for a human to understand the decisions made by a model [3, Ch. 3].

Interpretability is not limited to a single type of metric, and there are different types of interpretability methods that give different kinds of insights. *Feature summary statistics* can give indications as to the importance of, or interactions between, specific features. By looking at the *model internals*, such as their learned weights, their relationship to the model input and output may be examined. Another option to look at *data points*, and how changing feature values affects the model output [3, Sec. 3.2]. This project focuses primarily on feature summary statistics, as the interaction between input data features and output prediction is of primary interest.

There are different ways to achieve model interpretability. *Intrinsic* model interpretability is achieved by limiting the complexity of models, while *post-hoc* interpretability refers to methods that are applied after model training [3, Sec. 3.2], [13]. It is also possible to apply post-hoc interpretability methods to fully or partly intrinsically interpretable models [3, Sec. 3.2]. XAI methods are also categorized depending on the type of machine learning model they can be applied to. *Model-specific* interpretability methods only work for specific types of models, while *model-agnostic*

interpretability methods do not make any assumptions about the underlying model, and hence, can be plugged in for different types of models [3, Sec. 3.2], [13].

2.3.1 Feature importance

As described in Section 2.2, machine learning models are generally trained multiple data features. Not all of these features have to be equally important, however, and in practice they rarely are. To better understand the model, it is often useful to know how important specific features are to the predictions made by a model [11, Sec. 10.13.1].

Feature importance is one such metric, which describes how important the different features are for making predictions relative to each other. Computing feature importance scores for decisions trees is especially easy, as they are already intrinsically interpretable to a great extent [13]. The same greedy approach that is used for growing the trees, can be used to create a quantitative feature importance measure.

As described in Section 2.2.1, each node in a decision tree corresponds to a specific feature that is used to split the input data region into two sub-regions. To compute the importance of a specific feature, the contributions of all nodes in the tree corresponding to that specific feature are summed. The contribution of a single node is determined by the maximum possible error reduction when fitting two separate constant response values to the node's children [11, Sec. 10.13.1].

This definition should sound familiar however, as it is the same as what is computed when splitting the nodes during the initial construction of the tree. Hence, computing the feature importance for an existing tree is only a matter of summing all, already computed, contributions. The concept can be generalized to forests of decision trees, by simply aggregating the feature importance computed for all trees [11, Sec. 10.13.1].

2.3.2 Shapley Values

Lloyd S. Shapley proposed *Shapley Values* in 1951 [14], as a method for assigning payouts to the players of a game, based on their contributions to the game's outcome. The method stems from cooperative game theory, but has found use in other fields as well. In particular, the method can be transferred to the domain of prediction tasks by interpreting the task as a "game", each feature a "player", and the model output a "payout" [3, Sec. 9.6].

To better explain how Shapley Values are computed for prediction tasks, consider the following example inspired by an example given by Molnar in the book *Interpretable Machine Learning* [3].

The task is defined as predicting the value of houses. Three different features are defined: "Area", which is the house's area, "Build year", which is the year that the house was built, and "Garage", which is a Boolean value of whether the house has a garage or not. A model has been trained on the task, and the average predicted

value among all houses in the dataset is €420 000. Given a set of input values, the model can produce an output prediction as shown in Figure 2.8.

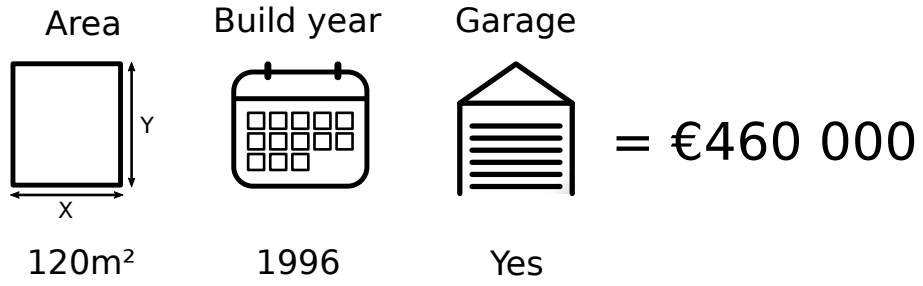


Figure 2.8: Example output of a prediction model given a set of input data.

The difference between the average prediction and the prediction in Figure 2.8 is €460 000 − €420 000 = €40 000. The key idea behind Shapley values is that this difference is the result of a sum of contributions made by the features. For example, the "Area = 120 m^2" feature may contribute with +€20 000, the "Build year = 1996" feature may contribute with −€10 000, and the "Garage = Yes" feature may contribute with +€30 000. In total, these contributions sum to €40 000, the observed deviation from the average prediction.

So how are these contributions computed in a meaningful way? Consider what happens to the prediction if all feature values are fixed, with the exception of one. The feature "Garage" is flipped, such that it is now negative, and a new prediction is computed with the updated set of input values. The resulting prediction is shown in Figure 2.9.

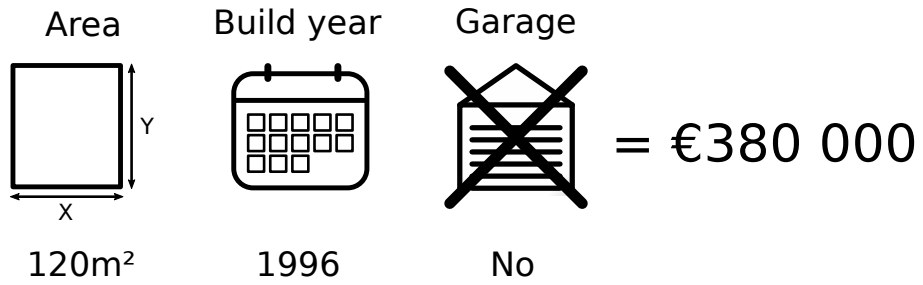


Figure 2.9: Example output of a prediction model given a set of input data with a single feature modified from the example in Figure 2.8.

The difference between the two predictions shown in Figures 2.8 and 2.9 is €380 000 − €460 000 = −€80 000. Since all feature values except one are fixed, this reduction in predicted value can attributed to the change in the value of the "Garage" feature. Formally, we call this a *marginal contribution* made by the "Garage" feature towards the predicted value.

If the values of the "Area" and "Build year" features were different, the marginal contribution of the "Garage" feature would likely not be the same. For this reason, many data samples with different feature values, so called *feature coalitions*, has

to be considered to get a statistically significant result. The Shapley value of a specific feature is formally defined as the average of all marginal contributions for this feature, across all possible feature coalitions [3, Sec. 9.5.1].

Computing exact Shapley values is an intractable problem, since there is an exponential number of coalitions that all have to be considered [3, Sec. 9.5.5]. For this reason, some type of approximation algorithm is typically used instead.

2.3.3 SHAP

Shapley Additive Explanations (SHAP) is a popular method for explaining model predictions introduced by Lundberg and Lee in 2017 [15]. The method is based on computing the Shapley values of a conditional expectation function of the original model. SHAP is an additive feature attribution method, where local explanations are formulated as a sum of SHAP values on the form

$$g(z) = \phi_0 + \sum_{i=1}^{M} \phi_i z_i, \tag{2.12}$$

where ϕ_0 is the attribution of the base prediction, ϕ_i is the attribution corresponding to each feature coalition, $z_j \in \{0,1\}^M$ is a binary variable for including or discluding a specific feature, and M is the number of features.

SHAP explanations tell how the base (average) prediction over the test set $E[f(z)]$ is pushed towards the actual model prediction $f(x)$, by adding attribution factors ϕ_i that are computed by conditioning on features $z_i = x_i$ for $i = 1, ..., M$ [15]. Figure 2.10 shows a simplified case of how SHAP values work, inspired by an example from the original SHAP paper [15].

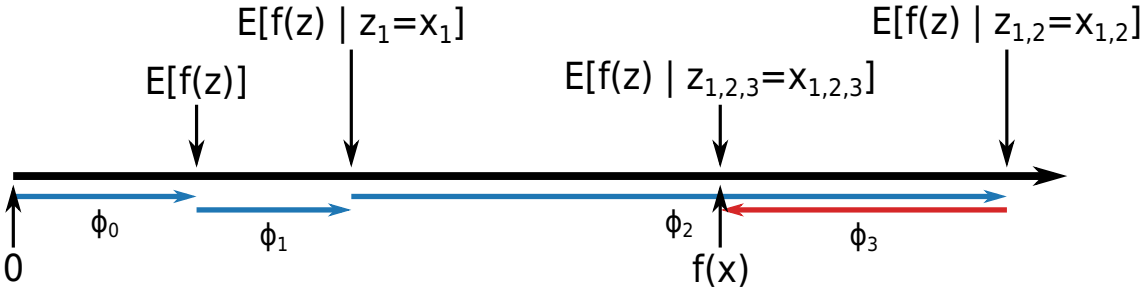


Figure 2.10: A simplified example of how SHAP values are used to explain a prediction. In the general case, the order in which variables are added matters, and hence, the average of all possible orderings is considered instead.

As can be seen in Figure 2.10, the expectations has to be computed using feature coalitions that may not contain all features. Since models can generally not deal with such arbitrarily missing input features, SHAP defines a simplified input mapping that resolves this issue by approximation [15]. An example of how this approximation is computed is shown in an upcoming example.

SHAP values are difficult to compute in much the same way as Shapley values, and for this reason, approximation methods are typically used [15]. SHAP values

are generally model-agnostic. However, using certain model-specific approximation methods can speed up computations considerably.

TreeSHAP is a model-specific estimation method for SHAP values introduced by Lundberg et al. in 2020 [16], that is specifically intended for tree-based machine learning models, such as those introduced in Section 2.2. The method brings the complexity of computing exact Shapley values down to polynomial time [16].

In the original TreeSHAP paper, Lundberg et al. proposed two variants of the algorithm, *Path-dependent TreeSHAP* and *Interventional TreeSHAP*. The two algorithms work in much the same way and share the same underlying algorithm, but use different approaches to approximating the contributions made by features that are not present in any given coalition.

To explain how TreeSHAP works, start off by considering how Path-dependent TreeSHAP can be used to explain the prediction of the decision tree shown in Figure 2.4. In addition to the tree itself and the input data, Path-dependent TreeSHAP also requires knowledge of the distribution of the number of data points over the tree's nodes. In this example, there are a total of 15 samples, and the distributions of these are shown in Figure 2.11 as the variable n in each node.

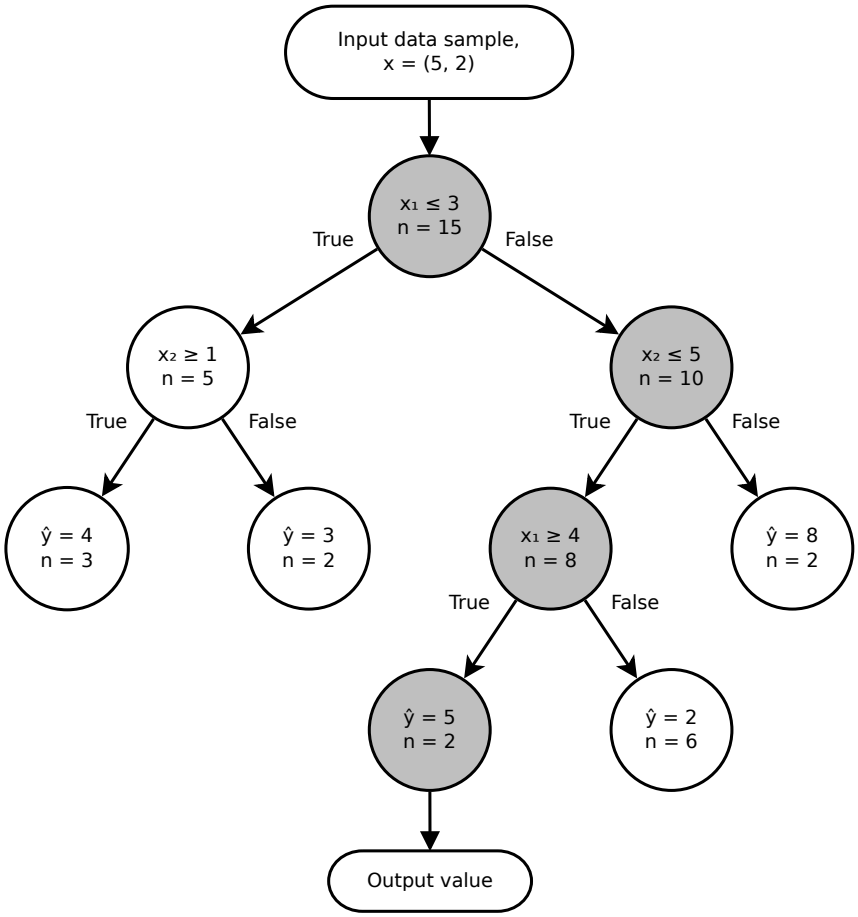


Figure 2.11: Example of a prediction tree, with the number of data samples corresponding to each node marked.

To start of with, the base attribution factor ϕ_0 is computed for the tree. This is done by evaluating the expectation $E[f(z)]$, and since it is not conditioned on any of the variables, this is approximated by computing the weighted sum over all leaf nodes with respect to the number of data points,

$$\phi_0 = E[f(z)] = \frac{3}{15} \cdot 4 + \frac{2}{15} \cdot 3 + \frac{2}{15} \cdot 5 + \frac{6}{15} \cdot 2 + \frac{2}{15} \cdot 8 = \frac{56}{15}. \tag{2.13}$$

Next, the marginal contributions of all possible coalitions is computed. Since the order in which variables are added to the feature coalition set matters, all possible sequences have to be considered. In this example, there are two (by adding either x_1 or x_2 first), and the final SHAP values will be an average over all sequences.

Starting with the first sequence, x_1 is added to the empty feature coalition set, and the conditional expectation $E[f(z) \,|\, z_1 = x_1]$ is evaluated. This is done by following the prediction path and evaluating the conditions for each node along the way.

Starting at the root node, the condition $x_1 \leq 3$ is evaluated. Since x_1 is in the coalition set, this condition can be directly evaluated to "False", and hence, the next node will be to the right of the root node.

In the second node, the condition $x_2 \leq 5$ is evaluated. However, since x_2 is not in the coalition set, the condition cannot be directly evaluated. Instead, the conditional expectation is approximated by the weighted sum of over all possible descendant leaf nodes.

If the condition evaluates to "False", the right node is chosen, and since this is a leaf node, there are no more descendants and its weighted contribution is be computed as $\frac{2}{10} \cdot 8$. If the condition evaluates to "True", the left node is chosen. Its condition, $x_1 \geq 4$, contains x_1, which is in the coalition set. Hence, it can be directly evaluated to "True". The next and final node is a leaf node, and its weighted contribution is be computed as $\frac{8}{10} \cdot 5$.

By summing up the weighted contributions, an approximation of the conditional expectation is obtained. Hence, the marginal contribution of x_1 in this sequence is

$$\phi^1_{x_1} = E[f(z) \,|\, z_1 = x_1] - E[f(z)] = \left[\frac{8}{10} \cdot 5 + \frac{2}{10} \cdot 8\right] - \frac{56}{15} = \frac{28}{15}. \tag{2.14}$$

Next, x_2 is added to the coalition and $E[f(z) \,|\, z_{1,2} = x_{1,2}]$ is evaluated. Since all variables are now known, this simply results in the actual predicted value, and so the marginal contribution of x_2 in this sequence is

$$\phi^1_{x_2} = E[f(z) \,|\, z_{1,2} = x_{1,2}] - E[f(z) \,|\, z_1 = x_1] = 5 - \left[\frac{8}{10} \cdot 5 + \frac{2}{10} \cdot 8\right] = -\frac{3}{5}. \tag{2.15}$$

For the second sequence, x_2 is added to the empty coalition set and $E[f(z) \,|\, z_1 = x_2]$ is evaluated. Since x_1 is missing from some of the nodes, the same method of approximation has to be applied as before. The only difference is that it will now

have to be applied twice, since there are two nodes along the path that contain x_1. The resulting marginal contribution of x_2 is

$$\phi_{x_2}^2 = E[f(z) \mid z_1 = x_2] - E[f(z)] = \\ = \left[\frac{5}{15} \cdot 4 + \frac{10}{15} \cdot \left(\frac{2}{8} \cdot 5 + \frac{6}{8} \cdot 2\right)\right] - \frac{56}{15} = -\frac{17}{30}. \quad (2.16)$$

Next, x_1 is added to the coalition and $E[f(z) \mid z_{1,2} = x_{2,1}]$ is evaluated. Once again, since all variables are now known, the result is simply the actual prediction value, and the marginal contribution becomes

$$\phi_{x_1}^2 = E[f(z) \mid z_{1,2} = x_{2,1}] - E[f(z) \mid z_1 = x_2] = \\ = 5 - \left[\frac{5}{15} \cdot 4 + \frac{10}{15} \cdot \left(\frac{2}{8} \cdot 5 + \frac{6}{8} \cdot 2\right)\right] = \frac{11}{6}. \quad (2.17)$$

With all the factors sequences accounted for, the total contribution of each variable is averaged as

$$\phi_{x_1} = \frac{\phi_{x_1}^1 + \phi_{x_1}^2}{2} = \frac{\frac{28}{15} + \frac{11}{6}}{2} = \frac{37}{20}, \quad (2.18)$$

$$\phi_{x_2} = \frac{\phi_{x_2}^1 + \phi_{x_2}^2}{2} = \frac{-\frac{3}{5} - \frac{17}{30}}{2} = -\frac{7}{12}. \quad (2.19)$$

The total explanation is given by summing all contributions as

$$\phi_0 + \phi_{x_1} + \phi_{x_2} = \frac{56}{15} + \frac{37}{20} - \frac{7}{12} = 5, \quad (2.20)$$

which indeed is the predicted output value.

Path-dependent TreeSHAP gets its name from how missing features are approximated with an average, weighted by the relative number of samples flowing down to each respective path. Interventional TreeSHAP works in a similar way, but instead of approximating missing feature contributions by a weighted average, it samples replacement values from a set of background data [16]. There are up- and downsides to both approaches, but since there is a large amount of data available to be used as background data in this project (specifically the training dataset), all subsequent references to TreeSHAP concern Interventional TreeSHAP.

It may be noted that the algorithm that was used above is actually in exponential time. However, even in this small example, multiple terms were computed more than once. The actual implementation of TreeSHAP follows the same general idea, but makes use of a cache to avoid recomputing terms when possible. This brings the overall complexity of the algorithm down to polynomial time [16]. Furthermore, since SHAP values are additive, the method can be easily translated to forests by simply computing the SHAP values for each tree and then averaging them across all trees [16].

2.4 DenseWeight

As described in Section 2.2, the purpose of machine learning algorithms for regression tasks is to produce some function $f : \mathbb{R}^n \to \mathbb{R}$ fitting a training dataset. If the training dataset is imbalanced, and contains an overrepresentation of a certain type of data, this will also be reflected in the trained model. This can be problematic, since rarer data points may actually be more important for fully understanding the task. In SOH prediction tasks, it is especially important to accurately predict outliers, since these deviate from what is to be expected and may suggest a need for intervention.

One solution to this problem is to weight the dataset, such that more common data points are given less consideration by the model during training, compared to data points that are more unique. This is commonly done in classification tasks, where class weights can be computed by simply considering the relative number of samples in each class [17].

Such an approach does not work for regression tasks, however, owning to the continuous output space. Additionally, such approaches only consider the target value, and do not consider the commonality of the feature values. Hence, if a data point has very unique feature values, but a common target value, it will still be weighted as any other common data point.

M. Steininger et al. proposed *DenseWeight* in 2020 [17], as an algorithm for computing sample weights from continuous features. This approach solves the issues mentioned above, and gives more fine-grain control since the weights are computed on a per-sample basis. The algorithm is centered around the use of Kernel Density Estimation (KDE) to estimate the Probability Density Function (PDF) of the dataset. The PDF is then used to construct a weighting function which can be used during model training.

Given a set of target values Y, the PDF is estimated with KDE as

$$f_p(y) = \frac{1}{|Y|h} \sum_{y_i \in Y} K\left(\frac{y - y_i}{h}\right), \tag{2.21}$$

where $|Y|$ is the number of data points in Y, K is a kernel function and h is the bandwidth, both selected as hyperparameters. The PDF is then normalized to $[0, 1]$ by

$$f'_p(y) = \frac{f_p(y) - \min[f_p(Y)]}{\max[f_p(Y)] - \min[f_p(Y)]}, \tag{2.22}$$

where $f_p(Y)$ is the element-wise application of f_p to Y.

Two parameters are then introduced: $\alpha \in \mathbb{R}^+$ is a scaling factor, and $\epsilon \in \mathbb{R}^+$ is a lower weight bound that prevents weights less than or equal to zero. The weighting function is computed as

$$f'_w(\alpha, \epsilon, y) = \max[1 - \alpha f'_p(y), \epsilon]. \tag{2.23}$$

Finally, the weighting function is normalized to have a mean of 1. Hence, the final DenseWeight weighting function is

$$f_w(\alpha, \epsilon, y) = \frac{f'_w(\alpha, \epsilon, y)}{\frac{1}{|Y|}\sum_{y_i \in Y} f'_w(\alpha, \epsilon, y_i)} = \frac{\max[1 - \alpha f'_p(y), \epsilon]}{\frac{1}{|Y|}\sum_{y_i \in Y} \max[1 - \alpha f'_p(y), \epsilon]} \text{ [17].} \tag{2.24}$$

As for the parameters, ϵ can typically be chosen as an arbitrarily low value (such as $\epsilon = 10^{-6}$), while α is usually fine-tuned as a hyperparameter since it has a considerable impact on the weights. Note that α can be greater than 1, and when $\alpha > 1$ more data points are "pushed" into ϵ.

A number of common kernel functions K exists, but in this project the *Epanechnikov kernel* [18] is primarily considered. This is mainly due to the kernel having a finite width, which considerably speeds up computations when compared to an infinite width kernel such as the Gaussian kernel.

When selecting the bandwidth h

2.5 Pearson correlation

The *Pearson correlation coefficient* is a measurement of the linear correlation between two variables. The Pearson correlation is defined as

$$\rho(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}, \tag{2.25}$$

where $\text{cov}(X, Y)$ is the covariance between variables X and Y, and σ_X and σ_Y are the standard deviations for variables X and Y respectively [19, Sec. 11.6].

3 | Datasets

In this chapter, the two datasets that were used in the project are presented: the *Vehicle Fleet Dataset*, consisting of histogram data measured from a large fleet of Volvo PHEVs, and the *Randomized Battery Usage Dataset*, consisting of time series data measured from lithium-ion battery cells cycled under controlled conditions. Table 3.1 shows a summary of the key differences between the two raw datasets.

Table 3.1: Summary of differences between the two raw datasets.

Dataset	Vehicle Fleet Dataset	Randomized Battery Usage Dataset
Measurement entity	Car (Battery pack)	Battery cell
Number of entities	$\sim 65\,000$	28
Number of samples	$\sim 200\,000$	77 700 820
Avg. measurement duration (days)	~ 487	~ 149

The Vehicle Fleet Dataset and the Randomized Battery Usage Dataset are described in more depth in Sections 3.1 and 3.2 respectively.

3.1 Vehicle Fleet Dataset

The *Vehicle Fleet Dataset* consists of measurements from a customer fleet of Volvo PHEVs built on the Volvo Scalable Product Architecture (SPA) platform. This fleet was chosen because of its suitable population size, and suitable data collection time frame for capturing relevant battery degradation characteristics.

The data was collected using digital readouts during workshop visits. The data was collected over a period of four years, and as the service interval for passenger cars is typically recommended to one year, this method of data collection results in few measurements for each vehicle. This is illustrated in Figure 3.1, which shows a cropped histogram for the relative frequency of the number of measurements per unique vehicle. The maximum number of measurements for any vehicle in the dataset was 19.

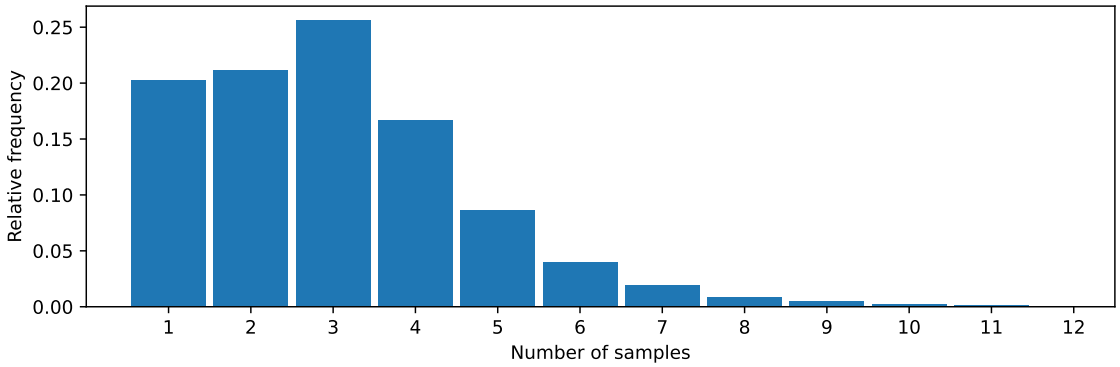


Figure 3.1: Histogram for the relative frequency of the number of measurements per vehicle.

New vehicles were added to the dataset as they were delivered throughout the measurement period. One readout is made as part of the finalizing steps before the vehicles are delivered to customers, which causes an overrepresentation of measurements for new vehicles in the dataset. A consequence of this is illustrated in Figure 3.2, which shows a histogram of odometer readouts where there is a clear spike around 0 km.

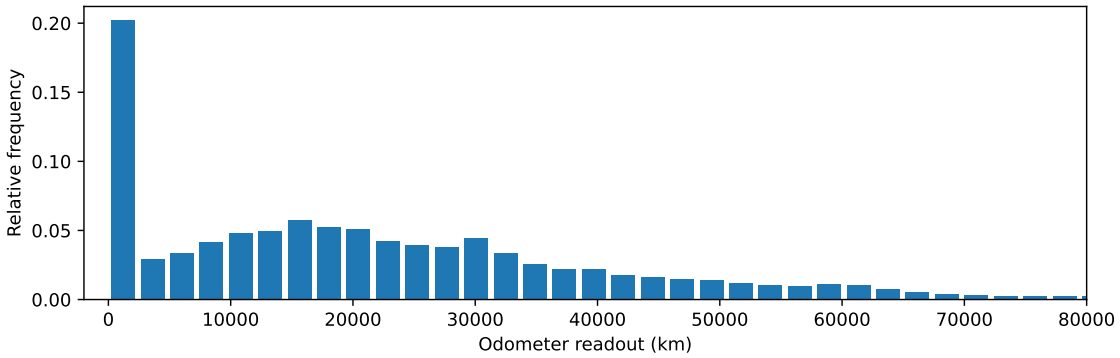


Figure 3.2: Histogram for the relative frequency of odometer readouts.

The dataset consists of a refined set of signals, that have been determined to be relevant to battery ageing during previous work at Volvo Cars. Some of these signals are in a histogram format, as described in Section 1.1. The dataset has previously been cleaned to remove measurements with high uncertainty, or that contain invalid values. An overview of the signals in the resulting dataset is shown in Table 3.2.

Table 3.2: Overview of the signals used in the Vehicle Fleet Dataset.

Type	Format	Signal
Identifier	Simple	Anonymous unique vehicle identifier
Time	Timestamp	Readout time
Target	Simple	SOH estimate
Feature	Simple	Global time
	Simple	Odometer value
	Histogram	Accumulated charge throughput at different total times
	Histogram	Accumulated running time in RMS current regions
	Histogram	Accumulated parked time in SOC & Temperature regions

Out of the dataset's five signals, two are represented as simple numerical values and three are represented in the histogram format. These three are the

- *Accumulated charge throughput at different total times*, which accumulates the charge throughput at pre-defined total times.
- *Accumulated running time in RMS current regions*, which accumulates the running time spent at specific battery RMS current regions. "Running time" refers to the time that the car is running.
- *Accumulated parked time in SOC & Temperature regions*, which accumulates the parked time spent in specific SOC and temperature regions. "Parked time" refers to the time that the car is not running.

Example readouts of the signals are shown in Figures 3.3 to 3.5 respectively.

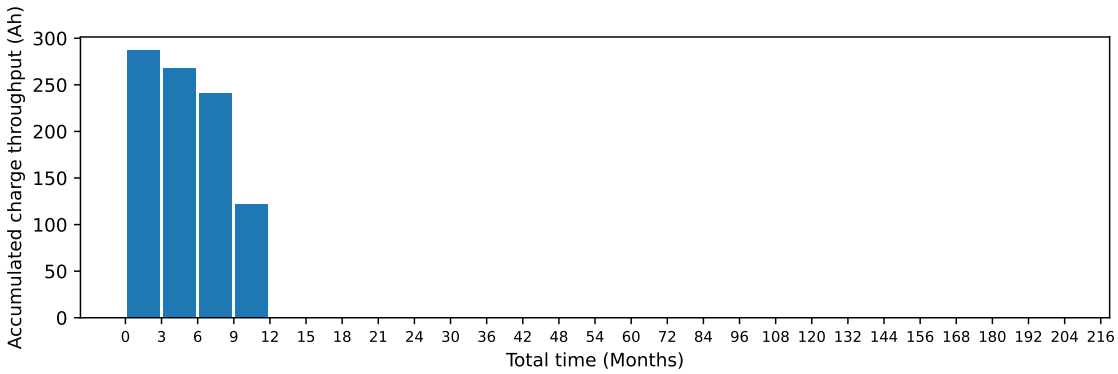


Figure 3.3: Example readout of the *Accumulated charge throughput at different total times* feature.

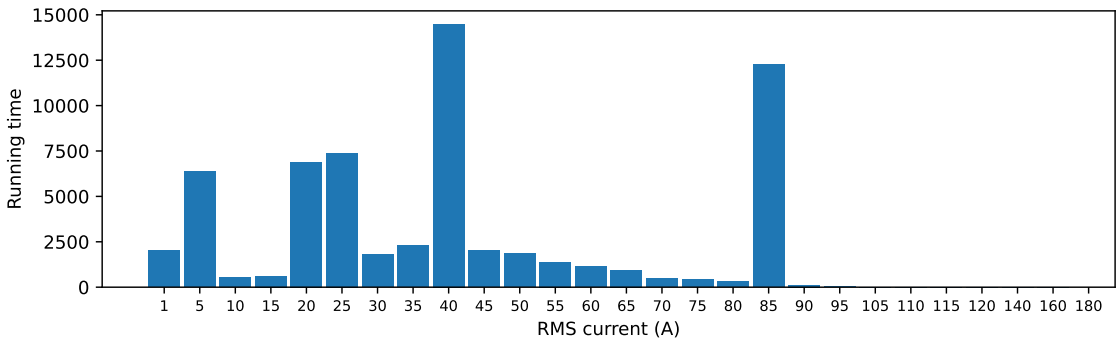


Figure 3.4: Example readout of the *Accumulated running time in RMS current regions* feature.

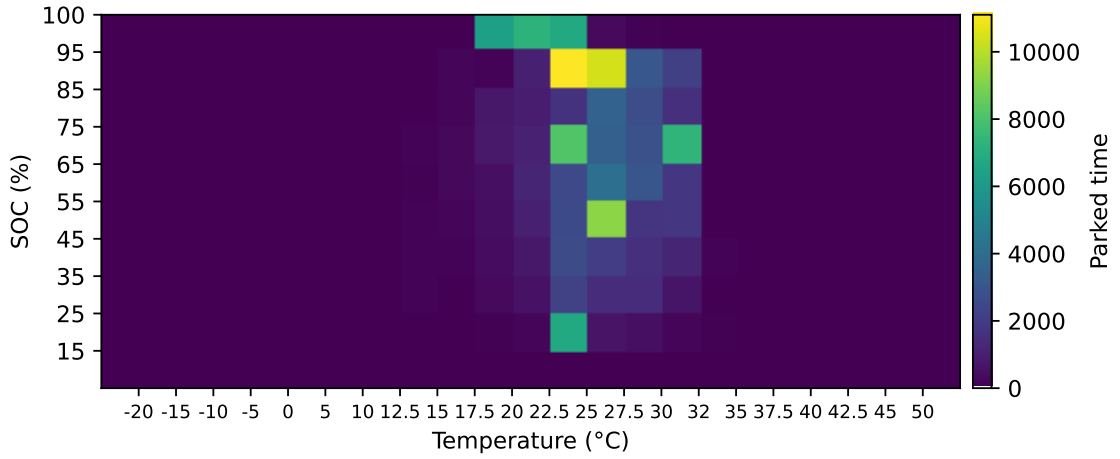


Figure 3.5: Example readout of the *Accumulated parked time in SOC & Temperature regions* feature.

The Vehicle Fleet Dataset has the benefit of reflecting true operating conditions, but it it also has a number of downsides. One major caveat is the highly proprietary data collection method, which affects both data pre-processing and modelling aspects. It is not known what downstream consequences this may have, and since data has to be collected over longer time periods, it is not possible to easily collect new data in short time using different methods.

Furthermore, the resulting proprietary data format makes it difficult to compare any results obtained on the dataset to those obtained on other battery ageing datasets, and as a result, to most previous work in the field.

3.2 Randomized Battery Usage Dataset

In order to examine the effects of the data collection method, a secondary dataset consisting of raw battery measurements was required. There exists a plethora of publicly available datasets fit for battery SOH prediction, but many of these lack reference cycles from which SOC and SOH can be directly and accurately estimated.

The *Randomized Battery Usage Data Set* [20], created by the NASA Ames Research Center, contains data of 28 battery cells cycled under various temperature conditions and randomly selected current profiles. The cells are split into groups of four, each group being tested under unique conditions. In addition to the randomized charge and discharge cycles, the dataset contains reference cycles from which SOC and SOH can be estimated. The dataset is given in a time series format, and an overview of the signals in the resulting dataset is shown in Table 3.3.

Table 3.3: Overview of the signals in the Randomized Battery Usage Dataset.

Type	Signal	Description
Meta	Comment	Information about the current cycle
	Cycle type	Type of cycle (charge, discharge or rest)
	Date	Global timestamp at start of current cycle
Time	Time	Time since the start of the current experiment
	Relative time	Time since the start of the current cycle
Feature	Voltage	Voltage readout in V
	Current	Current readout in A
	Temperature	Temperature readout in °C

The three signals, *Voltage*, *Current* and *Temperature*, are represented by a simple numerical format, where each readout corresponds to a single measurement. Examples of the feature signals are seen in Figures 3.6 to 3.8. Each figure shows a linear interpolation of the first 50 000 readouts for a single battery cell, plotted against the *Time* signal. Note that some of the feature values are slightly misleading as a result of the interpolation.

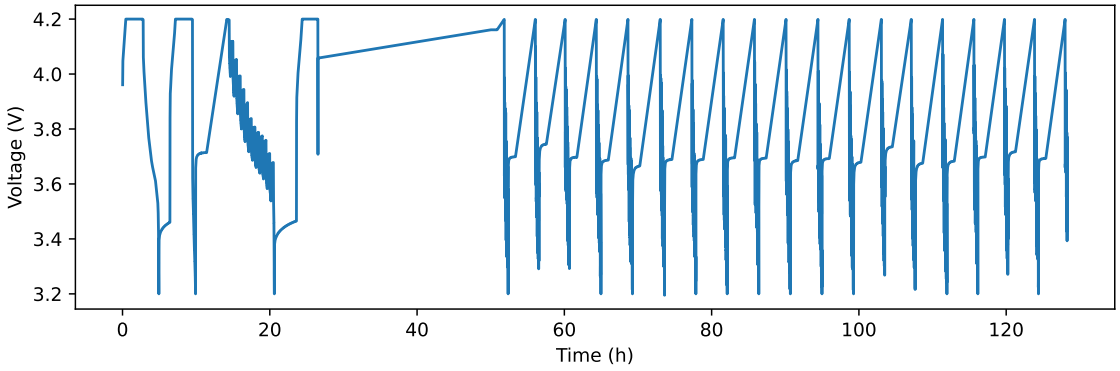


Figure 3.6: Example of the *Voltage* feature signal.

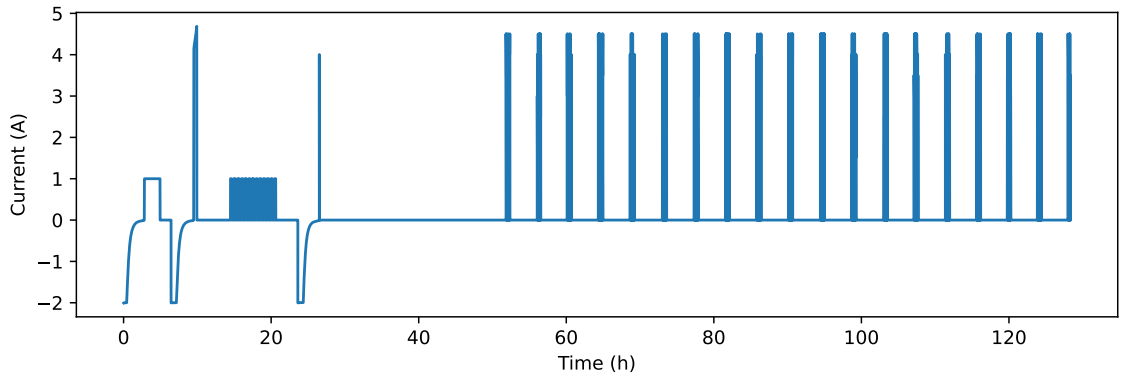


Figure 3.7: Example of the *Current* feature signal.

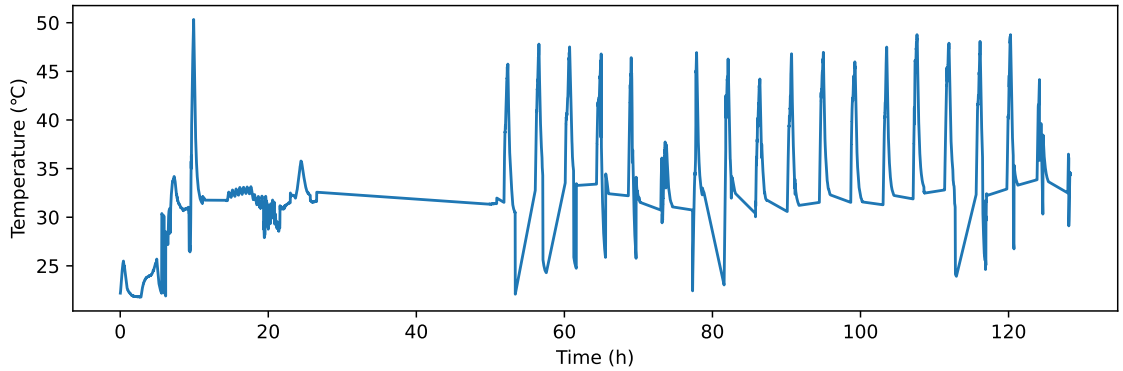


Figure 3.8: Example of the *Temperature* feature signal.

4 | Methodology

In this chapter, the methodology that was used for the project is presented. First, the method of data pre-processing is presented in Section 4.1. In Section 4.2, the modelling approach is described. In Section 4.4, the individual experiments that were conducted are presented.

4.1 Data pre-processing

In the following sections, the data pre-processing that was done for each of the two datasets is presented. In Section 4.1.3, a general pre-processing and feature engineering pipeline that was used to process histogram data into a simpler representation is presented. In Sections 4.1.1 and 4.1.2, the pre-processing steps that were specific to the Vehicle Fleet Dataset and the Randomized Battery Usage Dataset respectively, are presented.

4.1.1 Vehicle Fleet Dataset

The Vehicle Fleet Dataset was split into a training, a validation and two test sets by a 70-10-10-10 split. Splitting was done randomly on the unique vehicle identifier, such that all measurements from the same vehicle were placed in the same subset, to prevent data leakage between the subsets. Two test sets were created originally for the sake of redundancy, but the second test set was never used. For this reason, all references to the "test set" throughout the rest of this report refers to only the first of the two sets.

4.1.2 Randomized Battery Usage Dataset

For the Randomized Battery Usage Dataset, some initial data cleaning was required since some of the measurements for battery cells 3, 2 and 18 contained clearly invalid temperature values (reaching $-4000\,^{\circ}\mathrm{C}$). This was done by removing the data samples containing invalid measurements, resulting in all measurements for cell 2, 73% measurements of cell 3, and 17% measurements of cell 18 being removed. Not all cell 2 measurements were in fact invalid, but the remaining valid measurements were too few to be of use.

As described in Section 3.2, the Randomized Battery Usage Dataset only contains raw signals that were directly measured from the battery cells. To make the Ran-

domized Battery Usage Dataset comparable to the Vehicle Fleet Dataset, features such as the SOC and SOH had to be computed. Additionally, the time series data had to be accumulated into histograms to make the data formats analogous. For this reason, an additional data pre-processing and feature engineering pipeline was used for the Randomized Battery Usage Dataset. An overview of this pipeline is shown in Table 4.1.

Table 4.1: Description of the Randomized Battery Usage Dataset pre-processing pipeline.

Step	Description
1	Basic pre-processing and data formatting is performed
2	The absolute current is computed from the current
3	The accumulated charge throughput is computed by integrating the absolute current
4	The SOH is computed by Coulomb counting during reference cycles
5	Missing SOH values are interpolated
5	The SOC is computed by Coulomb counting during reference cycles
6	Missing SOC values are interpolated
7	Histograms are computed from the time series data at pre-fined steps
8	Non-histogram data points are dropped
9	Original signals are dropped

The SOH was estimated by coulomb counting. This was done by integrating the discharge current (the *Current* feature) with respect to time during the reference discharge cycles, labeled "reference discharge" in the dataset, to compute the total charge capacity. The SOH could then be computed using Equation 2.2, by using the maximum charge capacity estimate as the maximum nominal capacity. The SOH for the remaining cycles was then computed by linear interpolation.

The SOC was computed in two steps. First, the voltage-SOC relationship was determined by estimating an Open-Circuit Voltage (OCV)-SOC curve. For this step, the charge capacity was estimated by cumulatively integrating the discharge current (the *Current* signal) during the reference discharge cycles. The SOC could then be computed using Equation 2.1, by using the total charge capacity as the nominal capacity. The second step was to use the OCV-SOC curves to estimate the SOC from the *Voltage* signal for the remaining data points, that are not in any reference cycle. This was done by first clipping the *Voltage* signal to the same range as in the reference cycle, and then using linear interpolation to estimate the SOC. As the OCV-SOC relationship typically changes with battery age, the latest reference cycle was always used when interpolating the SOC.

After computing the features, the data was accumulated into histograms by binning the feature values from the beginning-of-life up until a set of target cycles. The histogram bins were manually selected to encompass the full range of possible values, whilst resulting in a similar number of bins compared to the Vehicle Fleet Dataset. The selected bins are shown in Table 4.2. Finally, the original signals were

dropped alongside all data points in between the data points at which histograms were computed.

Table 4.2: The histogram bins selected for the Randomized Battery Usage Dataset.

Corresponding feature	Unit	Bins
Total time vs. Accumulated charge throughput	Time (weeks)	1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49, 52, 55, 58, 61, 64, 67, 70, 73, 76, 79, 82
RMS current vs. Running time	Current (A)	0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0, 3.25, 3.5, 3.75, 4.0, 4.25, 4.5, 4.75, 5.0, 5.25, 5.5, 5.75, 6.0, 6.25, 6.5, 6.75
SOC & Temperature vs. Parked time	SOC (%)	0, 15, 25, 35, 45, 55, 65, 75, 85, 95, 100
	Temperature (°C)	10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62

The Randomized Battery Usage Dataset was split by randomly picking two cells from each group to the training set, and one each to the validation and test sets. This resulted in a roughly 50-25-25 split.

4.1.3 General data pre-processing pipeline

As described in Section 2.2, machine learning models typically assumes that the input data is on form $\mathbf{x} \in \mathbb{R}^n$. Histogram data such as the Vehicle Fleet Dataset, or the processed Randomized Battery Usage Dataset, are not on this form. For this reason, the data has to be converted into a simpler representation prior to usage in machine learning applications. To accomplish this, a general data pipeline was created that could be used on both datasets after initial pre-processing. This pipeline is outlined in Table 4.3.

Table 4.3: Description of the general data pre-processing pipeline.

Step	Description
1	The multi-dimensional histograms are marginalized with respect to each dimension, to construct a greater number of one-dimensional histograms.
2	The (one-dimensional) histograms are aggregated into a set of simple numerical features.
3	The histograms are dropped from the dataset.
4	All samples containing invalid values are dropped from the dataset.
5	The vehicle age is estimated as a cumulative sum of time between samples.
6	Constant features are dropped from the dataset

As a result of the first pipeline step, the two-dimensional *SOC & Temperature vs. Parked time* histogram is transformed into two one-dimensional histograms through marginalization. These are referred to as *SOC & Marginalized Temperature vs. Parked time* and *Marginalized SOC & Temperature vs. Parked time*.

In the second step, the one-dimensional histograms are transformed into simple numerical features. To accomplish this, a number of aggregations were performed on the distributions, that aimed to capture the distribution characteristics whilst resulting in considerably fewer features. As part of previous work done at Volvo Cars, the following aggregations had been selected:

- Minimum histogram count
- Maximum histogram count
- Median histogram count
- Sum of histogram counts
- Estimated histogram mean value
- 10th, 25th, 50th, 75th and 90th percentiles

The resulting feature names are given an affix to indicate the aggregation function. A complete list of the generated features can be found in Appendix A.

In the fifth step, an additional feature describing the vehicle age is estimated by cumulatively summing the time differences between measurements. This feature was given the name *Estimated age*. This was done as it had previously been found the *Global timestamp* feature was occasionally unreliable.

Finally, in the sixth step, features for which all data samples had the same value were removed, as they did not contribute any meaningful information. For the Vehicle Fleet Dataset, this resulted in the following constant features being removed:

- Minimum, Total time vs. Accumulated charge throughput
- Median, Total time vs. Accumulated charge throughput
- Minimum, RMS current vs. Running time
- Minimum, Marginalized SOC & Temperature vs. Parked time

For the Randomized Battery Usage Dataset, this resulted in the following constant features being removed:

- Minimum, Total time vs. Accumulated charge throughput
- Median, Total time vs. Accumulated charge throughput
- Minimum, RMS current vs. Running time
- Minimum, Marginalized SOC & Temperature vs. Parked time
- 10th ptl., Total time vs. Accumulated charge throughput
- 10th ptl., RMS current vs. Running time

- 10th ptl., SOC & Marginalized Temperature vs. Parked time

4.2 Modelling

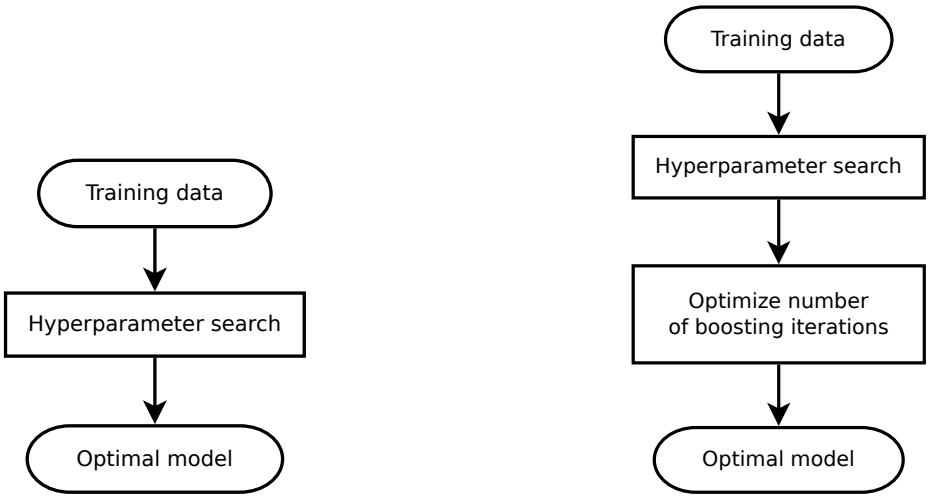
As described in Section 1.2, a number of different statistical methods have previously been used for SOH prediction. Many of these make assumptions on the chronological ordering and relatively high frequency of data samples. This can not be guaranteed for real vehicle data, and hence, these methods are not fully applicable. For this project, the problem was modelled as a simple regression task. Two different regression models were selected: a *Random Forest* algorithm using bagging, and a specific implementation of Gradient Boosting, called *Extreme Gradient Boosting (XGBoost)*.

4.2.1 Hyperparameter selection

As both random forest and gradient boosting models contain hyperparameters that need to be fixed prior to training, a method of hyperparameter selection was required. This was done in a one-step process only utilizing the training dataset for the random forest models, and a two-step process utilizing both the training and validation datasets for the gradient boosting models. Flowcharts describing the hyperparameter selection processes for each of the models are shown in Figure 4.1.

The first step, which was largely the same for both types of models, was to perform an exhaustive search over all possible combinations of hyperparameter values, from a pre-defined search space. This was done using five-fold cross validation on the training set. The models with the lowest average RMSE score across all folds were then selected. For the random forest model, this method was used to select all hyperparameter values. However, as gradient boosting models are trained iteratively, it is possible to precisely determine the optimal number of boosting iterations. For this reason, the number of boosting iterations was fixed in the first step of hyperparameter selection.

For the second step of hyperparameter selection, that was applied to the previously selected gradient boosting model only, the maximum number of boosting iterations was set to an arbitrarily high number. An early stoppage criteria was then used to halt training when the model stopped improving in performance. This was done by checking if the RMSE score computed over the validation set decreased over a fixed number of boosting iterations in sequence. If the validation score did not improve at least once over this period, training was stopped. The optimal number of boosting iterations was then selected as the number iterations prior to stoppage.



(a) Random Forest **(b)** XGBoost

Figure 4.1: Flowcharts of the hyperparameter selection processes.

4.3 Model analysis

Two different explainability methods were selected for analyzing the models: a model-specific feature importance score, and SHAP, a model-agnostic method.

The feature importance scores were computed when creating the underlying decision trees, as described in Section 2.3.1. As the random forest and gradient boosting models were implemented using different packages, the exact methods and criteria that were used to compute the feature importance metrics may differ slightly. The feature importance scores were normalized to have the same scale, and it is believed that any other effects from differences between the implementations are minor. See Appendix B for additional implementation details.

SHAP values for the test set were computed using Interventional TreeSHAP as described in Section 2.3.3, using the training dataset as background data. SHAP values are local explanations, that is, they are computed for individual data points, and hence cannot be directly compared to metrics such as feature importance scores, that are global explanations. To do this, the SHAP explanations first have to be aggregated over all data points. For this purpose, the mean magnitude of the SHAP values was computed for each feature.

4.4 Experiments

A number of different experiments were conducted as part of the study. These are explained in their respective sections below.

4.4.1 Correlation study of the datasets

To get a better understanding of the interactions between the target and feature variables, a basic correlation study was first performed on both datasets. This was done by computing the Pearson correlation coefficients for all variable pairs on both datasets, using Equation 2.25.

4.4.2 Model explanations for the Vehicle Fleet Dataset

When combined, the data pre-processing, modelling and model analysis methods, described in Sections 4.1 to 4.3 respectively, forms a process for creating and analyzing SOH prediction models. This *base process* is summarized in Figure 4.2.

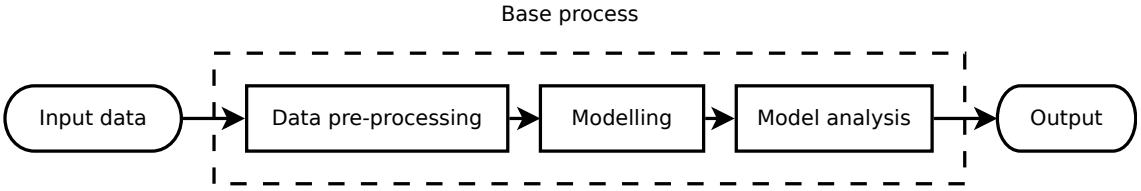


Figure 4.2: Overview of the base process.

To establish a set of baseline results for the process described above, an experiment was conducted using the base process without any additional modifications or additions. Data pre-processing was done as described in Section 4.1, followed by the modelling approach described in Section 4.2. Hyperparameter selection was conducted on the manually selected hyperparameter search space presented in Table 4.4. The models were then evaluated on the test set and analyzed using the methods described in Section 4.3.

Table 4.4: The hyperparameter search spaces for the models created using the base process on the Vehicle Fleet Dataset without modifications.

Model	Parameter	Possible values
Random forest	Num. trees	128, 160, 192
	Max. depth	3, 6, 9, 12
	Min. num. samples to split	2, 4, 6
	Min. num. samples in leaves	1, 2, 4
XGBoost	Max. depth	3, 6, 9, 12
	Learning rate	0.01, 0.05, 0.1, 0.3
	Min. loss reduction to split	0.0, 0.1, 0.3
	Min. sum of child weights	1, 3, 5, 7

4.4.3 Impact of sample weighting on model explanations for the Vehicle Fleet Dataset

In this experiment, the impact that the dataset imbalance described in Section 3.2 had on the results from the first experiment was examined. This was done by

retraining the models using sample weights that were computed with DenseWeight for different α-values.

First, sample weights were computed for the training set using the DenseWeight algorithm, described in Section 2.4. Weights were computed for $\alpha = 1.0, 1.2, 1.4, 1.6$. The remaining DenseWeight parameters were manually selected. The lower weight bound ϵ, was arbitrarily chosen to a low value of $\epsilon = 10^{-6}$. The bandwidth, h, was chosen by starting at an arbitrarily low value and incrementally increasing the bandwidth until the distributions computed using DenseWeight stabilized. A bandwidth above $\epsilon = 4 \cdot 10^8$ resulted in impractically long computational times, and hence, this was selected as the final bandwidth value.

The models were then retrained using the sample weights. For the random forest model, a feature with a higher weight is given an increased importance when performing node splitting. For the gradient boosting model, the weights are used to directly weight the gradients. After training, the models were evaluated and analyzed in the same way as in the first experiment.

4.4.4 Impact of sampling frequency on model explanations for the Randomized Battery Usage Dataset

In this experiment, the impact of the data collection frequency was investigated. This was done by applying the base process to different versions of the Randomized Battery Usage Dataset, which were created using different data sampling strategies.

A number of datasets were prepared from the processed Randomized Battery Usage Dataset, described in Section 3.2. This was done by first passing the dataset through the dataset-specific pre-processing pipeline described in Section 4.1.2, and then the general pre-processing pipeline described in Section 4.1.3. The dataset was then either sub- or super-sampled to simulate different data collection frequencies.

To control the sampling behaviour, a scaling factor $\beta = \{1/3, 1/2, 1, 2, 3\}$ was introduced. Sub-sampling was performed when $\beta \leq 1$, and was done by picking every $1/\beta$th data point from the original dataset. Super-sampling was done when $\beta > 1$, and was done by introducing new data points between the existing data points. Since these data points did not necessarily fall on reference cycles, their SOC and SOH values were estimated by linear interpolation. Super-sampling was done such that the number of total data points were multiplied by β. The new data points were inserted with uniform spacing between existing data points where possible. Data points that would be inserted at invalid positions, such as where data was removed as a result of data cleaning, were not included. See Figure 4.3 for an example of how the sub- and super-sampling was done.

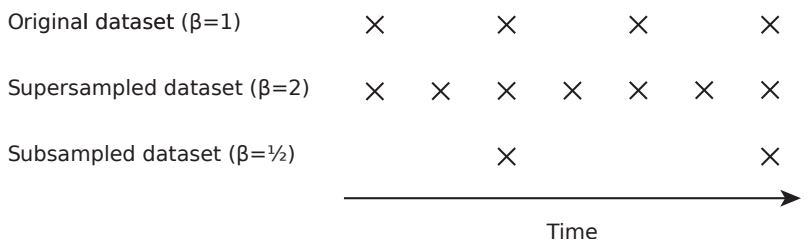


Figure 4.3: Example of how the data sub- and super-sampling was performed. Super-sampling was performing by inserting new data points, whilst sub-sampling was performed by selecting a subset of existing data points. Note that β corresponds to the relative size of the sub- or super-sampled dataset compared to the size of the original dataset.

The resulting datasets were then used to train models following the method described in Section 4.2.1. The set of possible hyperparameter values that were considered during hyperparameter selection are shown in Table 4.5.

Table 4.5: The hyperparameter search spaces for the models created using the base process on the Randomized Battery Usage Dataset.

Model	Parameter	Possible values
Random Forest	Num. trees	4, 8, 16, 24, 32, 64
	Max. depth	3, 6, 9, 12
	Min. num. samples to split	2, 4, 6
	Min. num. samples in leaves	1, 2, 4
XGBoost	Num. boosting iterations	
	Max. depth	3, 6, 9, 12
	Learning rate	0.01, 0.05, 0.1, 0.3
	Min. loss reduction to split	0.0, 0.1, 0.3
	Min. sum of child weights	1, 3, 5, 7

5 | Results

In this chapter, the results of the experiments described in the previous chapter are presented. The results for the experiments described in Sections 4.4.1 to 4.4.4 are presented in Sections 5.1 to 5.4, respectively.

5.1 Correlation study of the datasets

The Pearson correlation coefficients were computed for the Vehicle Fleet Dataset and for the Randomized Battery Usage Dataset, and are visualized in Figures 5.1 and 5.2 respectively.

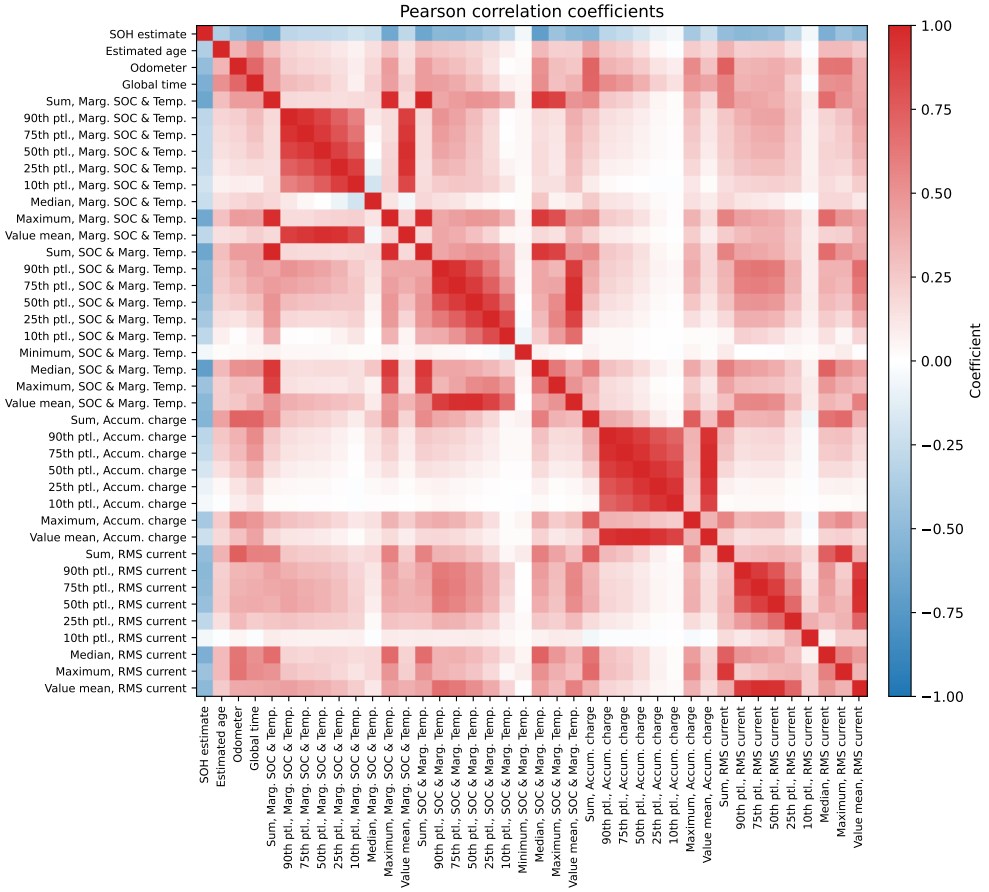


Figure 5.1: The Pearson correlation coefficients for the Vehicle Fleet Dataset.

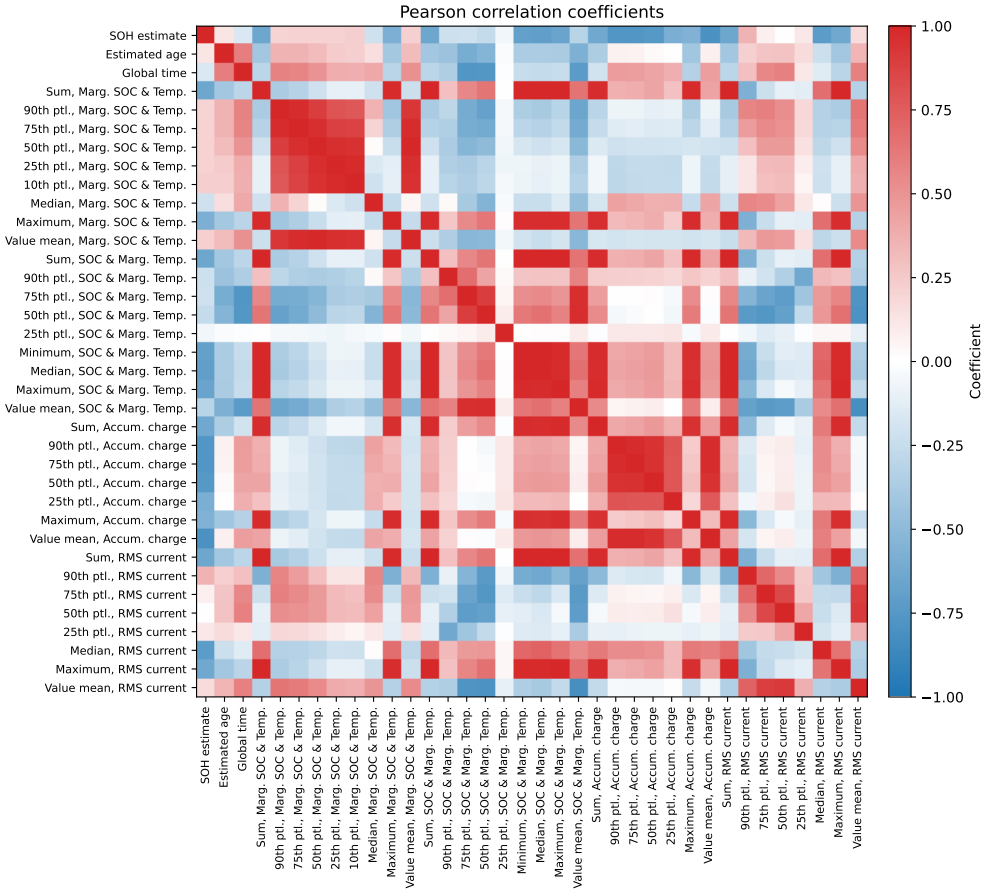


Figure 5.2: The Pearson correlation coefficients for the Randomized Battery Usage Dataset.

5.2 Model explanations for the Vehicle Fleet Dataset

Following the method described in Section 4.4.2, two different types of models were trained in this experiment. In total, 108 different Random Forest model candidates were trained using five fold cross validation for a total of 540 fits. 192 different XGBoost model candidates were trained using five fold cross validation, for a total of 960 fits. The best XGBoost candidate model was then retrained using the early stoppage criteria with a memory of the past five rounds. The model was trained for 145 iterations before being stopped by the criteria, and the training loss is shown in Figure 5.3.

The resulting optimal hyperparameters for both models are presented in Table 5.1. The optimal models were evaluated using RMSE and MAE on the train, validation and test sets. The results are presented in Table 5.2.

Feature importance scores and SHAP values were then computed for the models. The feature importance scores are presented in Figure 5.4. A global summary of the SHAP explanations for both models is presented in Figure 5.5. by computing the mean magnitude. The SHAP explanations are also presented as heatmaps in Figure 5.6.

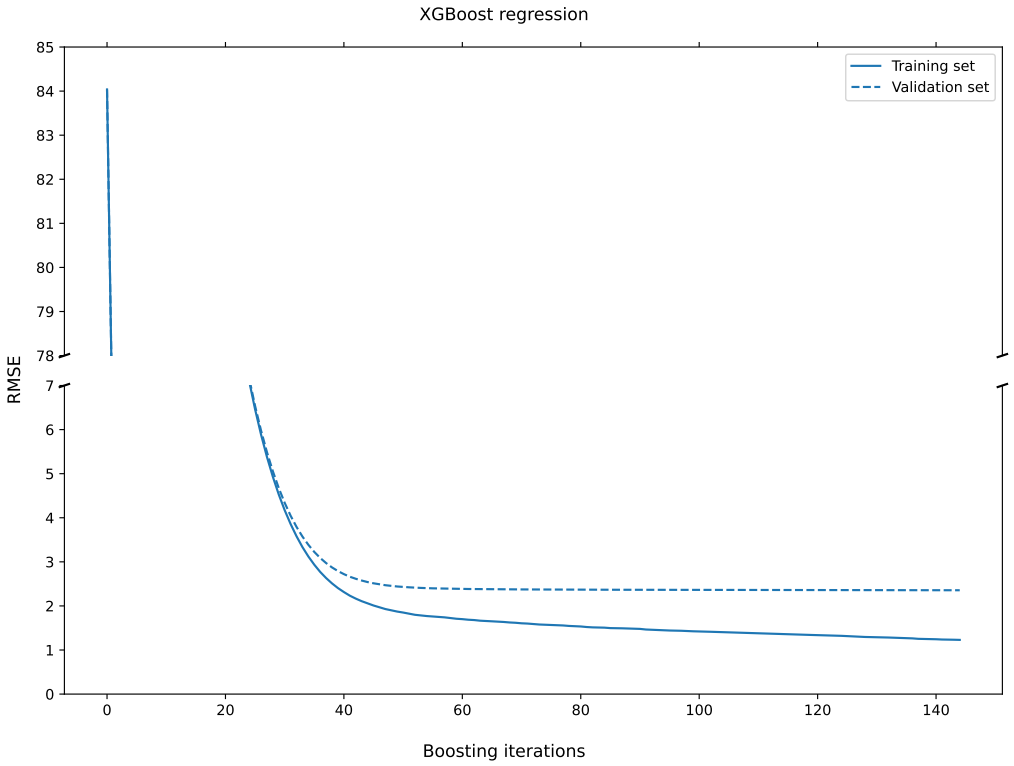


Figure 5.3: The XGBoost model's loss function values when trained on the Vehicle Fleet Dataset, computed at each boosting iteration for the training and validations sets.

Table 5.1: The optimal hyperparameters for the Random Forest and XGBoost models trained on the Vehicle Fleet Dataset.

Model	Parameter	Best value
Random Forest	Num. trees	192
	Max. depth	12
	Min. num. samples to split	2
	Min. num. samples in leaves	4
XGBoost	Num. boosting iterations	145
	Max. depth	12
	Learning rate	0.1
	Min. loss reduction to split	0
	Min. sum of child weights	3

Table 5.2: The final loss function values computed over the train, validation, and test sets, for the models trained on the Vehicle Fleet Dataset.

Model	Metric	Dataset	Value
Random Forest	RMSE	Train	2.18
		Validation	2.45
		Test	2.54
	MAE	Train	1.19
		Validation	1.34
		Test	1.34
XGBoost	RMSE	Train	1.25
		Validation	2.35
		Test	2.40
	MAE	Train	0.78
		Validation	1.28
		Test	1.29

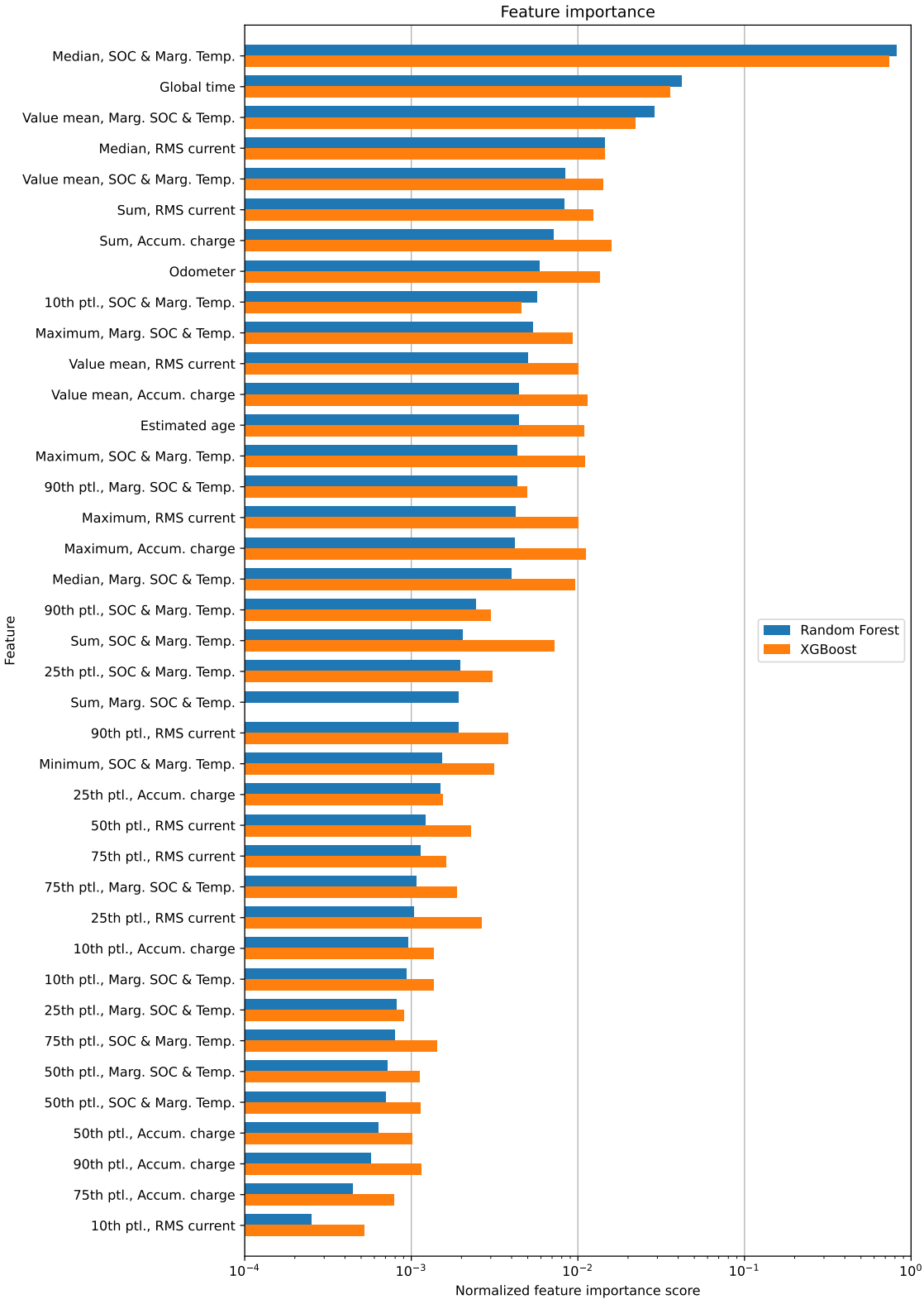


Figure 5.4: The feature importance scores for the models trained on the Vehicle Fleet Dataset.

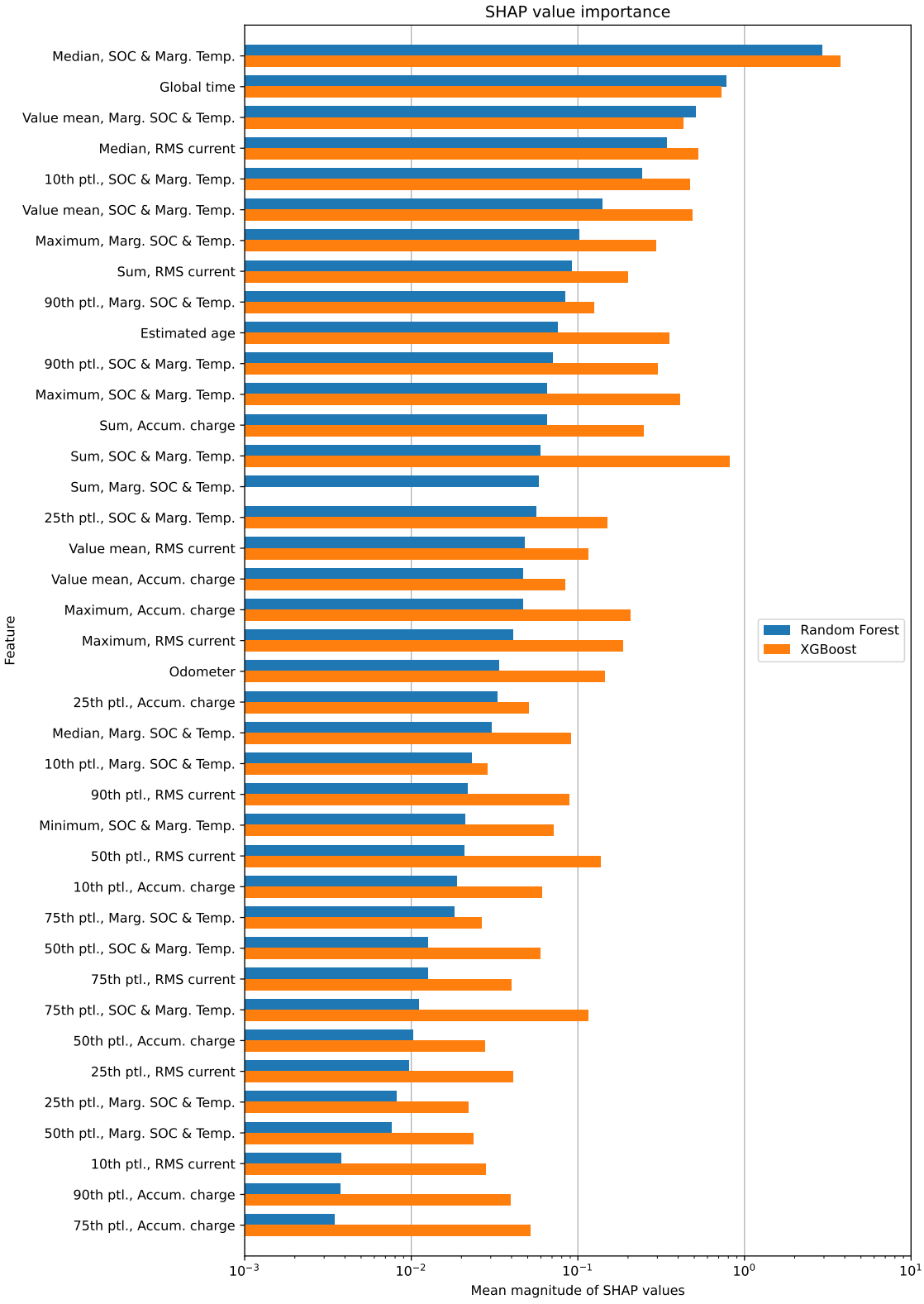
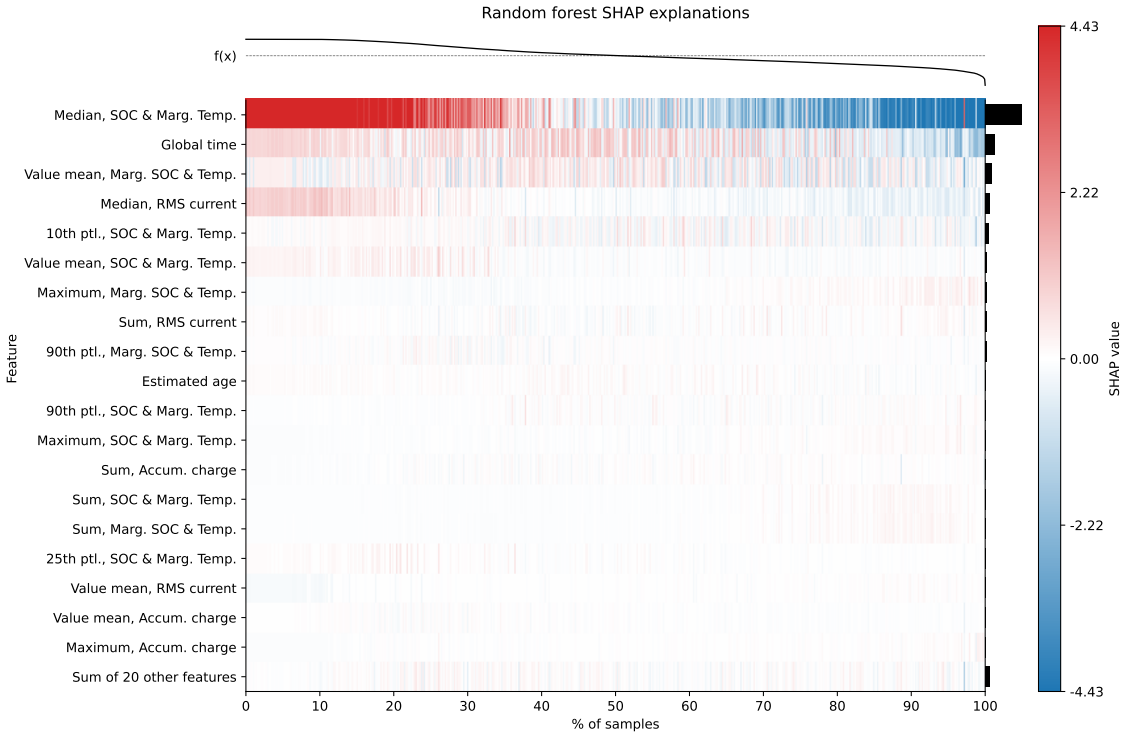
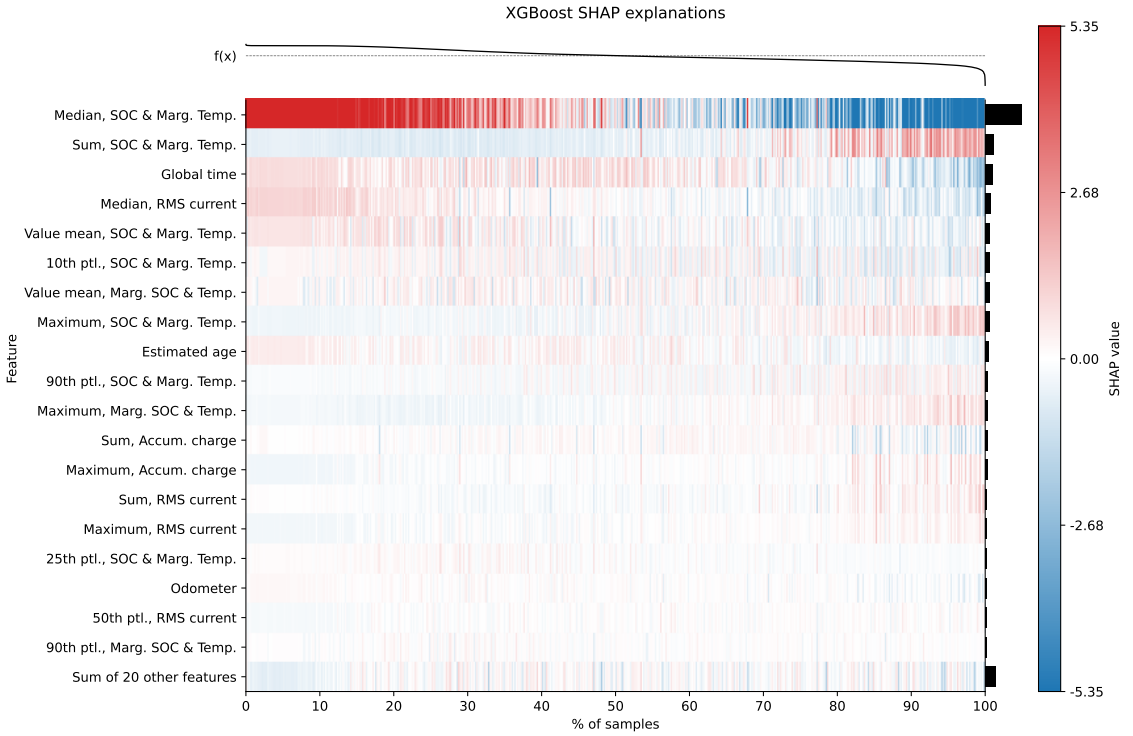


Figure 5.5: The mean magnitude of the SHAP values computed for the models trained on the Vehicle Fleet Dataset.



(a) Random Forest



(b) XGBoost

Figure 5.6: Overview of the SHAP explanations for predictions generated by the models trained on the Vehicle Fleet Dataset. The corresponding target value is shown above, and the mean magnitude to the right of the main figure. The color bar covers SHAP values between the 1st and 99th percentiles only, to improve readability by removing outliers.

5.3 Impact of sample weighting on model explanations for the Vehicle Fleet Dataset

This experiment was performing using the method described in Section 4.4.3. First, a set of DenseWeight models were fit to the data using $\alpha \in \{1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$. For the remaining hyperparameters, it was found that the bandwidth $h = 4 \cdot 10^8$ and lower weight bound $\epsilon = 1 \cdot 10^{-6}$ were good choices. Using the DenseWeight models, sample weights were then computed for the training dataset. Histograms of the resulting weight distributions are presented in Figure 5.7.

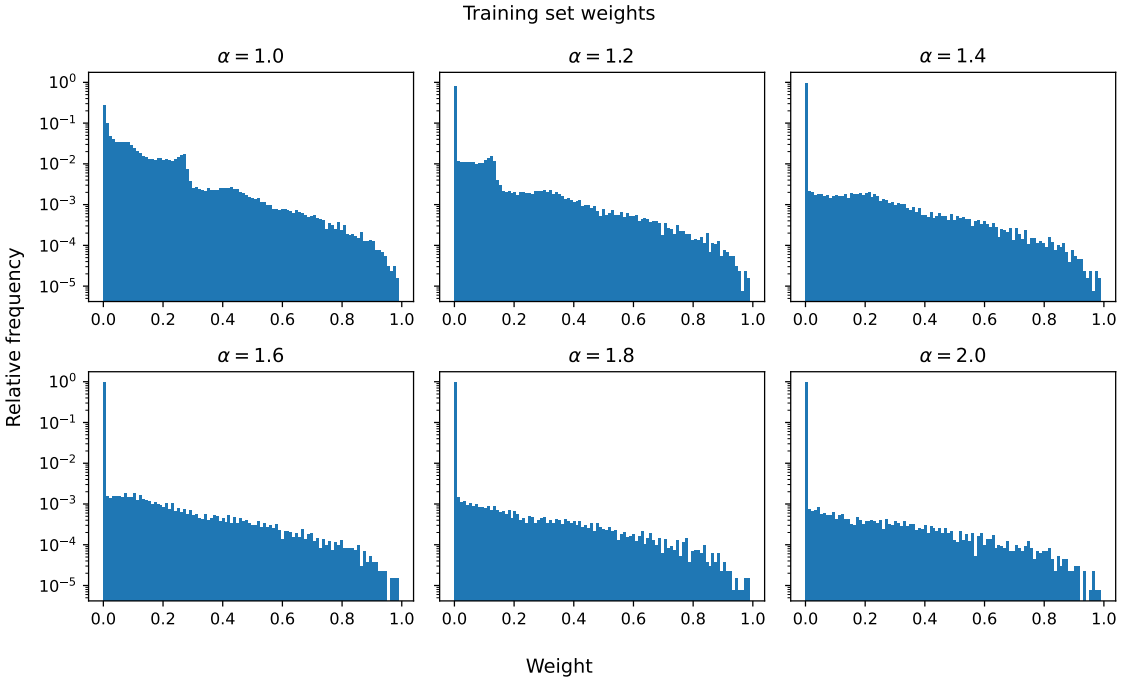


Figure 5.7: Distributions of the training dataset sample weights, presented as histograms for different α values.

The optimal models from the baseline experiment, presented in Section 5.2, were then retrained, but this time with the addition of sample weights. The XGBoost models trained for 110, 113, 103, 129, 147 and 125 boosting iterations for $\alpha \in \{1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$ respectively. The training losses for the XGBoost models are shown in Figure 5.8. After training, the models were evaluated using RMSE and MAE on the train, validation and test sets. The results are presented in Table 5.3.

Feature importance scores were then computed and are presented in Figure 5.9. SHAP values were also computed, and are presented as global explanations by their mean magnitude in Figure 5.10. To better highlight any trends with respect to α, the two metrics are also presented in trend plots in Figure 5.11. The SHAP explanations are also presented as heatmaps in Figure 5.12.

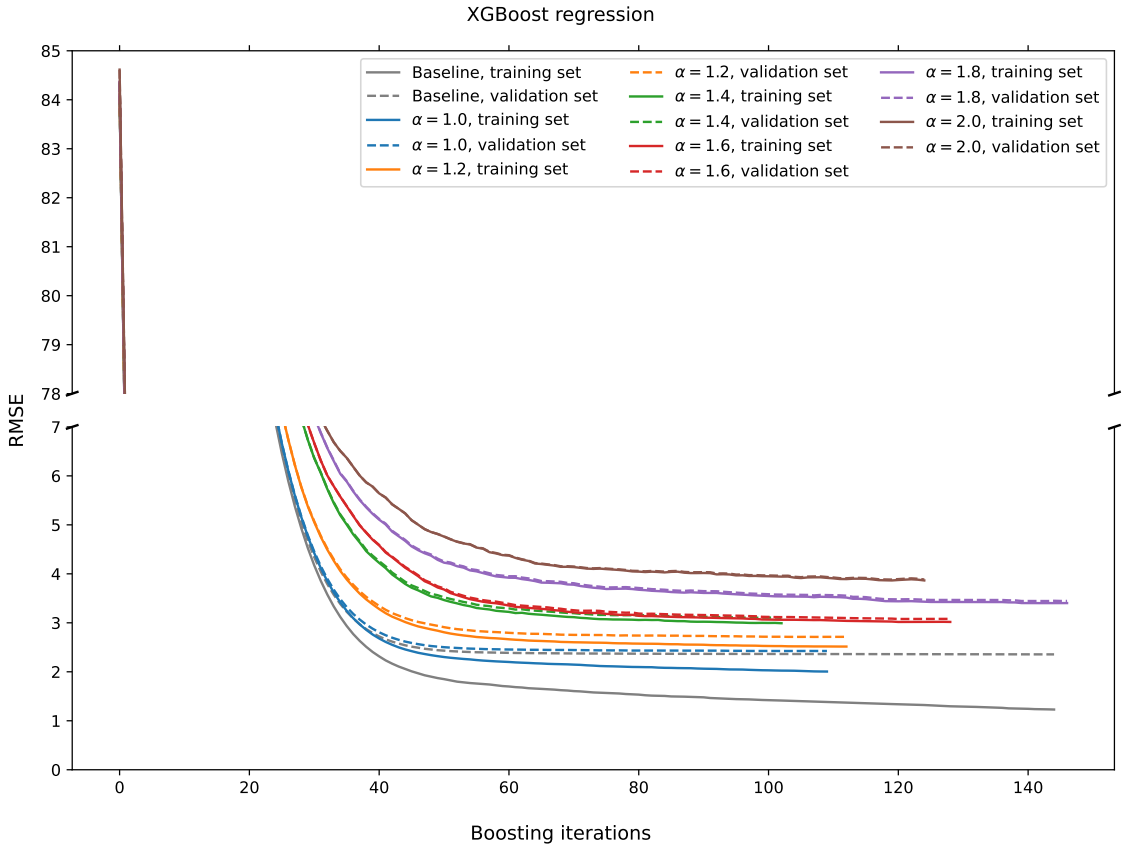


Figure 5.8: The weighted XGBoost models' loss function values when trained on the Vehicle Fleet Dataset using different α values, computed at each boosting iteration for the training and validations sets.

Table 5.3: The final loss function values computed over the train, validation, and test sets, for the weighted models trained on the Vehicle Fleet Dataset using different α values.

Model	Metric	Dataset	Values; $\alpha =$						
			Baseline	1.0	1.2	1.4	1.6	1.8	2.0
Random Forest	RMSE	Train	2.18	2.26	2.76	2.86	2.89	2.87	2.80
		Val.	2.45	2.49	2.87	2.96	2.98	2.95	2.88
		Test	2.52	2.55	2.95	3.03	3.04	3.02	2.94
	MAE	Train	1.19	1.27	1.81	1.95	1.98	1.95	1.84
		Val.	1.34	1.38	1.86	2.00	2.03	1.99	1.88
		Test	1.34	1.39	1.89	2.02	2.05	2.01	1.90
XGBoost	RMSE	Train	1.25	2.02	2.52	3.00	3.02	3.40	3.86
		Val.	2.35	2.42	2.71	3.10	3.08	3.44	3.88
		Test	2.40	2.49	2.78	3.17	3.16	3.51	3.95
	MAE	Train	0.78	1.15	1.57	2.08	2.04	2.43	2.88
		Val.	1.28	1.34	1.69	2.15	2.10	2.47	2.91
		Test	1.29	1.36	1.69	2.17	2.12	2.51	2.94

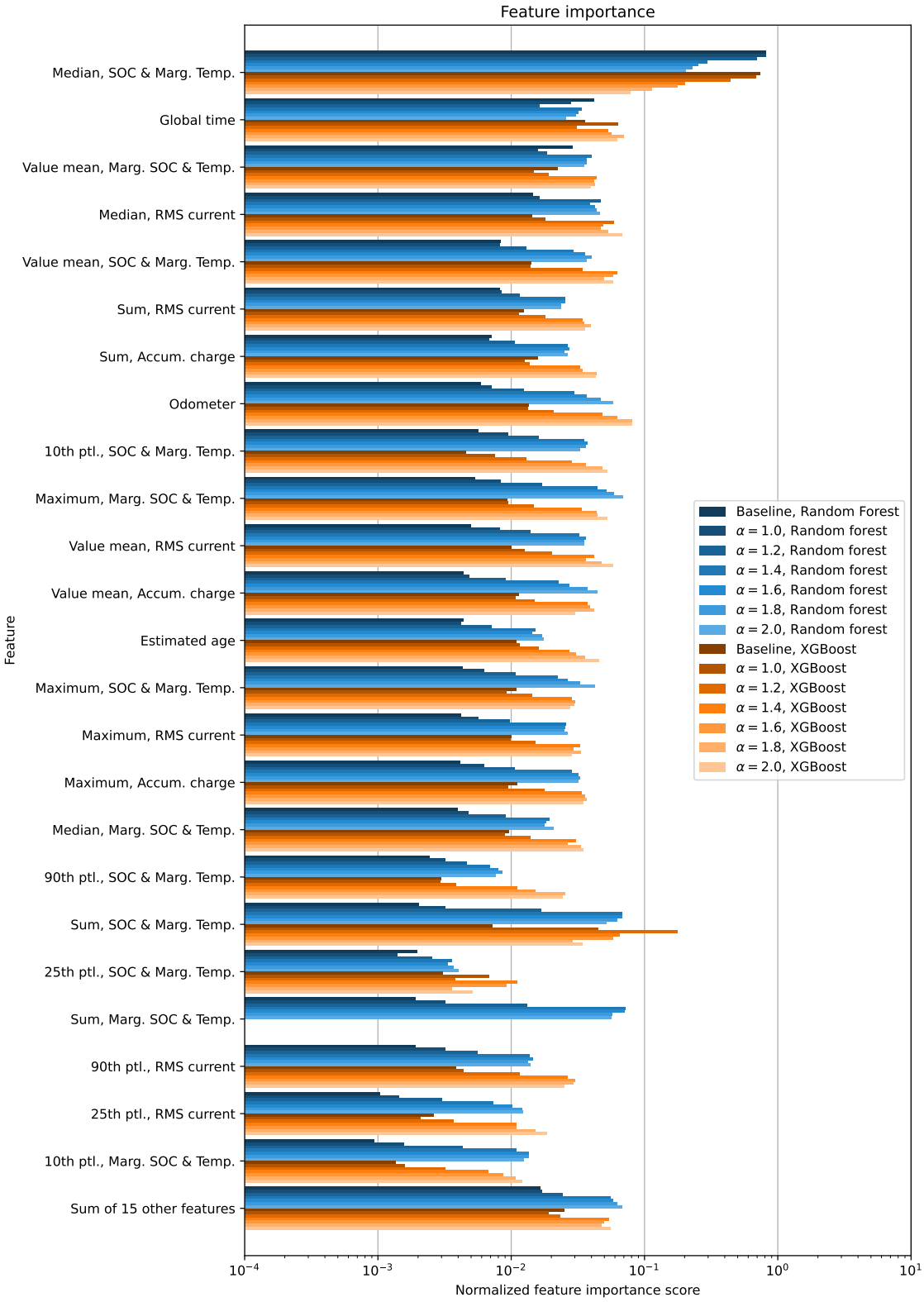


Figure 5.9: The feature importance scores computed for the weighted models trained on the Vehicle Fleet Dataset using different α values. Features for which all models have importance scores below $2 \cdot 10^{-2}$ are summed to improve readability.

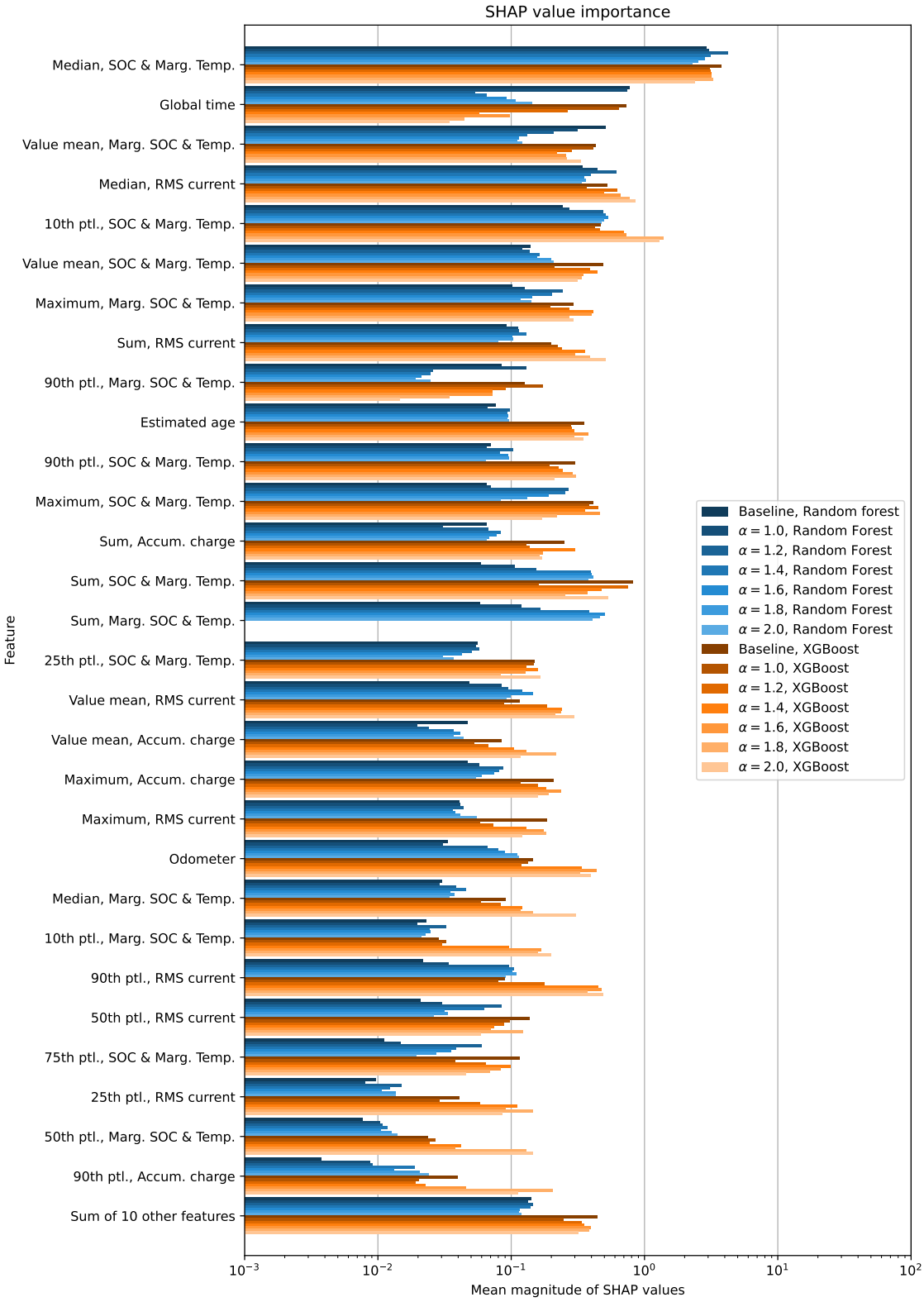
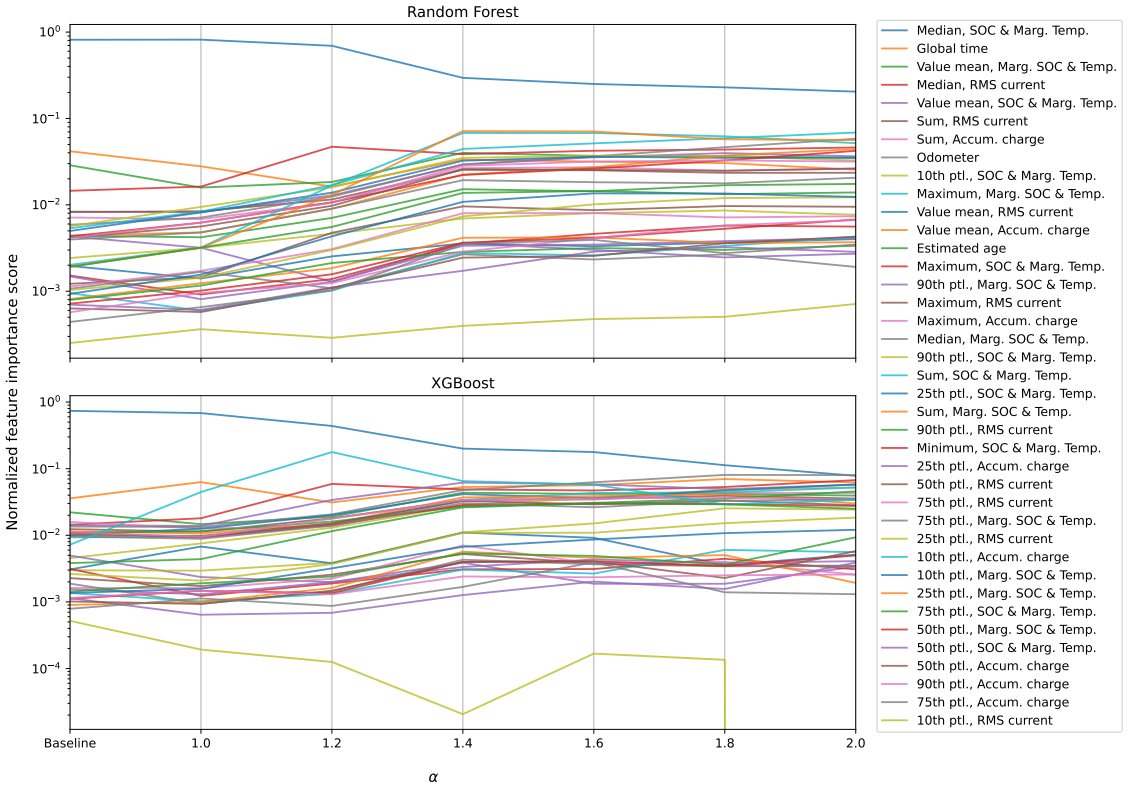
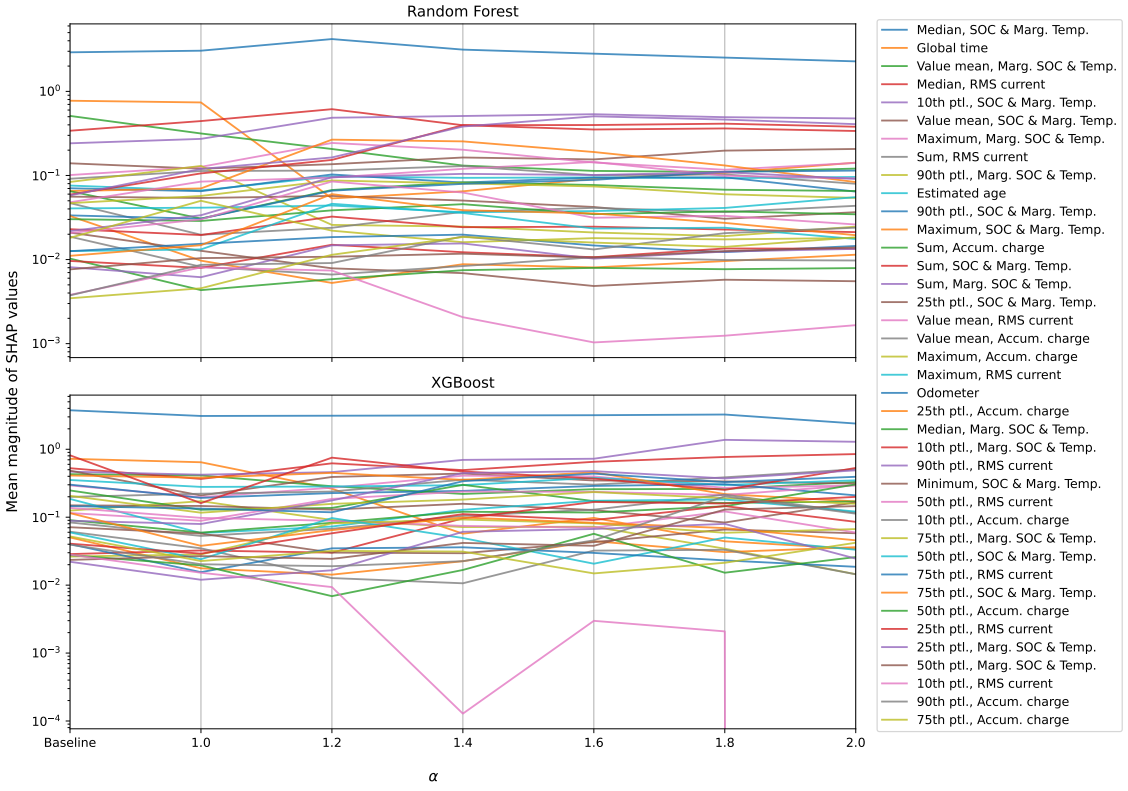


Figure 5.10: The mean magnitude of the SHAP values computed for the weighted models trained on the Vehicle Fleet Dataset using different α values. Features for which all models have importance scores below $2 \cdot 10^{-2}$ are summed to improve readability.

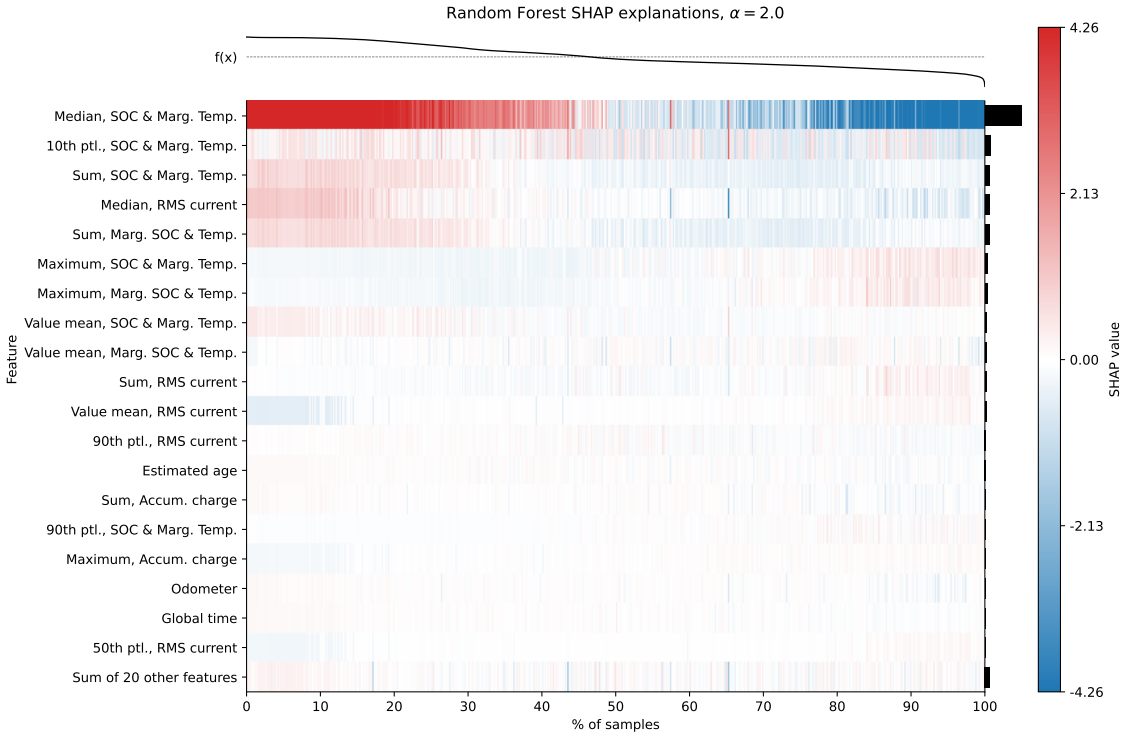


(a) Normalized feature importance score

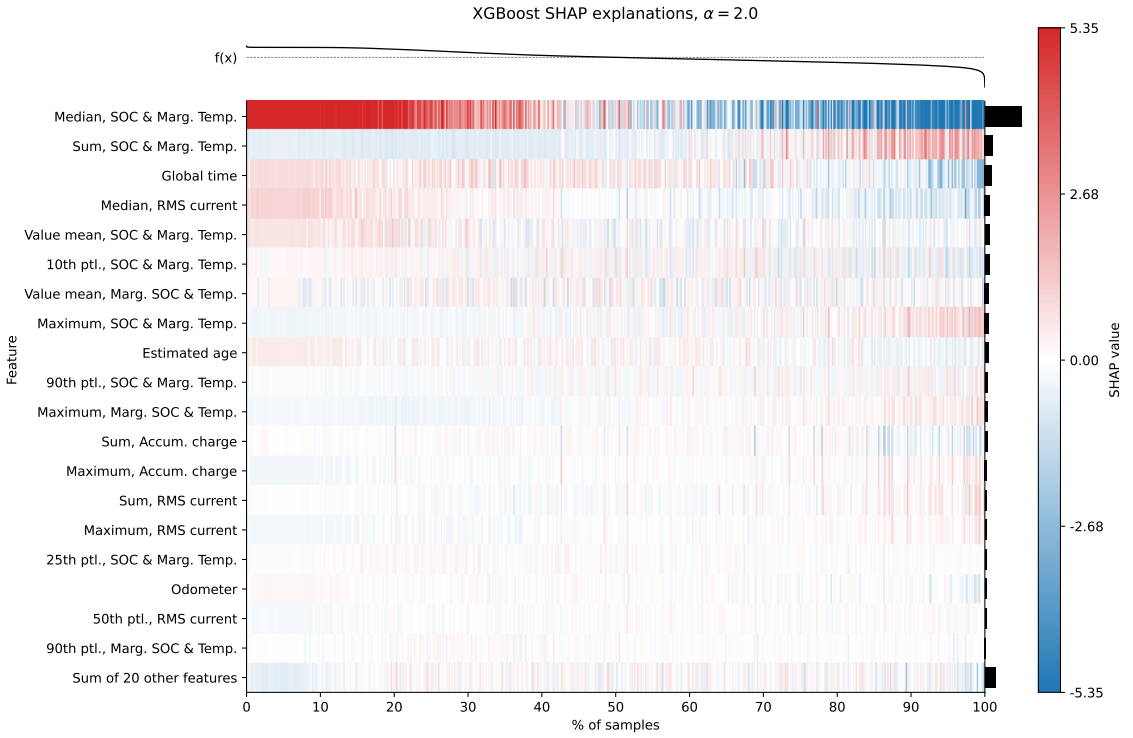


(b) Mean magnitude of SHAP values

Figure 5.11: Trend plots of the two importance metrics computed for the weighted models trained on the Vehicle Fleet Dataset using different α values.



(a) Random Forest



(b) XGBoost

Figure 5.12: Overview of the SHAP explanations for predictions generated by the weighted models trained on the Vehicle Fleet Dataset using $\alpha = 2.0$. The corresponding target value is shown above, and the mean magnitude to the right of the main figure. The color bar covers SHAP values between the 1st and 99th percentiles only, to improve readability by removing outliers.

5.4 Impact of sampling frequency on model explanations for the Randomized Battery Usage Dataset

This experiment was conducted following the method described in Section 4.4.4. First, five datasets were prepared from the pre-processed Randomized Battery Usage Dataset using different β values. A summary of the sizes of the resulting datasets is presented in Table 5.4.

Table 5.4: Size summary of the re-sampled Randomized Battery Usage Datasets.

Dataset	Number of samples; $\beta =$				
	$1/3$	$1/2$	1	2	3
Training set	144	219	439	868	1300
Validation set	58	90	181	358	533
Test set	62	96	192	379	567

For each of the datasets, one Random Forest model and one XGBoost model were trained. In total, 216 different Random Forest candidate models were trained using five fold cross validation for a total of 1080 fits. 192 different XGBoost models were trained using five fold cross validation, for a total of 960 fits. This was done for each of the five β values. The best XGBoost candidate models were then retrained using the early stoppage criteria with a memory of the past five rounds. This resulted in the models training for 80, 35, 67, 77 and 22 boosting iterations for $\beta = 1/3, 1/2, 1, 2$ and 3 respectively, before being halted by the criteria. The training losses for the XGBoost models are shown in Figure 5.13.

The resulting optimal hyperparameters are presented in Table 5.5. The optimal models were evaluated using RMSE and MAE on the train, validation and test sets. The results are presented in Table 5.6.

Feature importance scores were then computed and are presented Figure 5.9. SHAP values were also computed, and are presented as global explanations by their mean magnitude in Figure 5.10. To better highlight any trends with respect to β, the two metrics are also presented in trend plots in Figure 5.11. The SHAP explanations are also presented as heatmaps in Figures 5.17 and 5.18.

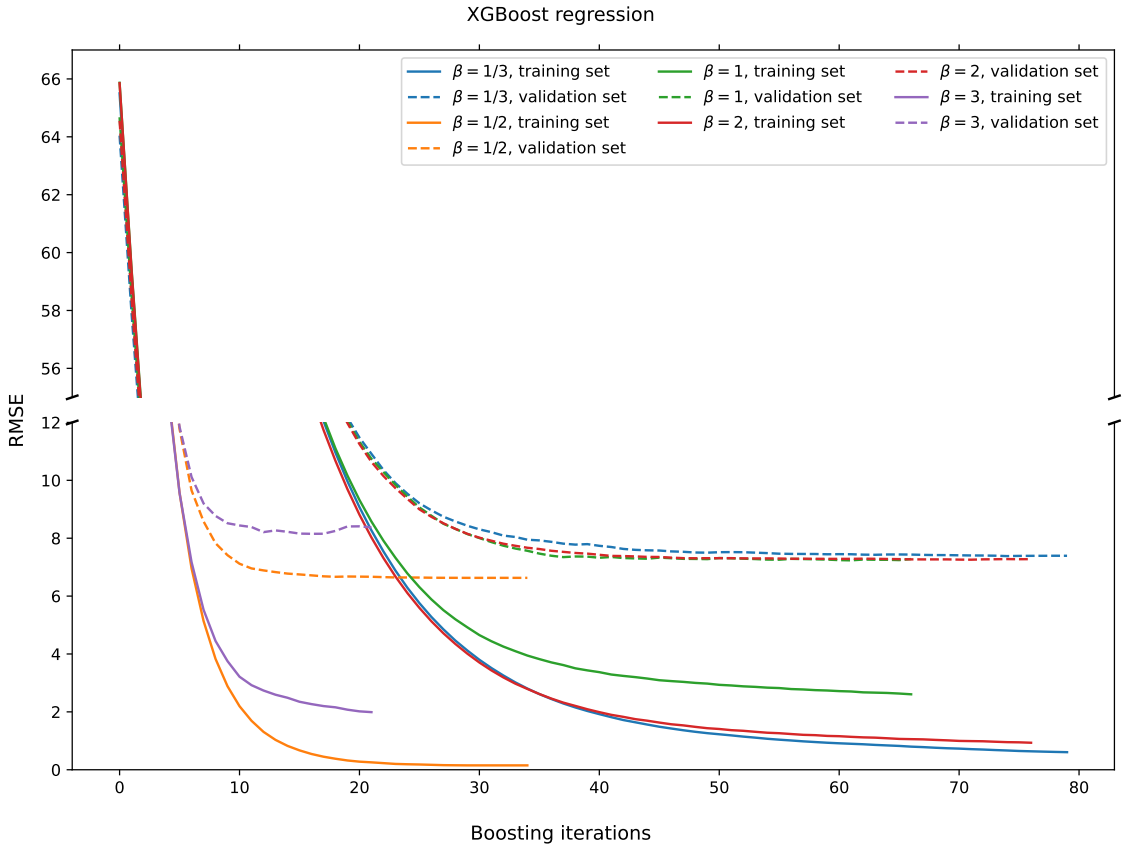


Figure 5.13: The XGBoost models' loss function values when trained on the Randomized Battery Usage Dataset re-sampled using different β values, computed at each boosting iteration for the training and validations sets.

Table 5.5: The optimal hyperparameters for the and XGBoost models trained on the Randomized Battery Usage Dataset re-sampled using different β values.

Model	Parameter	Values; $\beta =$				
		$^1/_3$	$^1/_2$	1	2	3
Random Forest	Num. trees	8	64	8	16	64
	Max. depth	6	6	9	12	12
	Min. num. samples to split	2	6	6	6	2
	Min. num. samples in leaves	4	1	2	1	1
XGBoost	Num. boosting iterations	80	35	67	77	22
	Max. depth	12	12	12	12	3
	Learning rate	0.1	0.3	0.1	0.1	0.3
	Min. loss reduction to split	0.1	0.1	0	0.1	0
	Min. sum of child weights	7	1	7	5	3

Table 5.6: The final loss function values computed over the train, validation, and test sets, for the models trained on the Randomized Battery Usage Dataset resampled using different β values.

Model	Metric	Dataset	Values; $\beta =$				
			$^1/_3$	$^1/_2$	1	2	3
Random Forest	RMSE	Train	2.58	3.40	3.92	3.14	1.07
		Validation	7.76	8.01	7.98	7.99	7.28
		Test	3.81	3.90	6.74	5.27	5.64
	MAE	Train	2.08	1.72	1.24	0.72	0.28
		Validation	5.85	6.13	5.98	5.81	5.40
		Test	3.18	3.09	3.51	3.15	2.96
XGBoost	RMSE	Train	0.67	0.15	2.70	0.99	2.27
		Validation	7.38	6.63	7.23	7.26	8.15
		Test	3.65	3.93	7.31	6.40	6.79
	MAE	Train	0.46	0.11	0.66	0.28	1.56
		Validation	5.79	5.00	5.35	5.45	6.17
		Test	2.97	2.88	3.86	3.53	4.08

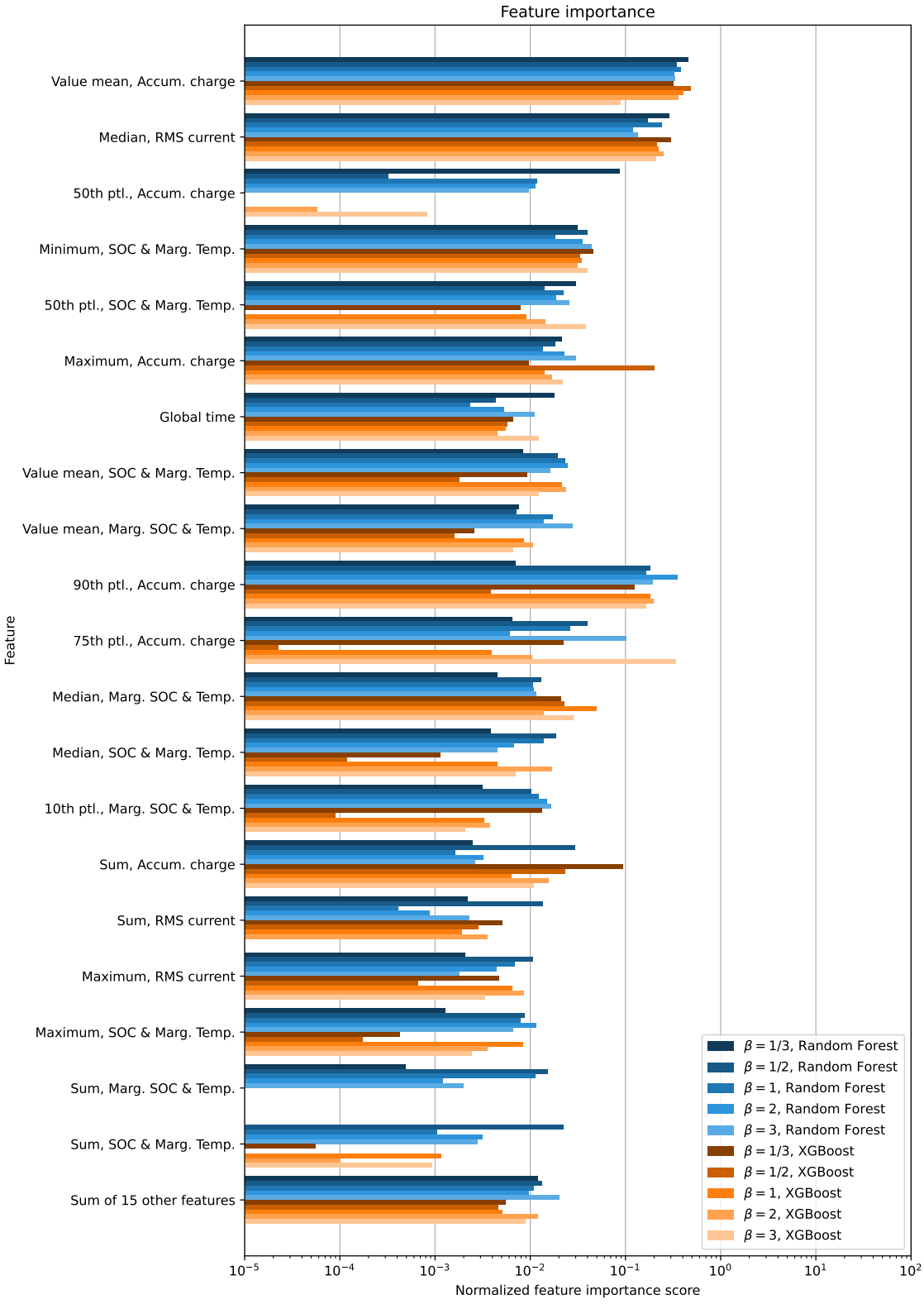


Figure 5.14: The feature importance scores computed for the models trained on the Randomized Battery Usage Dataset re-sampled using different β values. Features for which all models have importance scores below $1 \cdot 10^{-2}$ are summed to improve readability.

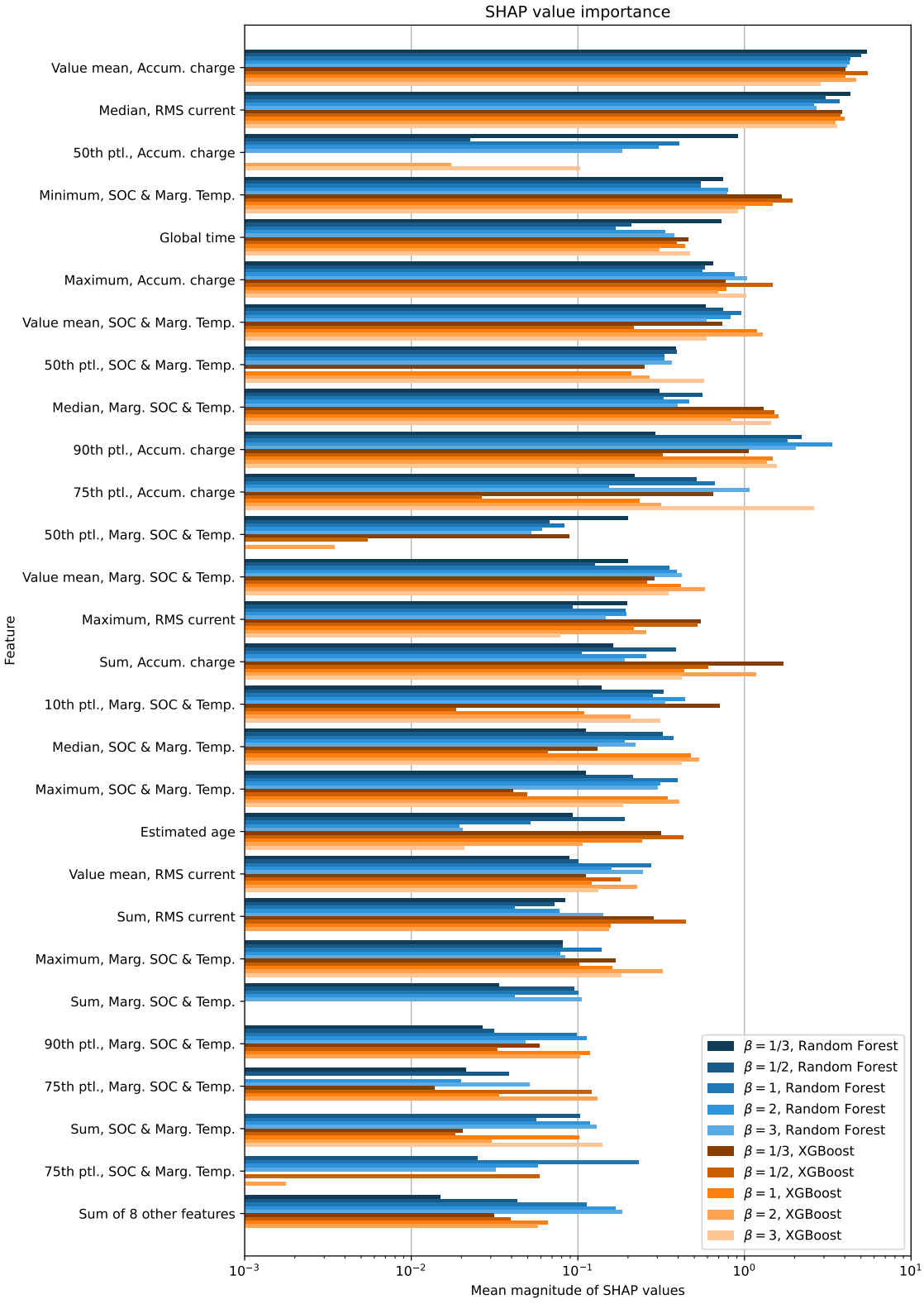
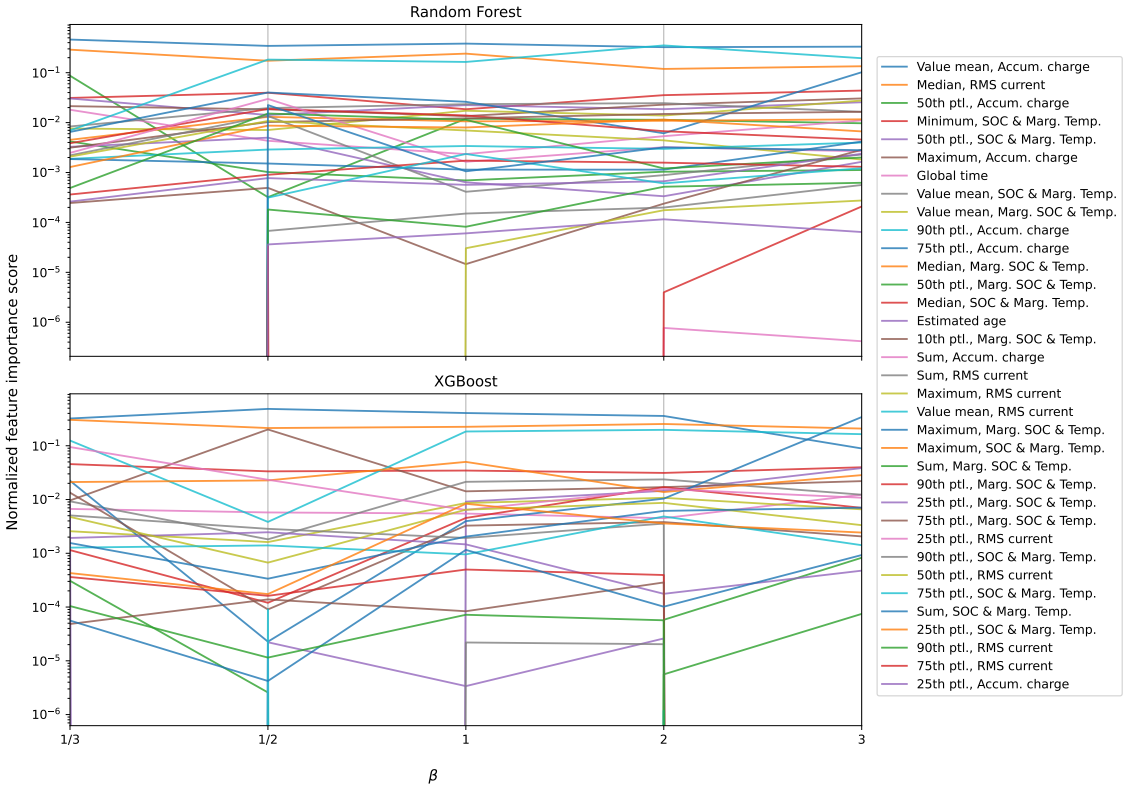
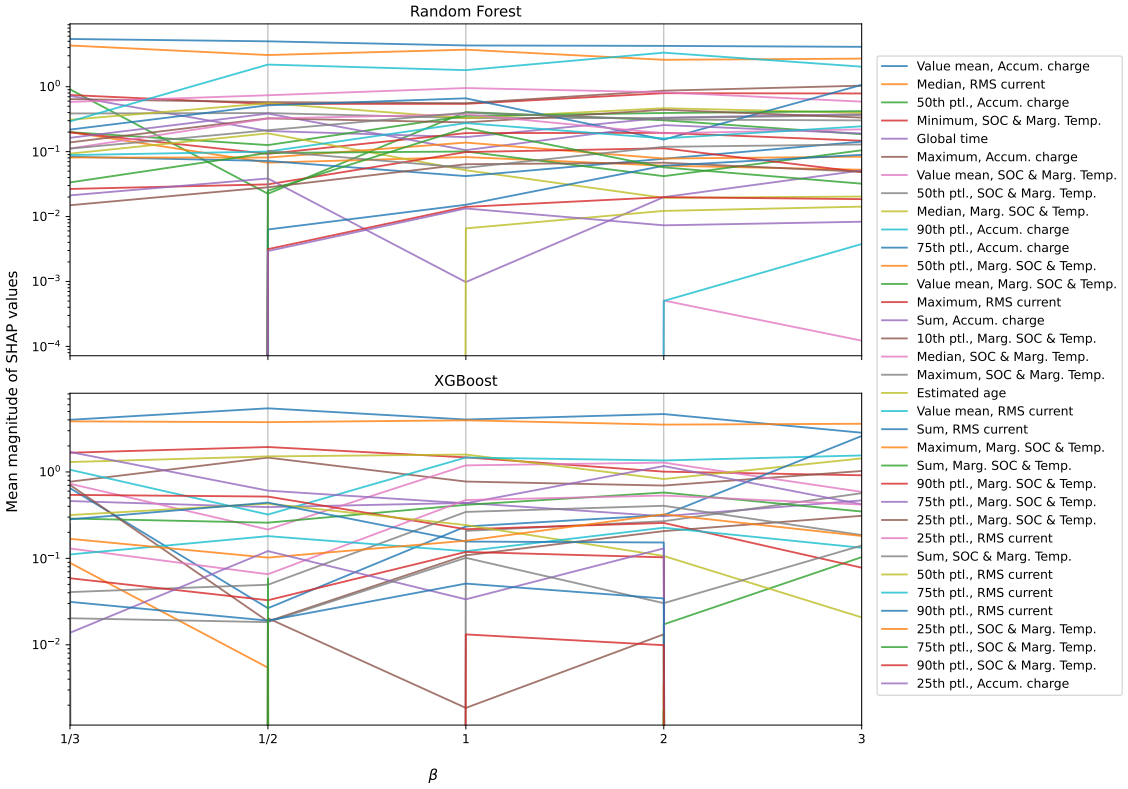


Figure 5.15: The mean magnitude of the SHAP values computed for the models trained on the Randomized Battery Usage Dataset re-sampled using different β values. Features for which all models have importance scores below $1 \cdot 10^{-1}$ are summed to improve readability.

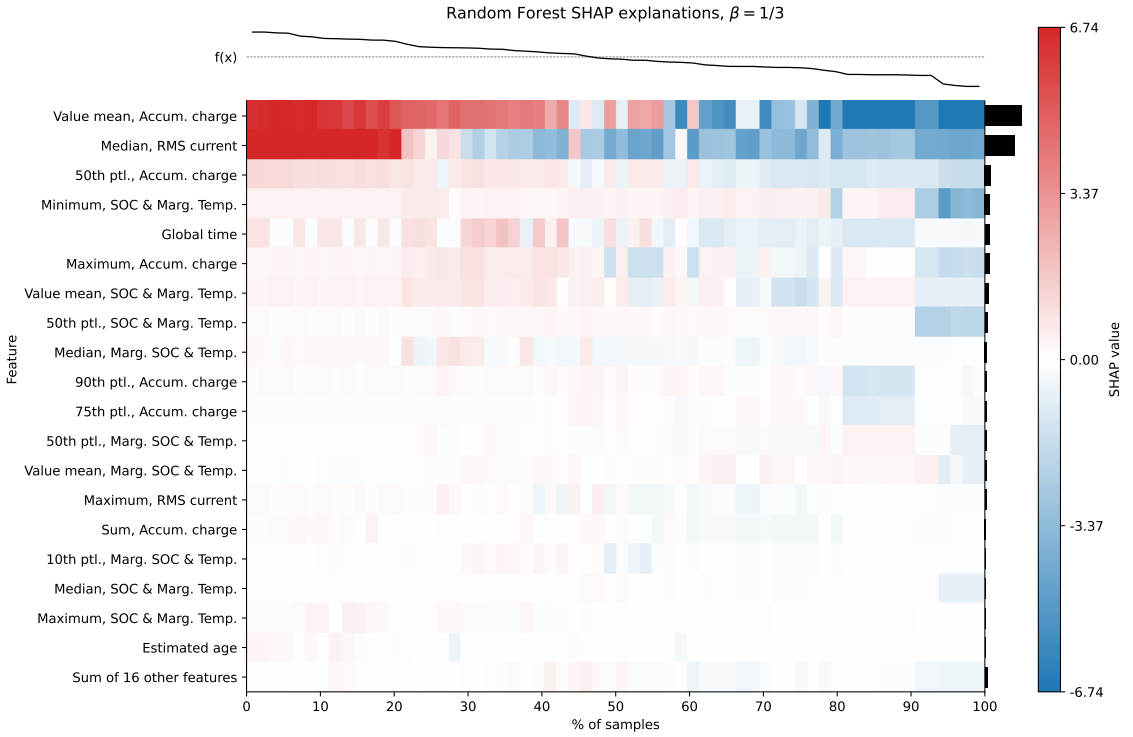


(a) Normalized feature importance score

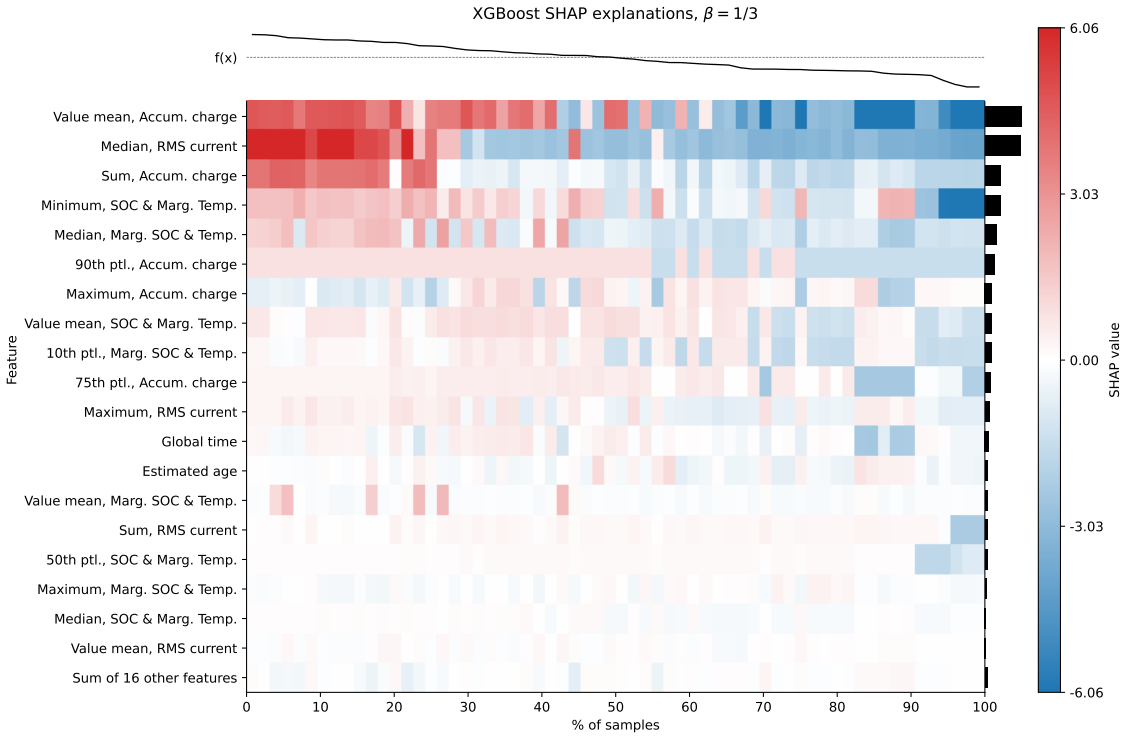


(b) Mean magnitude of SHAP values

Figure 5.16: Trend plots of the two importance metrics computed for the models trained on the Randomized Battery Usage Dataset re-sampled using different β values.

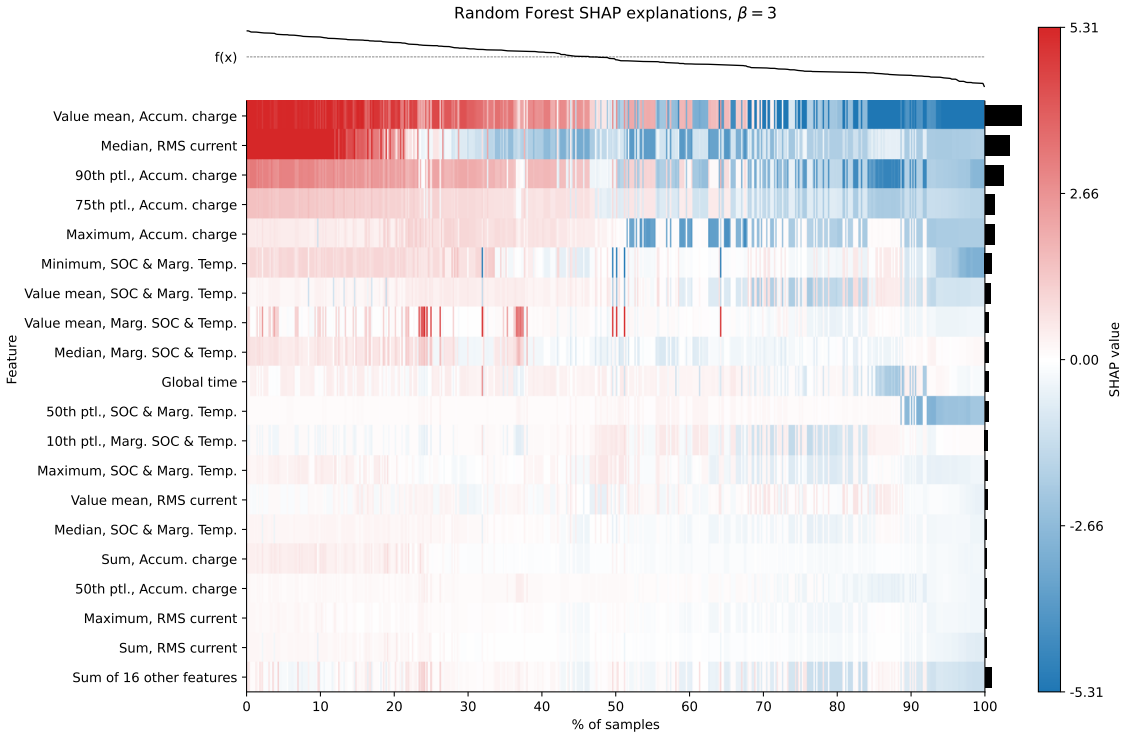


(a) Random Forest

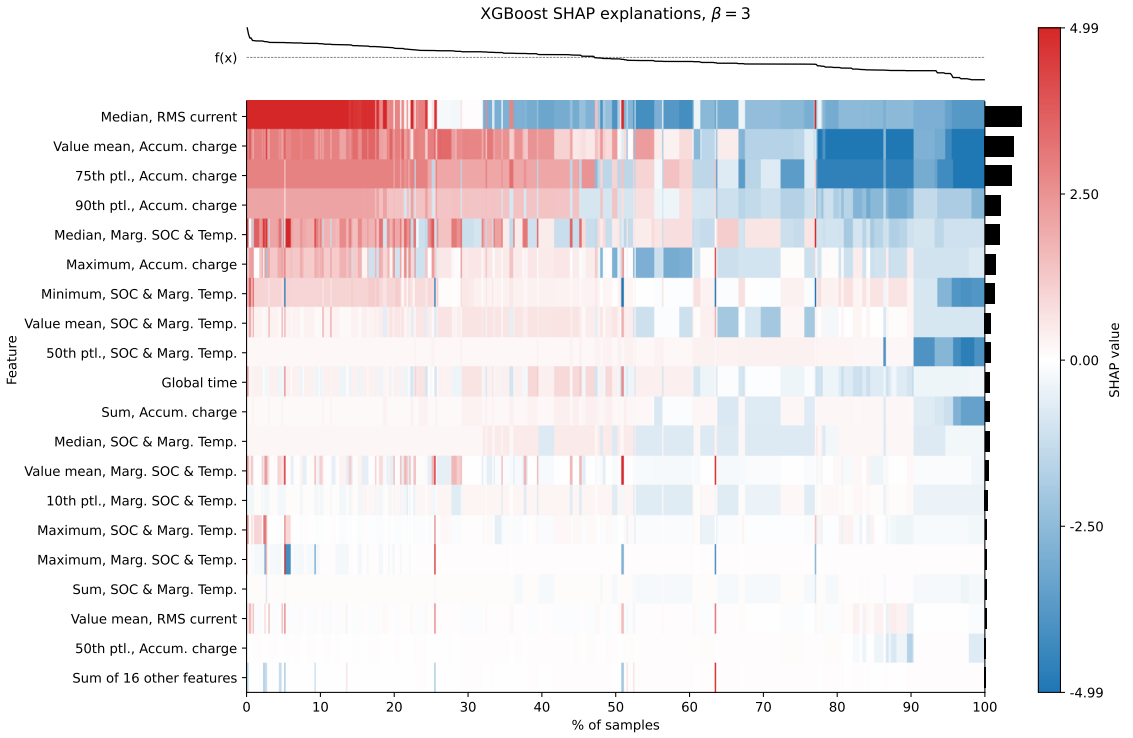


(b) XGBoost

Figure 5.17: Overview of the SHAP explanations for predictions generated by the models trained on the Randomized Battery Usage Dataset re-sampled using $\beta = 1/3$. The corresponding target value is shown above, and the mean magnitude to the right of the main figure. The color bar covers SHAP values between the 1st and 99th percentiles only, to improve readability by removing outliers.



(a) Random Forest



(b) XGBoost

Figure 5.18: Overview of the SHAP explanations for predictions generated by the models trained on the Randomized Battery Usage Dataset re-sampled using $\beta = 3$. The corresponding target value is shown above, and the mean magnitude to the right of the main figure. The color bar covers SHAP values between the 1st and 99th percentiles only, to improve readability by removing outliers.

6 | Discussion

In this chapter, the methodology and results presented in Chapters 4 and 5 are discussed. In Section 6.1, the datasets and pre-processing methodology are discussed. In Section 6.2, the results of the experiments are discussed.

6.1 Datasets and data pre-processing

In this section, some considerations and decisions that were made in regards to datasets and pre-processing are discussed. In Section 6.1.1, the Vehicle Fleet Dataset is discussed in detail, and in Section 6.1.2, the Randomized Battery Usage Dataset is discussed in detail.

When an estimated SOH value is used as the target variable, it is important to keep in mind that it is not a perfect indicator of the battery's real SOH. This is the case because estimating SOH always involves some degree of approximation. Under controlled conditions, the approximation can be accurately made by performing reference cycles, during which the battery capacity can be measured. For obvious reasons this cannot be done in vehicles that are in use, and hence, estimating the battery SOH of EVs is typically done with less accurate approximation methods.

The choice of which aggregation functions were used in the general pre-processing pipeline, or whether to use aggregation functions at all, was largely motivated by previous work in this project. The exact methodology that was used to derive these functions, or if any other function were considered, is not known. This could be an especially important topic to investigate further, since the aggregation functions have a major impact on the features and in turn how the models can be interpreted. For example, in their work from 2022, Zhang et al. used different aggregation functions such as the skewness and kurtosis [6]. However, there are also other major differences between the dataset that was used in their research and in this project.

6.1.1 Vehicle Fleet Dataset

Using actual vehicle data to train SOH estimators has both up- and downsides. The main benefit is that the data largely reflects true operating conditions, but it is important to remember that not all EV batteries behave the same. Different cell chemistries behaves differently, and the way that the battery is used may also be different between vehicles.

The Vehicle Fleet Dataset consists of measurements that came from PHEVs, that rely in part on an ICE for propulsion. This may in turn affect the high voltage battery usage, In particular, it might be deceiving to assume that higher feature values correspond to an increased usage of the high voltage battery used to power the electric engine. Since the battery only contains a limited amount of charge, the ICE may be used in conjunction with the electric engine, or on its own for longer driving cycles. In practice, this makes it harder to interpret and compare the results.

As mentioned in Section 3.1, there is generally an overrepresentation of cars with a single or few measurements in the Vehicle Fleet Dataset. This can be attributed to the relatively young age of the cars in the dataset, and new cars being added as they were produced.

One observation that does not fit into this hypothesis is the presence of a smaller number of cars that have a large number of measurements. While the exact cause of this was not investigated in this project, one idea is that is that it could be the result of faulty cars. When cars run as they run as they should, they will typically only visit a service center once a year. Hence, if a car has many measurements, it may suggest it has visited the service center repeatedly due to issues or faults. If this was the case, it could affect the predictions models' performance negatively. Even if the cars in question are few in numbers, they have relatively many corresponding measurements. It is unknown how much of an impact this actually has on the results, especially since the overwhelming majority of vehicles do not show this behaviour.

6.1.2 Randomized Battery Usage Dataset

The Randomized Battery Usage Dataset was collected in a controlled environment, and hence, the influence of external factors is limited when compared to real vehicle data. There are however some other major differences between the datasets that are important to keep in mind.

Firstly, even if the Randomized Battery Usage Dataset also used Lithium-ion cells, there can still be large differences in the the construction and chemistry of the cells. This will in turn affect the cell ageing behaviour, which makes it difficult to compare them to vehicle batteries. It should also be noted that the Randomized Battery Usage Dataset contains measurements of individual cells, while the vehicle data that was used in this project is typically for an entire battery pack.

When converting the time series data into histogram data, a number of design decisions had to be made. Perhaps the most influential of these were defining the bin limits, which in turn shaped the resulting histograms. Ideally, the constructed histograms would be as similar as possible to those of the real vehicle data. In reality, this was not possible as the conditions that the battery cells were cycled under differed from that of real vehicle data.

Another design decision related to the histograms were how to define what to actually count. In the vehicle data, many of the histograms count the time spent in certain states. This is difficult to reconstruct for the Randomized Battery Usage Dataset however, as there is no obvious way of interpreting terms such as "parked

time" and "running time" outside the domain of EVs. Instead, all measurements were considered for all histograms in this project, disregarding the battery usage state.

The pipeline for computing higher order features, as described in Section 4.1.2, is not entirely different from how these features are computed in the cars, but there are some known differences. The largest difference by far is how the charge capacity is estimated. This measurement is essential for computing SOC and SOH, and is computed by coulomb counting during references cycles in the Randomized Battery Usage Dataset. A similar method is employed in cars, but since the batteries are not regularly fully discharged and recharged, it requires some additional approximation. This may introduce an estimation error, but it also means the SOC and SOH estimate can be continuously updated using new information.

In the Randomized Battery Usage Dataset, the SOC and SOH estimates instead rely solely on the reference cycles to act as ground truth measurements, and all other data points are interpolated from these. The method of interpolation was selected as linear for this project, which is most likely not very accurate, but it was deemed sufficient given the relatively high frequency of reference cycles. It would be interesting to test other, perhaps more advanced, methods of interpolation and SOC/SOH estimation as well. This could also allow for using different datasets that do not contain reference cycles.

6.2 Experiments

In this section, the results of the experiments are discussed. The results of the experiments presented in Sections 5.1 to 5.4 are presented in Sections 6.2.1 to 6.2.4, respectively.

6.2.1 Correlation study of the datasets

Figures 5.1 and 5.2 shows the Pearson (linear) correlations between the target and all features, and highlights some interesting properties of the datasets.

Both datasets have distinct "patches" of strong positive correlation, which suggests multi-collinearity between some of the feature variables. The variables in question are derived of the same underlying data signals, and since the aggregation functions that were used for pre-processing are simple aggregates, this behaviour is to be expected.

One of the main differences between the correlation patterns between the two datasets, is the prevalence of negatively correlated variables. For the Vehicle Fleet Dataset, the main negative interaction happens between the target variable and many of the feature variables. The same is not true for the Randomized Battery Usage Dataset, however, since in addition to the target variables, many feature variables are negatively correlated to each other.

There are a number of factors that could cause the observations mentioned above.

The choice of aggregation functions most certainly has a large impact for both datasets. For the Randomized Battery Usage Dataset, the choice of histogram bins will also matter. For example, the odd behaviour of *25th ptl., SOC & Marginalized Temperature* can likely be attributed to the choice of the lower bin limit.

It is also likely that there are a number of factors that are unrelated to the data pre-processing. For example, many of the features are computed from the same electrical signals, which can be the source of correlation between multiple otherwise unrelated features. Finding and explaining such factors are outside the scope of the project, but are always important to keep in mind when interpreting results.

6.2.2 Model explanations for the Vehicle Fleet Dataset

The first experiment acted as a baseline, both in terms of evaluating the model performance and how the models worked.

As seen in Table 5.1, models with higher capacities, such as those with a greater number of trees or larger maximum depth, were generally preferred. This supports the idea that the real function $f : \mathbb{R}^n \to \mathbb{R}$ that the regression model is attempting to approximate is complex. It also validates the assumption that the task requires regression models with high capacity, such as Random Forest and Gradient Boosting.

One catch with higher complexity models is the inherent risk of overfitting, where the model fails to generalize to previously unseen data. To offset this, the XGBoost model was trained with an early stoppage criteria. As can be seen in Figure 5.3, this was successful preventing overfitting by stopping training when the model had converged. The figure also shows signs of a slight model bias, as there is a separation between the training and validation loss at convergence. This indicates that there may be additional headroom to further increase the model complexity. This is further supported by the optimal models having hyperparameters that maximize model complexity, as can be seen when comparing Tables 4.4 and 5.1.

Table 5.2 shows the final evaluation metrics for both models. The models perform well across all data subsets, with the XGBoost model performing slightly better than the Random Forest model. This is consistent with the theory that is presented in Section 2.2.4. The models perform similarly to the Random Forest model developed by Zhang et al. in 2022 [6], if only slightly worse. This is to be expected, however, since the both models used in for this experiment were much less complex.

An additional observation to be made in Table 5.2, is that the validation set loss is higher than the test set loss for the XGBoost model. This is typically not what is to be expected, as using a validation set for hyperparameter selection introduces a bias in favour of this data, that is often reflected by lower loss. There could be many reasons why this behaviour is not observed, but since it is consistent also for the Random Forest model, which does not make use of the validation set, it believed to be due to dataset imbalance. The randomness in the data splitting method that is used to generate the subsets could result in some datasets containing more ”easy” data points compared to others. This may be exacerbated by preserving all data points belonging to each vehicle in a single split, as described in Section 4.1.1. This

hypothesis could be verified by examining the data split distributions. However, since the validation set is only used for selecting a single hyperparameter value, it was not deemed to be a sufficiently significant problem to further investigate in this project.

The feature importance metric presented in Figure 5.4 shows some interesting results when it comes to which features are important for making predictions. In particular, a single feature, *Median, SOC & Marginalized Temperature*, is *far* more important than all other features. While it is not unexpected for the feature itself to have a considerable impact on predictions, as the battery state-of-charge level is well known to have a considerable impact of the rate of ageing, it is unexpected for it be so dominant. As described in Section 2.1.2, other factors such as the ambient temperature, accumulated charge throughput, current, et cetera, are also known to have a major effect. Since the extensive knowledge of battery ageing is unlikely to be wrong, this suggests that some other factors are affecting the results. Two such potential factors were investigated in the two remaining experiments, but there likely exists more.

A final observation is that the feature importance scores were similar between the models, which suggests they have learned to perform the prediction task in similar ways – with one exception. The *Sum, Marginalized SOC & Temperature* feature is given an importance score for the Random Forest model, but not for the XGBoost model. The way to interpret this is that the feature does not appear in any of the trees in the forest, and hence, cannot contribute when making predictions. This means that when training the XGBoost model, it must have actively chosen to not use this feature. One potential explanation for this is that it opted not to use this feature in favour of other, strongly correlated features that give largely the same information. It can be seen in Figure 5.1 that the feature in question is strongly correlated to *Sum, SOC & Marginalized Temperature*. Indeed, it can be seen that this feature has a much higher importance when compared to the Random Forest model, for which the two features have roughly the same importance.

The SHAP value explanations are largely consistent with the feature importance metric. The order of importance is different for some features, but the *Median, SOC & Marginalized Temperature* feature is still far more dominant compared to the others. A behaviour that is very similar to the previously observed behaviour of the *Sum, Marginalized SOC & Temperature* and *Sum, SOC & Marginalized Temperature* features is also seen. There is generally a higher variance in scores for the less important features, but the magnitudes of these are still very low when compared to the dominant feature.

The heatmaps of the SHAP value shown in Figure 5.6, give a complementary perspective to the barplot. These show that the dominant feature remains dominant for predictions of both low and high SOH. Furthermore, it appears this feature is used equally much to push the prediction value up (seen as red), as it is to push it down (seen as blue). Some of the other features share this behaviour, such as *Median, RMS current*, while others do not. For example, *Global time* is mainly used to push predictions towards higher values. There are once again some differences

between the two models, but the overall behaviours are much the same.

The first experiment validates the general process presented in Chapter 4, and establishes a baseline for later comparisons. Explanations with feature importance and SHAP shows that the models are perhaps overly reliant on single features, in a way that does not reflect the theoretical knowledge of battery ageing behaviour.

6.2.3 Impact of sample weighting on model explanations for the Vehicle Fleet Dataset

The second experiment aimed to investigate how the previously mentioned dataset imbalance affects the model behaviours, and whether this can be offset by using weights in favour of less common data samples.

DenseWeight is an algorithm for estimating sample weights from continuous variables. The algorithm uses KDE to estimate the PDF of the data distribution, which requires a careful selection of the bandwidth. Typical rules of thumbs for selecting the bandwidth were found to not work particularly well in this experiment as they resulted in far too small values that were unable to capture any sort of feasible distribution. Instead, the value for the bandwidth was incrementally increased until the resulting weight distribution stabilized. At some point, increasing the bandwidth was infeasible since it would result in extremely long computational times. It is likely that this value can be much further optimized however, and it could also be useful to consider other implementations that can use different bandwidths for each variable.

Histograms of the resulting weights are seen in Figure 5.7, and show that that the weight distributions are indeed imbalanced. The large spike at the minimum weight threshold (for $\alpha = 1.0$) supports the idea that a large amount measurements share similar feature values, as they are given similar weights. This observation could be explained by an overrepresentation of newer cars, supporting the original hypothesis, but additional experiments and analysis would have to be done to draw any decisive conclusions.

The optimal hyperparameters from Experiment 1 were reused for all weighted models trained in Experiment 2. This was done to make comparisons between the models as straight forward as possible, however, it could be equally interesting to see how the choice of optimal hyperparameters would change for the weighted models. Figure 5.8 shows that the weighted XGBoost models tends to be less complex than the baseline XGBoost model, but this could also simply be the higher data variance of less common data points triggering the early stoppage criterion.

Table 5.3 shows the final evaluations metrics compared to the baseline. The loss typically increases across all splits as the scaling factor α is increased. This is to be expected, since the models are still being evaluated on all data, but are trained to primarily focus on only a (smaller) subset of it. Another interesting observation is that Random Forest outperforms XGBoost for higher α values. The only major difference between the models that this can be attributed to, are the differences between bagging and boosting, as described in Sections 2.2.3 and 2.2.4. With bagging,

the Random Forest model has an equal probability of selecting each data sample, as the weights are only used for splitting nodes. This is not the case for the XGBoost model however, which selects data using gradients that are weighted by the sample weights. This would lead the model to focusing more on data with higher weights, which hurts its overall performance when evaluated on all data.

The feature importance scores for the weighted models are presented in Figure 5.9, and overall shows similar results to the baseline. The importance of the *Median, SOC & Marginalized Temperature* feature is still far ahead, and *Sum, Marginalized SOC & Temperature* is still missing for all of the XGBoost models. When comparing the results between the baseline and the weighted models trained using different α values, an interesting trend is observed. Figure 5.11a shows there there is a positive correlation between the α value and corresponding feature importance scores for most less important features, with exception of the dominant feature which sees a negative correlation instead. This suggest that while the weighted models still work similarly to the baseline models, the dataset imbalance may indeed contribute to the results as discussed in Section 6.2.2.

Compared to the feature importance scores, there is a greater deal of variance between the SHAP explanations for the different features, as can be seen in Figure 5.10. Figure 5.11b shows that the correlation pattern between α and the SHAP importance that was seen feature importance metric is not as clearly defined, especially for the Random Forest model. Moreover, the SHAP explanations for some features, such as *Median, SOC & Marginalized Temperature* and *Estimated age*, are largely unaffected by increasing α values.

The heatmaps for the SHAP explanations of the the heavily weighted models, shown in Figure 5.12, also supports this. When compared to Figure 5.6, the SHAP explanations for the XGBoost models are mostly similar, but there are considerable differences between the SHAP explanations for the Random Forest models. This suggest that even if the Random Forest models perform better than the XGBoost models for higher α values, their internal behaviours are still affected by training with sample weights.

The results of the second experiment shows that when training the models with sample weights computing using the DenseWeight algorithm, not only are the models performance affected, but also their behaviours. The results of the experiment seem to indicate that the overrepresentation of certain data skews the importance of certain features. While additional work would have to be done to establish an exact cause of this, some potential causes could be hypothesized. Under the assumption that there is an overrepresentation of newer vehicles in the dataset, it could be caused by certain degradation modes that are more active early on in the batteries' life. As described in Section 2.1.2, different degradation processes interact with each other, and some of this interaction may not happen until later in the batteries' life.

6.2.4 Impact of sampling frequency on model explanations for the Randomized Battery Usage Dataset

The third experiment was aimed at investigating the impact of the data sampling frequency on the models. For time series data, increasing the sampling frequency simply results in a correspondingly higher data resolution. A similar sampling frequency increase for data in the histogram format, as described in Section 1.1, is not as easy to interpret however. Since each sample is a recollection of all past signals, there is an implicit interaction between samples.

Table 5.5 shows that there is a lot of variance between the optimal hyperparameters for the models training on the different datasets. Notably, there is no clear correlation between dataset size and model capacity, as determined by the number of trees and model depth.

Figure 5.13 shows the loss curves for the XGBoost models. All models share the same general training behaviour, but notably, the models corresponding to $\beta = 1/2$ and 3 converged much faster than the others. It is unclear why these specifically were faster, but it only affects the rate of converge and not the final model loss as indicated by the models corresponding to $\beta = 1/3$ and $\beta = 2$ receiving better final evaluation scores. The figure also shows that there is a significant model bias, and at least in the case of the model corresponding to $\beta = 3$, there is an indication of overfitting as the validation loss is starting to increase.

Table 5.6 shows the final evaluation results for the models. The loss is in general much higher for the validation set compared to the test set. Since there are very few battery cells included in each split, this may simply be because some splits contain cells that are easier to predict than others. There are also some major differences when comparing the different metrics between different splits. Since RMSE is by nature more sensitive to fewer large deviations as opposed to many smaller deviations than MAE, this may suggest that there exists a set of data points that the models are making major prediction errors on.

The normalized feature importance scores are seen in Figure 5.14, and show a lot of variance between the features' importance values for different β values. While this makes it hard to spot any immediate trends, some of the same signs of multicolinearity between features are seen as before. For example, *50th percentile, Accumulated charge throughput* and *Sum, Marginalized SOC & Temperature* do not appear in the majority of the XGBoost models. As can be seen in Figure 5.2, there are indeed strongly correlated to a number of other features that do appear in the models.

Figure 5.16a paints a similar picture, but also further highlights trends (or lack thereof). In particular, some features such as *Value mean, Accumulated charge throughput* and *Median, RMS current* are mostly constant with respect to β. This indicates that these features have a consistent importance for the models' predictions regardless of the sampling frequency. An interpretation of this observation is that these features are truly important for making SOH predictions, as opposed less stable features that could just as well be artifacts of the sampling ratio. In a future experiment it would also be interesting to see if this behaviour is consistent between

different models that are trained on the same dataset.

The SHAP value explanations seen in Figure 5.15 with the corresponding trend plots in Figure 5.16b, are largely consistent with the feature importance metric and show the same behaviours. The heatmaps seen in Figures 5.17 and 5.18 further shows that despite some differences, the overall behaviours of the models trained on the sub- and super-sampled datasets are the same. The figures also highlight the large differences between the behaviours of the Random Forest and XGBoost models, which once again are most likely caused by the multicolinearity.

The general takeaway from this experiment is that the frequency at which histogram data is collected does appear to have a considerable impact on not just the performance, but also on the behaviours of the models. It is hard to draw any concrete conclusions, however, owning to the very small dataset, and high result variance. Connecting these results to the case of vehicle data collection also proves to be a challenge, as there are then a number of data and design choices that are fundamentally different. At the very least, the differences of the Randomized Battery Usage Dataset and actual vehicle data, and how these affects the results, would have to be better investigated before any general conclusions could be drawn.

7 | Conclusion

In this thesis, statistical SOH prediction models were developed and evaluated on a novel dataset gathered from over 63 000 Volvo PHEVs. The dataset consisted of battery measurements in a histogram format, which required additional aggregation steps to be converted into usable features. The initial results were promising, and the models performed similarly to those found in previous work, but when analyzed with feature importance and SHAP, some unexpected behaviours were observed. In particular, there was a very high reliance on one single feature for creating predictions.

To further examine the behaviours of the models, two additional experiments were conducted. The first experiment used sample weighting to weight the dataset in favour of less common data points, to offset an overrepresentation of certain signals. In the second experiment, the impact of the frequency at which the histograms were sampled were investigated. This was done with a separate laboratory generated dataset, which was processed into a format similar to that of vehicle data, before being either sub- and super-sampled to simulate different sampling frequencies.

The results showed that the dataset imbalance had an impact on how the models behaved, seemingly contributing to the above mentioned issue of overreliance on a single feature. Furthermore, it was shown that the data sampling frequency had a minor impact on the model performance, but a larger impact on the model behaviour. Owning to the small dataset size, and the large amounts of feature engineering that had to be performed to simulate the format of vehicle data, it is unclear how well this result translates to more general cases. It was also established that both datasets suffered from strong multicolinearity as a result of the complex feature aggregation, which is believed to have a considerable impact on the models' interpretability.

This thesis main point of contribution is two-fold. First, it has shown some of the limitations and potential pitfalls of state-of-health prediction models trained on electric vehicle histogram data, which may be used to guide future work within the field. Secondly, it has shown the viability of model-agnostic explainability methods as not only a tool to help interpret black-box models, but also to guide data engineering and modelling aspects of the data science process. This is not to say that the methods that were employed in this project will work in all situations, but the results show that they can be useful tools to gain further insights and better understand the data and the underlying problem.

7.1 Future work

Over the duration of this project, a number of unanswered questions and topics that would be interesting to explore further have appeared. To preserve the scope of this thesis, and due to a lack of time, not all of these have been picked up or thoroughly examined. There is however reason to believe that they could either affect the results of this project, or lead to entirely new results.

At the very start of the project, the prospects of using various intrinsically interpretable machine learning models was investigated. This idea was dropped in favour of model-agnostic interpretability methods, but some of the methods that were examined showed various degrees of promise. One very promising set of methods was *symbolic regression*. The idea behind such methods is to directly learn a symbolic expression that best fits the data. The symbolic expression would hence be directly interpretable, as it would give complete insight into how the model works in a fully reproducible way.

The main problem with symbolic regression is that it requires searching through an enormous amount of different symbolic expressions to find those that fit the data well. However, recent research within the area of symbolic regression has shown that if the underlying problem is that of a physical system, various techniques can be used to speed this process up, making it possible to discover complex symbolic relationships. AI Feynman [21], is an algorithm that works by utilizing certain mathematical properties which are common in physical equations. It would be interesting to see if such a method could be used to establish symbolic formulas for state-of-health prediction.

A major hurdle that was encountered during the project was to locate suitable datasets that the results for the vehicle fleet dataset could be compared against. The best suitable candidate that was found, the Randomized Battery Usage Dataset, which is described in Section 3.2, is still quite different from the vehicle fleet dataset and required major feature engineering and pre-processing to be at all usable. For this reason it was only used for one isolated experiment, even if ideally, all results would be benchmarked against a comparable, but different dataset.

The methodology for pre-processing the Randomized Battery Usage Dataset could also be improved. For example, the vehicle fleet dataset uses various time measurements such as ”parked time” and ”running time” for the histogram features. This is not reflected in the method used to compute the histograms for the Randomized Battery Usage Dataset however, something that would most likely have an impact on the result were it to be changed.

Another potentially interesting modification to this project would be to see how the results would change if the methods were applied to a dataset that collected over a longer time frame. The idea is that more distinct ageing patterns could have time to develop over this period, which could be picked up by the models. It would also be interesting to have access to a dataset in which samples are generated at a higher frequency, such that the compromises that had to be made with the Randomized Battery Usage Dataset could be completely circumvented. This would

require fundamentally changing how data is collected, however, which would affect many more levels of the data flow than what are touched on in this project.

Similarly, having access to the actual signals that are used on-board to compute the histogram data could enable comparisons between the two data formats. Even if the data is not collected from cars, such a comparison could be still made with for example the Randomized Battery Usage Dataset. This was not done in this project as it would blow up the project scope too much, but it would nevertheless be interesting to investigate in future work. Finally, it would be interesting to see these methods applied to a fleet of fully electric vehicles, such as battery electric vehicles, instead of the PHEV fleet used in this project. This would negate any potential issues that occur as a result of usage of the internal combustion engine.

As discussed in Section 6.2.3, some of the features were very resistant to changes brought on by the sample weighting. It is not clear why this was the case, but it may suggest that these features have a deeper connection to the ageing process. Examples of these features were the *Median, SOC & Marginalized Temperature*; *Median, RMS current*; *10th percentile, SOC & Marginalized Temperature*; and *Estimated Age*. Knowing what factors typically affect the battery ageing process, it can be thought that it is no coincidence these particular features were stable, as it would make sense for them to be correlated to the degradation rate. This could perhaps be examined further.

Other methods for focusing the models' attention on outlier data, aside from sample-weighting, could also be interesting to investigate. For example, the dataset could be re-sampled in a way that prefers less common data points. This would additionally remove any potential influence that DenseWeight and its hyperparameter choices has on the results.

The method for aggregating histogram features into numerical features was picked early on in this project, without too much consideration. It would be interesting to further investigate how this choice affected the model performance and interpretations, and to explore others methods of performing these aggregations.

Bibliography

[1] IEA, "Global ev outlook 2021," IEA, Report, 2021. [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2021.

[2] A. Barré, B. Deguilhem, S. Grolleau, M. Gérard, F. Suard, and D. Riu, "A review on lithium-ion battery ageing mechanisms and estimations for automotive applications," *Journal of Power Sources*, vol. 241, pp. 680–689, 2013, ISSN: 0378-7753. DOI: 10.1016/j.jpowsour.2013.05.040. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378775313008185.

[3] C. Molnar, *Interpretable machine learning*, Electronic Book, 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book.

[4] Volvo Car Group, *Volvo cars to be fully electric by 2030*, Press Release, 2021. [Online]. Available: https://www.media.volvocars.com/global/en-gb/media/pressreleases/277409/volvo-cars-to-be-fully-electric-by-2030 (visited on 04/27/2022).

[5] P. Chanchaipol and L. Sirikul, "Ev battery degradation forecasting based on high dimensional data," Thesis, 2020. [Online]. Available: https://hdl.handle.net/20.500.12380/302222.

[6] Y. Zhang, T. Wik, J. Bergström, M. Pecht, and C. Zou, "A machine learning-based framework for online prediction of battery ageing trajectory and lifetime using histogram data," *Journal of Power Sources*, vol. 526, p. 231 110, 2022, ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2022.231110. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378775322001331.

[7] M. Armand and J. M. Tarascon, "Building better batteries," *Nature*, vol. 451, no. 7179, pp. 652–657, 2008, ISSN: 1476-4687. DOI: 10.1038/451652a. [Online]. Available: https://doi.org/10.1038/451652a.

[8] J. S. Edge, S. O'Kane, R. Prosser, *et al.*, "Lithium ion battery degradation: What you need to know," *Physical Chemistry Chemical Physics*, vol. 23, no. 14, pp. 8200–8221, 2021, ISSN: 1463-9076. DOI: 10.1039/D1CP00359C. [Online]. Available: http://dx.doi.org/10.1039/D1CP00359C.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, Electronic Book, 2016. [Online]. Available: http://www.deeplearningbook.org.

[10] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate Research*, vol. 30, no. 1, pp. 79–82, 2005, 10.3354/cr030079. [Online]. Available: https://www.int-res.com/abstracts/cr/v30/n1/p79-82/.

[11] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*, Electronic Book, 2009. [Online]. Available: https://hastie.su.domains/ElemStatLearn/ (visited on 05/10/2022).

[12] C. Li, *A gentle introduction to gradient boosting*, Generic, 2014. [Online]. Available: http://www.chengli.io/tutorials/gradient_boosting.pdf.

[13] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020, ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2019.12.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253519308103.

[14] L. S. Shapley, *Notes on the N-Person Game - II: The Value of an N-Person Game*. Santa Monica, CA: RAND Corporation, 1951. DOI: 10.7249/RM0670. [Online]. Available: https://www.rand.org/pubs/research_memoranda/RM0670.html.

[15] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[16] S. M. Lundberg, G. Erion, H. Chen, *et al.*, "From local explanations to global understanding with explainable ai for trees," *Nature Machine Intelligence*, vol. 2, no. 1, pp. 56–67, 2020, ISSN: 2522-5839. DOI: 10.1038/s42256-019-0138-9. [Online]. Available: https://doi.org/10.1038/s42256-019-0138-9.

[17] M. Steininger, K. Kobs, P. Davidson, A. Krause, and A. Hotho, "Density-based weighting for imbalanced regression," *Machine Learning*, vol. 110, no. 8, pp. 2187–2211, 2021, ISSN: 1573-0565. DOI: 10.1007/s10994-021-06023-5. [Online]. Available: https://doi.org/10.1007/s10994-021-06023-5.

[18] V. A. Epanechnikov, "Non-parametric estimation of a multivariate probability density," *Theory of Probability & Its Applications*, vol. 14, no. 1, pp. 153–158, 1969, ISSN: 0040-585X. DOI: 10.1137/1114019. [Online]. Available: https://doi.org/10.1137/1114019.

[19] J. S. Milton and J. C. Arnold, *Introduction to probability and statistics*, 4th ed. McGraw-Hill, 2003, ISBN: 978-0-071-19859-2.

[20] B. Bole, C. S. Kulkarni, and M. Daigle, *Randomized battery usage data set*, Dataset, 2014. [Online]. Available: http://ti.arc.nasa.gov/project/prognostic-data-repository.

[21] S.-M. Udrescu and M. Tegmark, "Ai feynman: A physics-inspired method for symbolic regression," *Science Advances*, vol. 6, no. 16, eaay2631, 2019, doi: 10.1126/sciadv.aay2631. DOI: 10.1126/sciadv.aay2631. [Online]. Available: https://doi.org/10.1126/sciadv.aay2631.

[22] The Pandas development team, *Pandas documentation*, Web Page, 2021. [Online]. Available: https://pandas.pydata.org/pandas-docs/version/1.2/index.html.

[23] The NumPy Community, *Numpy v1.21 manual*, Web Page, 2021. [Online]. Available: https://numpy.org/doc/1.21/index.html.

[24] The SciPy Community, *Scipy documentation*, Web Page, 2022. [Online]. Available: https://docs.scipy.org/doc/scipy-1.8.0/index.html.

[25] XGBoost developers, *Xgboost documentation*, Web Page, 2021. [Online]. Available: https://xgboost.readthedocs.io/en/stable/.

[26] scikit-learn, *Scikit-learn: Machine learning in python*, Web Page, 2021. [Online]. Available: https://scikit-learn.org/1.0/.

[27] J. Yang, "Fast treeshap: Accelerating shap value computation for trees," *arXiv preprint arXiv:2109.09847*, 2021. DOI: 10.48550/arXiv.2109.09847. [Online]. Available: https://arxiv.org/abs/2109.09847.

A | Complete set of features in the Vehicle Fleet Dataset

Table A.1 shows a complete list of features in the Vehicle Fleet Dataset after aggregation of the complex data signals. Note that this is a list of intermediate features, which includes constant features that are not in the final dataset.

Table A.1: The complete set of intermediate features, and the base features they were computed from, in the Vehicle Fleet Dataset.

Base feature	Intermediate feature
Global time	Global time
Odometer value	Odometer value
Estimated age	Estimated age
Accumulated charge throughput at different total times	Minimum, Accumulated charge throughput
	Maximum, Accumulated charge throughput
	Median, Accumulated charge throughput
	Sum, Accumulated charge throughput
	Value mean, Accumulated charge throughput
	10th percentile, Accumulated charge throughput
	25th percentile, Accumulated charge throughput
	50th percentile, Accumulated charge throughput
	75th percentile, Accumulated charge throughput
	90th percentile, Accumulated charge throughput
Running time spent in RMS current region	Minimum, RMS current
	Maximum, RMS current
	Median, RMS current
	Sum, RMS current
	Value mean, RMS current
	10th percentile, RMS current
	25th percentile, RMS current
	50th percentile, RMS current
	75th percentile, RMS current
	90th percentile, RMS current
Parked time spent in SOC & Marginalized Temperature regions	Minimum, SOC & Marginalized Temperature
	Maximum, SOC & Marginalized Temperature
	Median, SOC & Marginalized Temperature
	Sum, SOC & Marginalized Temperature
	Value mean, SOC & Marginalized Temperature

Parked time spent in Marginalized SOC & Temperature regions	10th percentile, SOC & Marginalized Temperature
	25th percentile, SOC & Marginalized Temperature
	50th percentile, SOC & Marginalized Temperature
	75th percentile, SOC & Marginalized Temperature
	90th percentile, SOC & Marginalized Temperature
	Minimum, Marginalized SOC & Temperature
	Maximum, Marginalized SOC & Temperature
	Median, Marginalized SOC & Temperature
	Sum, Marginalized SOC & Temperature
	Value mean, Marginalized SOC & Temperature
	10th percentile, Marginalized SOC & Temperature
	25th percentile, Marginalized SOC & Temperature
	50th percentile, Marginalized SOC & Temperature
	75th percentile, Marginalized SOC & Temperature
	90th percentile, Marginalized SOC & Temperature

B | Implementation details

Data processing and model training was done on shared cloud compute nodes with access to at most 16 CPU cores and 64 GB of memory. The true resources usage varied, as a result of the virtual environment which dealt with resource allocation automatically. However, a minimum of 8 CPU cores and 32 GB of memory were always allocated.

Python 3.10 was used as the primary implementation language for all parts of the project. The data pre-processing pipelines were mainly implemented using Pandas 1.2 [22], NumPy 1.21 [23] and SciPy 1.8 [24], and also saw heavy use of the standard python multiprocessing library to utilize parallelism for parts of the pipelines. The gradient boosting models were implemented using XGBoost 1.6 [25] and Scikit-learn 1.0 [26]. In addition to these, a number of other packages were used to persist trained models and store data.

SHAP was implemented using FastTreeSHAP 0.1.2 [27], an optimized implementation of the original SHAP package. Some manual edits were made to the source code to fix and modify some visualisations, but these did not affect the underlying algorithms.

B.1 DenseWeight

The original DenseWeight paper by M. Steinberger et al. comes paired with the authors' own implementation of the algorithm [17]. Due to limitations in a package dependency however, this implementation does not support high dimensional data. For this reason, a custom implementation was developed and used instead. The major difference between the authors' original implementation and this custom implementation of the algorithm was the use of the scikit-learn implementation of KDE. This implementation uses a different estimation method compared to the what is used in the authors' original algorithm. All other differences were superficial.

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY