MASTER'S THESIS 2021

Predicting Lithium-Ion Battery Cell Quality Indicators

Filip Vitéz

ISSN 1650-2884
LU-CS-EX: 2021-24

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2021-24

Predicting Lithium-Ion Battery Cell Quality Indicators

Prediktion av Litiumjonbattericellskvalitet

Filip Vitéz

Predicting Lithium-Ion Battery Cell Quality Indicators

(Using production line data and machine learning to predict battery cell quality indicators at the end of the production line)

Filip Vitéz
bas15fvi@student.lu.se

June 14, 2021

Master's thesis work carried out at Northvolt AB.

Supervisors: Marcus Ulmefors, marcus.ulmefors@northvolt.com
Kenan Šehić, kenan.sehic@cs.lth.se

Examiner: Elin Anna Topp, elin_anna.topp@cs.lth.se

Abstract

Lithium-ion batteries are becoming ubiquitous in a society transitioning from fossil fuels to clean energy. To meet the demand, an explosive increase of production is required. Ramping up production while retaining cell quality is difficult due to a complex production chain. In the present study, we design and evaluate a machine learning pipeline for predicting three battery cell quality indicators using data collected from a battery cell production line. Further, we apply an exhaustive feature importance analysis on the trained models to gain insights regarding what production parameters are most important for predicting the quality indicators. For one of the predicted quality indicators, the best model achieves a root mean squared percentage error of 0.448%. For the same quality indicator, we find a production parameter as a potential bottleneck for achieving the desired performance. For all targets, the models struggle to explain the variance within the target distributions.

Keywords: Lithium-ion, predictive quality, latent space, ensemble methods, Gaussian processes, Catboost, UMAP, Shapley values, SHAP

Acknowledgements

I would like to thank...

My LTH supervisor Kenan Šehić for contributing to fruitful discussions regarding the scope and execution of the project. Also for having patience when access to the data we were supposed to use in the project was delayed.

My Northvolt supervisor Marcus Ulmefors for entrusting me with this project. You are a great engineer and always have interesting insights to provide regardless of engineering discipline.

My Northvolt colleagues Lef Filippakis and Lage Ragnarsson for excellent feedback on the final report, as well for having the patience of listening to me complain about problems relating to the dataset.

Contents

Acronyms

CA cell assembly. 10, 29, 53, 54

CER cause and effect relationship. 10, 17

DCIR direct current internal resistance. 16, 17

DOE design of experiments. 17

EOL end of production line. 9–11, 13, 27, 53

F&A formation and ageing. 9, 10, 16, 29, 53

FMEA failure mode and effects analysis. 17

KPI key performance indicator. 10, 17

Li-ion lithium-ion. 9, 14, 29

LIME local interpretable model-agnostic explanations. 28

MAPE mean absolute percentage error. 36, 43, 44

ML machine learning. 19

OCV open circuit voltage. 16, 17

RFECV recursive feature elimination with cross validation. 31

RMSPE root mean squared percentage error. 36, 43

RSCV randomized search with cross validation. 34

SHAP SHapley Additive exPlanations. 27, 34–36

SOC state of charge. 17

SOH state of health. 10, 11, 29

t-SNE t-distributed stochastic neighbor embedding. 32, 33

UMAP uniform manifold approximation and projection. 33

Chapter 1
Introduction

With the energy sector shifting from fossil fuels to clean energy, the demand for electrical energy storage has never been greater. In 2017, the global battery market size was evaluated to $62 billion almost doubling to $120 billion in 2019 [39]. Currently, the battery type that is most established at scale is the lithium-ion (Li-ion) battery [38]. Li-ion batteries can be found in a wide range of products ranging from consumer electronics to electric vehicles, and are alone expected to reach a market value of $53.8 billion by 2024 [39].

Northvolt tries to meet the increasing Li-ion cell demand while retaining high quality in their products. However, battery cell manufacturing is difficult. Successful manufacturing of high quality cells requires tight tolerances and high repeatability [15]. Quality is traditionally validated via extensive testing in a process called formation and ageing (F&A) at the end of production line (EOL). During the process, the cells undergo controlled thermal and electrical processes to gain the necessary properties to become a finished product. Simultaneously, quality indicators are measured to ensure that the battery cells are conforming to predefined standards. Understanding how these quality indicators correlate with the production process would be beneficial for the company in multiple ways. First of all, one could find and highlight outlier cells early in the process for rework or removal off the line. Secondly, one could use the insights of what drives quality to alter the production process and in turn increase the yield of high quality cells. A third potential gain is related to a concept called performance validation. Here, a sample of the cells from a batch is selected for additional performance testing. The process is done to validate that the cells will retain quality for long after finished production. Optimally selecting how many and which cells to sample is a complicated process. Gaining further knowledge of the production line could enable the selection of smaller sample sizes, without losing knowledge of the entire cell population.

Thus, this thesis investigates a machine learning approach for predicting EOL quality indicators using data collected from the production line. Further, an exhaustive feature importance analysis is done on the trained models to find out what production parameters are most important for predicting the quality indicators.

1.1 Research question

The research undertaken in the thesis can be summarized in the following questions.

1. Can data collected from a battery cell production line be used as input to a regression model to predict EOL quality indicators?
2. What conclusions can be drawn with respect to the correlation between production line data and EOL quality indicators?

1.2 Limitations

As for any data-driven modelling problem, perhaps the most prominent limitation when it comes to model performance is the available data. Although major breakthroughs have been made the last decades in the area of machine learning, no model will perform well without having high quality data to learn from.

The dataset used in the thesis includes data from two subsections of the production line called cell assembly (CA) and formation and ageing (F&A). As we will see in Section 2, the production line contains additional areas from where we will not include data.

Another limitation is that we need to understand the reasoning behind the predictions produced by the models in order to be able to draw fruitful conclusions. Hence, black-box models are not of interest for this use case.

1.3 Related work

To the author's knowledge, the current literature does not cover the topic of using machine learning to estimate EOL battery cell quality indicators. However, in the closely related area of state of health (SOH) estimation, there is a multitude of studies to be found. In that area, successful approaches include Support vector machines [1], Gaussian process regression models [30] and neural network models [37]. In [29], the authors show the potential of random forest regressors with the help of feature engineering. Although there is no one-to-one mapping between EOL quality and SOH estimation, inspiration is drawn from studying the progress made in the SOH-area.

Another related field is that of overall quality assurance in a battery cell production line. The work presented in [6] examines variability in battery cell performance that comes from defects during electrode coating. A method for examining the electrolyte mixing process is given by [23]. Previous methods for quality assurance in battery cell production have in common that they examine the role of a single process. Only a few try to capture the effects of the entire production line.

Researchers at the BMW group propose a multivariate key performance indicator (KPI)-based method for cause and effect relationship (CER) identification in the production line [15]. Although their prediction targets are not the same as in this thesis, the data that they have used to drive the modelling is similar to the data used in this thesis. They show that by using their data-driven model they could reduce the scrap rate (the rate of battery cells that are discarded) by 3%. This clearly shows the potential of data-driven approaches.

1.4 Contribution

Combining machine learning and battery science has become increasingly popular in the last decade. However, research has mainly been done in the area of estimating the state of health of batteries. The main contribution of this thesis is to apply machine learning methods that have been proven successful in the area of SOH-estimation, to the novel area of EOL quality indicator prediction.

1.5 Outline

The thesis is divided into six main chapters. Chapter 2 covers a theoretical introduction to battery cells and battery cell manufacturing. Chapter 3 describes the needed theoretical framework. Chapter 4 presents the datasets used and the approach followed in the thesis. Chapter 5 examines the validity of our methods by applying them on a synthetic dataset. Then, the results on the Northvolt dataset are presented and discussed. Finally, chapter 6 summarizes the findings of the thesis and discusses further work.

Chapter 2
Battery cells & Northvolt production

In the following section, an introduction to lithium-ion cells is given from an electrochemical point of view. Then a walk-through of battery cell production at Northvolt is given, along with an explanation of the EOL quality indicators that we aim to predict in the thesis. Finally an overview of traditional methods for quality assurance in a production line is presented.

2.1 The lithium-ion battery cell

Batteries have become a central part of modern-day society, where most people have come across a battery in some shape or form. A first necessary distinction to make is that between a battery cell and a battery pack. A *battery cell* is a single electrochemical unit whereas a *battery pack* is a collection of battery cells. In the pack, cells are connected in series and/or in parallel depending on desired voltage, power and capacity capabilities of the battery. Already, a few terms have been introduced that may not make much sense to readers unfamiliar with electrochemistry. Hence, the following table with explanations of the electrochemical concepts discussed in the thesis is provided:

Table 2.1: Descriptions of electrochemical concepts [17].

Term	Meaning
Capacity (Ah)	The quantity of charge that a battery (cell) contains.
Current (A)	Rate of charge. Time derivative of capacity.
Power (W)	Rate per unit time of energy transfer, for example in an electrical circuit.
Voltage (V)	Difference in electric potential between two points.

A lithium-ion battery cell, illustrated in Figure 2.1, consists of negative and positive electrodes, electrolyte and a separator. The electrodes exchange ions during the charge/discharge of a cell. The electrolyte is an ionic conductor that serves as the medium for the ions to travel

through between the electrodes. The separator is a permeable membrane with holes to let lithium ions through. The holes need to be large enough for the ions to move through unimpeded, but small enough to make sure the negative and positive electrode particles do not touch. The latter would lead to a short circuit that would destroy the cell.

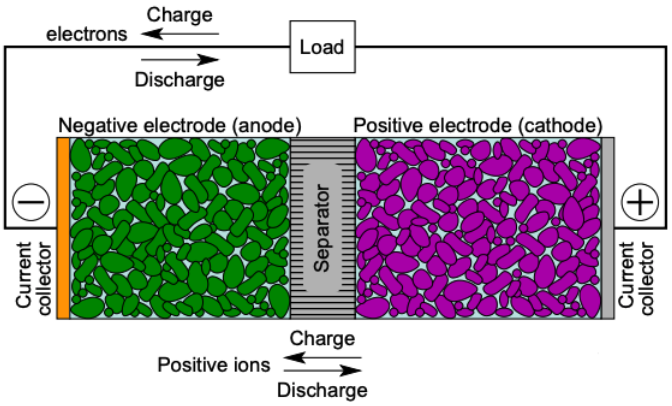


Figure 2.1: Model of lithium-ion battery cell. Positive (lithium) ions travel back and forth between the electrodes during the charge/discharge of the cell. [17]

During discharge, lithium ions travel through the electrolyte from the negative electrode to the positive. Compensating electrons are then passed through the outer circuit to balance the reaction inducing a current. The lithium ions enter the electrode particles without changing the electrode structure in a process called intercalation [26]. The process requires the electrodes to have open structures to enable the intercalation of ions, and the possibility of accepting electrons to balance the charge. The intercalation mechanism is completely reversible and much gentler than the electrochemical reaction that takes place in original battery cells, such as the lead-acid type used in car batteries. This makes lithium-ion cells more durable. One important property of the electrodes of the lithium-ion battery cells is that they are more similar to a composition of millions of small particles rather than something homogeneous. This enables reactions over a larger surface area which both decreases cell resistance and enhances power delivery capability [26].

The negative electrode of the lithium-ion battery cell is often some form of graphite [17]. Graphite consists of stacked layers of graphene between which lithium ions can intercalate. The positive electrode offers more variability in terms of its constituent parts. Different constitutions of cobalt, nickel, manganese and aluminium are common depending on the requested properties. The electrolyte of the lithium-ion battery cell must be non-aqueous due to the violent reactions that arise when mixing water and lithium.

2.2 Battery cell production at Northvolt

The most common form factors of Li-ion battery cells are cylindrical, prismatic, button, coin and pouch cells. Northvolt produces cylindrical and prismatic cells, which are illustrated in Figure 2.2. The production processes and the final product properties vary a lot between the cell form factors.

Figure 2.2: Illustration of prismatic (left) and cylindical (right) battery cells produced at Northvolt.

The production line of battery cells can be divided into subsections. In each such subsection, various process steps are undergone where measurements are taken to ensure that the production is going according to plan. The purpose of each subsection is explained below.

2.2.1 Upstream

In the upstream part of the production line the active materials that go into the batteries are prepared. The active materials of each electrode are mixed into a slurry. If contamination between the positive and negative electrode slurry materials were to happen, the resulting cells would become useless. Hence, it is of crucial importance that these processes are separated. This is usually ensured by mixing the slurries in different parts of the factory.

2.2.2 Downstream

Electrode coating

Here, the active material for the different electrodes is layered on top of metal foil. The foil in between the layering is the current collector that conducts the current in the cell. To ensure that the negative and positive electrodes are balanced, the thickness of the coating is chosen accordingly. An illustration of the layering is shown in Figure 2.3.

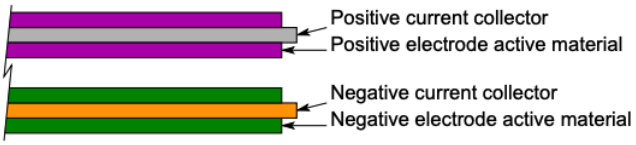


Figure 2.3: Illustration of coated foil [17].

The coated foil is then dried in a long drying oven, and then rolled up into a big roll. The coated rolls are then unwound and inserted into a pressing machine where the active materials are compressed to optimize the spaces between particles.

Cell assembly

As the name suggests, this part of the process is where the cell is assembled from its constituent parts. The process consists of many sub-processes that differ depending on the type

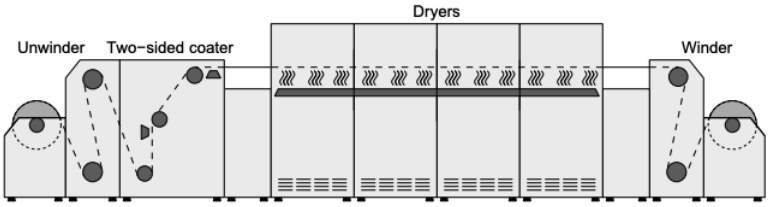


Figure 2.4: Overview of coating & drying process [17].

of cell that is manufactured.

Similar for both cylindrical and prismatic cells is that the positive and negative electrode foils are cut into electrode sheets that are then either rolled or stacked in a zig-zag fashion, kept apart by the separator. The negative and positive tabs are then welded in parallel to their respective terminals (negative/positive). The socket or cassette is then put into a can which is welded together with a corresponding lid. One small opening is left in the lid to allow for the filling of electrolyte.

The electrode and cell assembly parts of the production line must be kept in a dry room. Humidity is kept close to zero since the electrolyte is so highly reactive with water. If moisture is present in the air, the electrolyte decomposes resulting in the emission of toxic gases.

Formation and ageing

Formation and ageing is done both as a part of the actual manufacturing process, and as a part of quality validation to make sure the cell performance is conforming to predefined standards.

During the ageing process a cell is kept at specific temperatures for time intervals ranging between hours up to weeks. During the formation process, the cell is charged/discharged in intervals creating a solid electrolyte interphase on the negative electrode. This serves as a passivating layer to protect the negative electrode from further reactions [17].

Some parts of the F&A process are solely done as quality validation. After the multiple ageing processes are completed, three quality indicators are measured that are of great importance for the cell's performance; discharge capacity, direct current internal resistance (DCIR), and open circuit voltage (OCV).

Discharge capacity

Battery capacity represents the maximum amount of charge that can be extracted from the cell under specified conditions, and is hence arguably the most important quality indicator. A common method to measure the capacity of a cell is to sum the current during a full discharge session. It is calculated as [29]:

$$q = \int_{t_0}^{t_{end}} I(t)\, dt, \tag{2.1}$$

where t_0 and t_{end} are the start/end times of the discharge session, and I the current.

Direct current internal resistance

Direct current internal resistance (DCIR) is, as the name suggests, the resistance that the current faces when traveling through the battery cell. It is a strong indicator of a battery cell's power characteristics [13]. A higher internal resistance is also correlated with thermal instability since more of the charging energy is converted into heat. DCIR measurements are physically done by injecting pulses of current into the electrodes of the cell, while monitoring the changes of cell voltage and current. The DCIR is then calculated as the ratio of voltage variation and current variation:

$$DCIR = \frac{\Delta V}{\Delta I}, \tag{2.2}$$

where V is the voltage and I is the current.

Open circuit voltage drop over time

Open circuit voltage (OCV) is the voltage between battery terminals when no load is applied. There is a strong correlation between the OCV and the state of charge (SOC) of a battery cell [5]. OCV is measured multiple times during formation and ageing. If the OCV drops significantly from one measurement to another, we have an indicator that there are internal defects in the cell.

2.3 Traditional methods for production line quality assurance

Monitoring a production line has been of interest since the start of the industrial revolution. There are many existing tools for quality management and assurance at hand today. In the following section, two common methods for quality assurance in a production line are explained briefly.

i) Failure mode and effects analysis (FMEA) uses a systematic experience-based recording of failures, risks and consequences in order to initiate preventive measures. This method works in smaller settings, but struggles for larger projects where there is a high number of cause and effect relationships (CERs) [12].

ii) Design of experiments (DOE) is a data-driven approach for finding and quantifying CERs between production line data and key performance indicators (KPIs). The method requires a lot of time and resources to produce interesting results. Also, system behaviour can only be analysed when factors are experimentally prescribed [15]. This means the method can only strengthen or falsify a hypothesis rather than finding new connections.

To summarize, the current available methods for quality assurance in a production line face the following problems. Either they are designed solely for monitoring individual processes, failing to capture the complexity of the entire process chain. Or, they depend too heavily on human expert input making them impractical for large scale production [15]. Hence, this thesis proposes a novel approach using machine learning.

Chapter 3

Theory

In the following chapter we provide the theoretical framework. First, brief introductions to machine learning and regression are given. We then describe the regression models used in the thesis. We conclude the chapter by explaining the necessary theory relating to feature importance and model evaluation.

3.1 Machine learning

Machine learning (ML) is a subarea of computer science where the objective is to create algorithms that can learn to make decisions and predictions based on data rather than explicit programming. There are three main areas within machine learning: supervised, unsupervised and reinforcement learning. In supervised learning, the dataset consists of input-output pairs, where the goal of the model is to find a function that accurately maps an input to its corresponding output (ground truth). In unsupervised learning, the algorithm tries to find similarities in the data without ground truth labels to guide the learning. Here the algorithms need to learn by finding patterns in the training data without explicitly knowing if these patterns are relevant or not. In reinforcement learning, the algorithms teach themselves how to maximize the reward given an interactive environment and reward function.

There are different ML models suitable for different problem types. In some cases we are only interested in a model's prediction/classification, rather than in understanding why/how the model came to the conclusion that it did. However, more often than not, the model's reasoning is of crucial importance for its prediction to become truly useful. Model interpretability has hence become a major subject within machine learning [18].

3.2 Regression

The objective of regression is to determine the relationship between independent and dependent variables. It attempts to find a function that can accurately output a target value given a set of input values. This can mathematically be described as:

$$y = f(\mathbf{x}), \tag{3.1}$$

where, $\mathbf{x} \in \mathbb{R}^D$ is the set of D-th dimensional input variables. y is the target value and can be either single or multi-dimensional. f represents the mapping/relationship between input and output values. A simple assumption is that the relationship between $\mathbf{x}$ and y is linear, yielding the following mapping function:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathbf{b}. \tag{3.2}$$

Here $\mathbf{w}$ represents the weight/importance corresponding to each feature, and $\mathbf{b}$ is the the bias that quantifies the simplifying assumptions made by the model to make the target function easier to approximate. Many real life problems are not linear, and thus require more complicated mapping functions f. The models used in the thesis are described in the below section (3.3).

Once the mapping function is chosen, the regression model compares the function output to the ground truth values to determine the function parameters (w and b in the linear model (3.2)). This comparison, represented as a loss function, can be done in many ways. Common for regression problems is to use some variant of least squares approximation:

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{3.3}$$

where y_i is the ground truth value and $\hat{y}_i$ is the function output for each input sample $\mathbf{x}_i$, and n the number of training samples. The objective of the model is to choose its mapping function parameters in a way that minimizes the loss function L.

3.3 Regression Models

There are a plethora of regression models to choose from when solving a regression problem. Finding the optimal model is problem dependent, and it is hence common to compare the results of multiple models. The models evaluated in this thesis are explained below.

3.3.1 Decision trees

A decision tree algorithm is often visualised as an upside down tree with the root at the top. The algorithm works by continuously splitting the data into separate branches based on the values of the input features, until a stopping criterion is reached or no more splits are possible. When that happens, the algorithm has reached a leaf node. If the algorithm works as expected, the data in a leaf node is similar enough for the algorithm to be able to make a decision/prediction for all samples within that node.

A functioning decision tree algorithm effectively finds the best features (and their values) to use for splitting the data down to the next layer. Finding the optimal splitting attributes can be done in multiple ways. Perhaps the two most common methods are to minimize Gini impurity or to maximize information gain. Gini impurity measures how often a sampled element from the data set would be incorrectly labeled if it was labeled according to the majority class within that set. Consequently, the algorithm strives to split the data in a way that minimizes the Gini impurity for resulting splits [28]. For a set S with K classes, it is calculated as follows:

$$I_G(S) = \sum_{i=1}^{K} p_i \sum_{k \neq i} p_k, \tag{3.4}$$

where p_i is the probability of drawing a sample with label i, $\sum p_k = 1 - p_i$ is the probability of miss-classifying that sample.

Information gain measures how much we can say of a random variable, by observing the outcome of another. To understand how information gain is used for deciding splitting attributes, we first need to get a grasp of information entropy. Information entropy can be thought of as variance. If a set contains only one type of object, the entropy of that set is zero. Entropy is given by [28]:

$$H = -\sum_{i}^{K} p_i \log_2 p_i, \tag{3.5}$$

where the other variables are defined as previously. In Figure 3.1 we see a data set with two categorical classes (blue and green), and two continuous features (x and y). A split has been made at $x = 1.5$ that divides the data into two subsets.

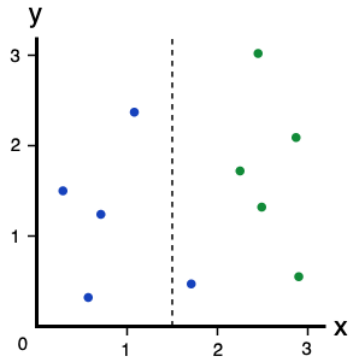


Figure 3.1: Visualization of an imperfect split.

The information gain tells us how much a given split has reduced the entropy for the entire data set. If we denote the entropy before the split as H_b, and the entropies of the right subset and left subsets of the data as H_r and H_l respectively, this gives us:

$$\begin{aligned} IG = H_b - (w_r H_r + w_e H_l), \\ w_r + w_e = 1, \end{aligned} \tag{3.6}$$

where w_r and w_e are weights determined by the fraction of elements within each subgroup. The decision tree tries to find the splitting attribute (and value) that yields the highest possible information gain.

The main disadvantage of decision trees is their proneness to overfitting that arises from the decision function's lack of smoothness. A small change in one of the input features can have large consequences on the predicted outcome.

3.3.2 Ensemble methods

The principle behind ensemble methods is to combine several weaker models to gain greater predictive power and more robustness. There are multitude of ensemble methods available, where the most common ones include bootstrap aggregating (bagging) and boosting.

Bagging

Bootstrap aggregating or bagging creates artificial samples of the original data by using the bootstrap method. The samples, usually the same size as the original dataset, are drawn uniformly from the original dataset with replacement. This means that the same data point can reside multiple times within the same sample. Each sample is then assigned to a model for training. The models are then aggregated into a final model by averaging the results.

Random forest

The random forest algorithm combines multiple decision trees using bagging together with the random subspace method. With the inclusion of the random subspace method, we also sample the feature space for a subset of the features to use for training each individual tree. Together with bagging, this forces variation amongst the trees in the forest yielding lower correlation between their predictions. A more thorough explanation of the algorithm is given by [4].

To estimate the standard deviation of the predictions made by a random forest regressor, one can make use of a version the jackknife method as proposed by [35]. The idea of the jackknife method is to omit one sample from the dataset and then compute a model's prediction using the remaining samples. This is done for all samples of the dataset, and the standard deviation can be obtained from the set of predictions.

Boosting

Boosting is a weighted ensemble learning method where individual models are combined in an iterative way. In each learning step additional weight/importance is given to the data samples that the previous model struggled the most with [36]. For a combination of two decision trees with MSE as loss function, this is mathematically described as:

$$\begin{aligned} f_1(x) &= \arg\min_f \frac{1}{n}\sum_{i=1}^{n}(f(x_i)-y_i)^2\,, \\ f_2(x) &= \arg\min_f \frac{1}{n}\sum_{i=1}^{n}(f_1(x_i)+f(x_i)-y_i)^2\,. \end{aligned} \tag{3.7}$$

Here f_1, f_2 are the first and second decision trees, f the space of decision tree regressors, n the number of training samples and x_i, y_i the i:th input/output pair. Since we focus our

learning based on a subset of the training data, we have a higher risk of overfitting than for bagging methods.

Gradient boosting

The difference between gradient boosting and vanilla boosting lies in the manner in which the algorithms identify the shortcomings of the previous models. Gradient boosting looks at the gradient of the loss function, rather than at the data samples and their corresponding weights [7].

CatBoost

Catboost is a variant of the traditional gradient boosting algorithm that was first introduced in 2017 [25]. The key advantages of Catboost are an innovative way of dealing with categorical features [25], and a mechanism called ordered boosting. To understand ordered boosting we first need to examine the concept of prediction shift. In traditional boosting algorithms, the gradients at each step are calculated from the same target values that the current model is trained on. This can lead to a prediction shift since the distribution of estimated gradients within a subdomain of the feature space is shifted compared to the true distribution of gradients in this domain. The authors propose the following version of gradient boosting to solve the problem:

Algorithm 1: Ordered boosting [25]

Data: Your dataset with n samples
Result: Ensemble model trained using ordered boosting
initialization;
for *log(n) iterations* **do**
  1. calculate errors (and gradients) for each data point with a model that has been trained on all data points except for that one,
  2. train a model by using the residuals of each data point as class values
end

The $\log n$ number of iterations is chosen to make the algorithm computationally viable for larger datasets. The gradients are calculated for each point with a model that has not previously seen that data point, alleviating the problem of prediction shift.

3.3.3 Gaussian process regression

Bayesian approach

Gaussian process regression uses the Bayesian approach for function approximation. The Bayesian approach (contrary to the frequentist) does not only take into account the observed data, but also prior knowledge of the situation at hand. One example could be the task of predicting the gender distribution in a company, based on a subset of the workers height. The frequentist approach would base its prediction solely on the data at hand. This might in some situations be problematic, especially if the sample size is small compared to the entire

data set. It is not clear that the drawn sample gives an accurate representation of the entire population. The Bayesian approach could here incorporate a prior gender distribution equal to that of the country or work industry and is thus better equipped to handle the uncertain nature of only having a subset of the data at hand.

We start by using the Bayesian approach to estimate function parameters for a standard linear model with Gaussian noise:

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{x}^T \boldsymbol{w}, \\ y &= f(\boldsymbol{x}) + \epsilon, \\ \epsilon &\sim \mathcal{N}(0, \sigma_n^2). \end{aligned} \tag{3.8}$$

Here, $\boldsymbol{x}$ and $\boldsymbol{w}$ are the vectors of inputs and weights respectively. f is the function value and y the observed target value, differing from the function output due to independent, identically distributed Gaussian noise with zero mean and variance σ_n^2.

We can now construct a likelihood function which gives the probability density of our observations given the weights. We assume independent samples, and can hence factor over the samples in the training set giving:

$$p(\boldsymbol{y} \mid X, \boldsymbol{w}) = \prod_{i=1}^{n} p(y_i \mid \mathrm{x}_i, \boldsymbol{w}) = \mathcal{N}(X^T \boldsymbol{w}, \sigma_n^2 I), \tag{3.9}$$

where $\boldsymbol{X}$ is simply the matrix of inputs where each row represents a sample and each column represents a feature, and $\boldsymbol{n}$ the number of samples.

The Bayesian approach requires us to specify a prior over the weights. We can here, as earlier explained, incorporate domain knowledge to guide the fitting of the function. In this case we assign a zero mean Gaussian prior:

$$\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p), \tag{3.10}$$

where Σ_p is the covariance matrix.

We then adapt the weight distribution based on the observed data using Bayes' rule [32]:

$$p(w \mid \boldsymbol{y}, X) = \frac{p(\boldsymbol{y} \mid X, \boldsymbol{w}) p(\boldsymbol{w})}{p(\boldsymbol{y} \mid X)}. \tag{3.11}$$

Here p represents a probability distribution, where $p(\boldsymbol{w})$ the prior. $p(\boldsymbol{y} \mid X)$ is a normalizing constant called the marginal likelihood. It is calculated as the integral over the likelihood times the prior:

$$p(\boldsymbol{y} \mid X) = \int p(\boldsymbol{y} \mid X, \boldsymbol{w}) p(\boldsymbol{w}) d\boldsymbol{w}. \tag{3.12}$$

The other variables are defined as previously. The resulting distribution that incorporates both prior knowledge, and likelihood given the observed data, is called the *posterior distribution*.

Once the posterior is obtained, we can predict values for new, unseen data (x_*) from the predictive distribution for f_* [32]:

$$p(\boldsymbol{f}_* \mid \boldsymbol{x}_*, X, \boldsymbol{y}) = \int_w p(\boldsymbol{f}_* \mid \boldsymbol{x}_*, \boldsymbol{w}) p(\boldsymbol{w} \mid X, \boldsymbol{y}) d\boldsymbol{w}. \tag{3.13}$$

Gaussian process regression

In Gaussian process regression, our objective is to describe a distribution over functions rather than over weights to a specific function.

Definition 3.3.1. *A Gaussian process is a collection of random variables, such that any finite number of which have a joint Gaussian distribution [32].*

Here, the random variables represent the values of our function $f(\boldsymbol{x})$. Mathematically, a Gaussian process is given by [32]:

$$f(\boldsymbol{x}) \sim GP(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')), \tag{3.14}$$

where $\boldsymbol{m}$ and $\boldsymbol{k}$ are the mean and covariance functions, given by:

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbb{E}[f(\boldsymbol{x})], \\ k(\boldsymbol{x}, \boldsymbol{x}') &= \mathbb{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}') - m(\boldsymbol{x}'))]. \end{aligned} \tag{3.15}$$

The covariance function/kernel encodes how the random variables are correlated to each other and thus dictates the structure of the response function that we can fit. For a real valued feature space, a state of the art family of shift invariant kernels for Gaussian process regression is the Matérn family [3]. The mean function represents the bias. The format of the mean and covariance functions of the Gaussian process represent the prior knowledge we can specify depending on problem domain.

Just as in the linear example, we assume independent identically distributed Gaussian noise yielding:

$$\begin{aligned} \boldsymbol{y} &\sim GP(m(\boldsymbol{x}, k(\boldsymbol{x}, \boldsymbol{x}') + \delta_{ij}\sigma_n^2), \\ \boldsymbol{y} &= f(\boldsymbol{x}) + \epsilon, \\ \epsilon &\sim \mathcal{N}(0, \sigma_n^2), \end{aligned} \tag{3.16}$$

where δ is the Kronecker delta:

$$\delta_{ij} = \begin{cases} 1, & i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{3.17}$$

The Gaussian process model provides an analytical expression for the marginal likelihood of the data, where the term *marginal* refers to the marginalization done over the function values $\boldsymbol{f}$. The expression for the log marginal likelihood is given by [32]:

$$\log p(\boldsymbol{y} \mid X) = \overbrace{-\frac{1}{2}\boldsymbol{y}^T(\boldsymbol{K} + \sigma_n^2 I)^{-1}\boldsymbol{y}}^{a} \underbrace{-\frac{1}{2}\log\left|\boldsymbol{K} + \sigma_n^2 I\right|}_{b} \overbrace{-\frac{n}{2}\log 2\pi}^{c}, \tag{3.18}$$

where a zero mean is assumed, and $\boldsymbol{K}$ is the covariance kernel matrix consisting of covariances evaluated at different samples $\boldsymbol{x}_i$. The first part (a) of Eq. (3.18) gives information regarding how well the model manages to fit the data. The second term (b) quantifies the model complexity. Naturally, we want a model with as low complexity as possible while retaining predictive power. The third part (c) is a term proportionate to the number of samples

within the dataset. We see that the likelihood decreases with larger datasets. The hyperparameters of the kernel function are optimized using the marginal likelihood [31].

Since we have a Gaussian prior, the collection of training and test points are joint multivariate Gaussian distributed [32]:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f_*} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right), \tag{3.19}$$

where X_* and $\boldsymbol{f}_*$ represent unseen input samples and function output for those samples respectively. To get the posterior distribution of functions, we condition the joint Gaussian prior distribution on the observations, yielding [32]:

$$\begin{aligned} \boldsymbol{f}_* \mid X, \boldsymbol{y}, X_* &\sim \mathcal{N}(\bar{\boldsymbol{f}}_*, \Sigma_*), \quad \text{where} \\ \bar{\boldsymbol{f}}_* &= \mathbb{E}[\boldsymbol{f}_* \mid X, \boldsymbol{y}, X_*] = K(X_*, X)[K(X,X) + \sigma_n^2 I]^{-1} y, \\ \Sigma_* &= K(X^*, X^*) - K(X^*, X)[K(X,X) + \sigma_n^2 I]^{-1} K(X, X^*). \end{aligned} \tag{3.20}$$

Our predictions are $\bar{\boldsymbol{f}}_*$, and we obtain the variances of each prediction from the diagonal of the covariance matrix Σ_*.

3.4 Feature importance analysis

Feature importance analysis revolves around finding out how the input features to a model has impacted its predictions. Feature contributions are trivial to calculate for the linear regression model:

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{w}^T \boldsymbol{x} \\ &= w_0 + w_1 x_1 + \ldots + w_p x_p. \end{aligned} \tag{3.21}$$

Here $\mathbf{x}$ is the sample for which we want to calculate the marginal contributions, and $\boldsymbol{w}$ the weight vector. We get the feature contribution ϕ_j to the prediction $f(\mathbf{x})$ by calculating the difference between the effect of the feature minus the average effect:

$$\begin{aligned} \phi_j(f) &= w_j x_j - \mathbb{E}(w_j \bar{x}_j) \\ &= w_j x_j - w_j \mathbb{E}(\bar{x}_j), \end{aligned} \tag{3.22}$$

where $\bar{x}_j$ is the average value of feature j and $\mathbb{E}(w_j \bar{x}_j)$ the estimate of the mean effect of feature j. For more complex models, where we do not have such explicit definitions of feature weights, we can make use of SHAPley values.

The Shapley value is a solution concept for distributing payout among players in a game. This can be used in feature importance analysis, where we model each feature as a player. The model prediction is the payout/outcome, and the objective is to distribute the prediction fairly among the features. The features cooperate in a coalition to gain a certain profit from the cooperation. The objective is to explain the difference between a specific prediction and the average prediction with respect to the possible coalitions. The Shapley value of a feature is the average marginal contribution of its value across all possible coalitions. The marginal contribution of a feature measures how much the inclusion of the feature changes the model prediction away from the average, given all other features kept stationary. This is done for each possible coalition and a weighted average is calculated. By estimating the Shapley values

for all feature values, we get the distribution of the prediction among the feature values. The Shapley value for one feature value is defined as follows:

$$\phi_j(v_{\mathbf{x}}) = \sum_{S \subseteq N \setminus \{x_j\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!} \left(v_{\mathbf{x}}\left(S \cup \{x_j\}\right) - v_{\mathbf{x}}(S) \right). \tag{3.23}$$

Here, S is a subset of the available features N. $\mathbf{x}$ is the the sample and $v_{\mathbf{x}}(S)$ is the payout for the subset S of features. The equation is far from intuitive at first glance. We can rewrite it to:

$$\phi_j(v_{\mathbf{x}}) = \frac{1}{|N|} \sum_{S \subseteq N \setminus \{x_j\}} \binom{|N| - 1}{|S|}^{-1} \left(v_{\mathbf{x}}\left(S \cup \{x_j\}\right) - v_{\mathbf{x}}(S) \right), \tag{3.24}$$

and examine it with an example:

We run a battery cell factory and want to understand how much 3 hypothetical production parameters, (A, B and C), affect an EOL quality indicator. There are multiple effects that impact the EOL quality indicator simultaneously. If we want to measure the contribution of production parameter A, we get:

$$\begin{aligned} N &= \{A, B, C\} \\ j &= A. \end{aligned} \tag{3.25}$$

(3.24) tells us to form all possible subsets, excluding our feature of interest. This gives us:

$$\{\varnothing\}, \{B\}, \{C\}, \{BC\}, \tag{3.26}$$

where $\varnothing$ is the empty set. Then, for each such subset, we want to calculate the marginal value of adding our feature of interest, A, to the game. In our example, we get 4 different marginal values.

The

$$\binom{|N| - 1}{|S|}^{-1} \tag{3.27}$$

part from (3.24) then tells us to divide each marginal value by the number of permutations possible for each subset size $|S|$. For $|S| = 1$, we can construct two subsets of that size ($\{B\}, \{C\}$) and thus we should scale the marginal value corresponding to subsets of that size by $\frac{1}{2}$. This is done to average out the effect that the other features have for each subset size. After summing the average marginal contributions together, we divide the resulting sum by the total number of features $|N|$. This is needed to average out the effect of the total feature group size. We want to see how much A contributes to the EOL quality indicator independent of the total number of features.

If we do this for all features, we can then fairly distribute the prediction among the features:

$$f(\mathbf{x}) = v_{\mathbf{x}}(\{A, B, C\}) = \phi_A(v_{\mathbf{x}}) + \phi_B(v_{\mathbf{x}}) + \phi_C(v_{\mathbf{x}}). \tag{3.28}$$

One major issue with the original implementation of Shapley value calculation is that it is computationally heavy. For a set of $|N|$ features, we get $2^{|N|}$ subsets for which to calculate average marginal contributions for. This quickly becomes unfeasible. Lundberg and Lee [18] propose the kernel-based method SHAP for estimating Shapley values, inspired by local surrogate models. A step-by-step approach for calculating SHAP-values is given in Section 4.7.2

Local surrogate models

Surrogate models work by trying to mimic the performance of trained black box models while retaining interpretability. There are global and local surrogate models. Local surrogate models focus on explaining specific predictions rather than trying to explain the global dependence between the model features and output. In [27], an implementation of local surrogate models, Local interpretable model-agnostic explanations (LIME), is proposed.

LIME works by studying the model in a specific region in the feature space in which we want to understand its predictions. A new dataset is created where feature values are sampled from the region of interest, and the labels are chosen as the model predictions for these sampled feature values. The samples are weighted according to their proximity to the midpoint x of the region of interest. An interpretable surrogate model, e.g. a decision tree regressor, is then trained on this newly created dataset. The original models prediction is explained based on interpreting the surrogate model. Mathematically this is given by:

$$g* = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g), \tag{3.29}$$

where $g*$ is the optimal surrogate model, g the model that minimizes the loss L, G is the space possible models (e.g the family of decision tree models), f the original black box model, and π_x is the kernel that gives the weights to the samples. The surrogate model complexity is given by Ω. Naturally, we want to keep the surrogate model as simple as possible. The objective is to achieve a model $g*$ that best mimics the original model f while retaining interpretability.

Chapter 4
Approach

In this chapter we begin by examining the datasets used in the thesis. We then continue by visualizing and explaining the steps of the proposed machine learning pipeline.

4.1 Data

Synthetic dataset

The synthetic dataset is a collection of battery cell data taken from [29], where the authors have tried to estimate the state of health of the batteries. It consists of 226x19 samples of tabular data collected from Li-ion battery cells in the field. There are 18 features, and the prediction target is discharge capacity.

Northvolt dataset

The Northvolt dataset consists of 5573x132 samples of tabular data where each row represents a cell and each column represents a measurement taken during the CA and F&A sections of the production line. The battery cells are early samples from the demonstration manufacturing line and research facility Northvolt Labs. There are three different cell models having slightly different production parameters. In Chapter 5, the features and targets have been generalized to conform to a non-disclosure agreement.

4.2 Pipeline

The proposed machine learning pipeline is illustrated in Figure 4.1. Each part of the pipeline is explained in detail in the sections below.

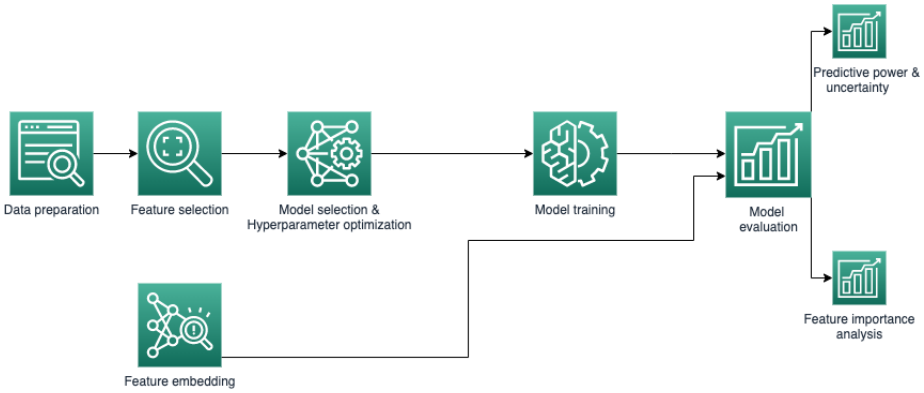


Figure 4.1: Illustration of the proposed machine learning pipeline.

4.3 Data preparation

The data is collected raw from the machines and hence requires some preparation before being used as inputs to the models. We describe the data preparation below.

Outlier removal

A first step when working with untidy data is to look for outliers. Outliers exist in almost any dataset and can severely damage the performance of a model [10]. The outliers for each feature are removed by keeping the central 95% within each battery model sub-distribution. We choose 95% to ensure that machine mismeasurements are removed from the population.

Normalization

Many models require normalization of the data. For instance a Gaussian process regressor assumes the target having zero mean. Standard score (Z-score) normalization is used in this thesis. The formula is given by:

$$z = \frac{x - \mu}{\sigma}, \tag{4.1}$$

where x is the original value, μ and σ the mean and standard deviation for that feature/target.

Categorical value encoding

Most regression models require numerical features (rather than categorical). There are several methods to represent a categorical feature numerically. The performance of the methods depend both on the data and models at hand. [24] gives a comparative study of encoding methods for neural network classifiers.

One-hot encoding and label-encoding are tried in this thesis. One-hot encoding creates n number of binary *dummy*-features, where n is the number of distinct values within the categorical feature. If we have a categorical value of color, with three samples distributed into three classes: $[red, green, blue]$, the one-hot encoding would be:

Table 4.1: One-hot encoding example.

sample	Red	Green	Blue
x_1	1	0	0
x_2	0	1	0
x_3	0	0	1

The main disadvantages of one-hot encoding are the explosive increase of dimension for categorical values with many classes, and the impossibility to rank the categorical features. For some categorical features, there might exist some ordinality that we want to express.

Label-encoding on the other hand simply maps each distinct categorical value to an integer. So in the same example as above, a label encoding would yield: $[\{red : 0\}, \{green : 1\}, \{blue : 2\}$. Here, we do not encounter the problem with increased dimensionality. However, in contrast to one-hot encoding, we here impose an ordinality that might not exist. Green (1) becomes the mean of red (0) and blue (2) which may or may not be true.

Train/test split

The dataset is divided into a training set, and test set, with a ratio of 80/20. The test set is untouched for the entirety of the training process, and only used for the final evaluation of the models. The training set is used most efficiently by training the models using k-folds cross validation. K-folds cross validation divides the training set into K subsets. Then, in K iterations, one subset is left out and the model is trained on the included subsets of the data. The model is then validated on the subset that was left out. This is done until all subsets have been used for validation. The model results are then calculated as the mean of the results from each subset [11].

4.4 Feature selection

When building regression models working with large data sets, a lot of feature space exploration and is needed to optimize performance both in terms of computation and predictive performance. Since the factory data available from Northvolt is vast, simply adding all possible features into our models would yield disappointing results. Many of the available features would only bring noise into the models making it harder for them to determine the actual signals. Hence, removing abundant features is of crucial importance.

The feature selection is done in a two-part process. For each target, an optimal random forest regressor is fitted for the entire feature space using randomized hyperparameter search with cross validation as is explained in Section 4.6. The resulting model is then passed into a recursive feature elimination algorithm with cross validation (RFECV) to find the optimal number of features for each target. This is done by first including all features as input to the model, evaluating the results of the model, ranking the features by importance and then discarding the least important ones, and finally refitting the model. The process is repeated until either no more features remain, or if a minimum number of features decided by the programmer is obtained. The evaluation results of each set of features are then compared to find the optimal features.

4.5 Feature embedding

Feature embedding is another tool for reducing the dimensions of the feature space. The objective here is to find a latent space representation of the feature space to be used for visualization or as input to the model. In this thesis, feature embedding is used solely for visualizing the relations between our features and targets. There are myriad of methods for embedding the feature space. The below methods were tried in the thesis.

4.5.1 t-SNE

t-distributed stochastic neighbor embedding (t-SNE) is a machine learning algorithm for finding a 2- or 3D latent space for higher dimensional data visualization. The t-SNE algorithm constructs a probability distribution over pairs of data points, such that similar data points are assigned higher probability than dissimilar ones. It then constructs a 2- or 3D map of the higher dimension probability distribution and minimizes the difference between the two distributions. Consult the original paper [33] for additional details.

t-SNE works great for grouping data samples that are similar in feature space to a 2- or 3D mapping. However, the main issue with t-SNE is its inability to preserve the global structure of the original feature space. This means that we would never be able to understand the inter-cluster relatedness in the latent space that t-SNE creates. The phenomenon is illustrated in Figure 4.2.

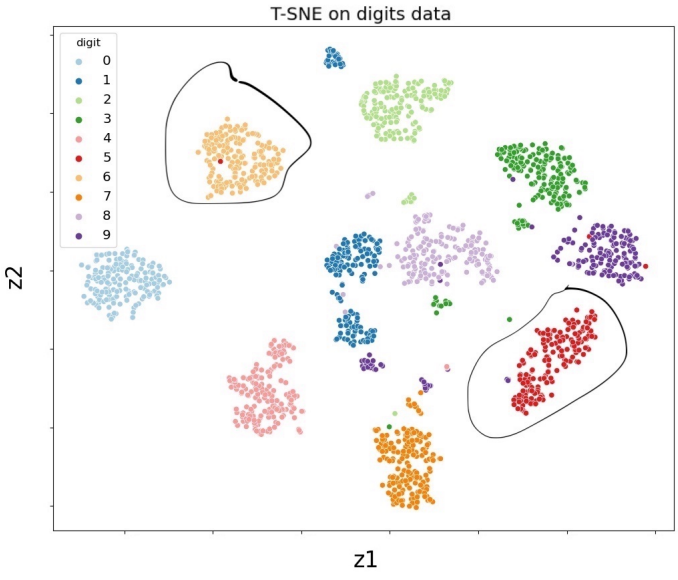


Figure 4.2: t-SNE on digits data where each color represents one digit. The axes $z1$, and $z2$ represent the latent space dimensions.

Here, we have applied the t-SNE algorithm on a version of the mnist dataset [16], consisting of handwritten digits together with ground truth labels. Each image is 8x8 yielding a dimensionality of 64. As we can see in Figure 4.2, t-SNE successfully clusters most digits of

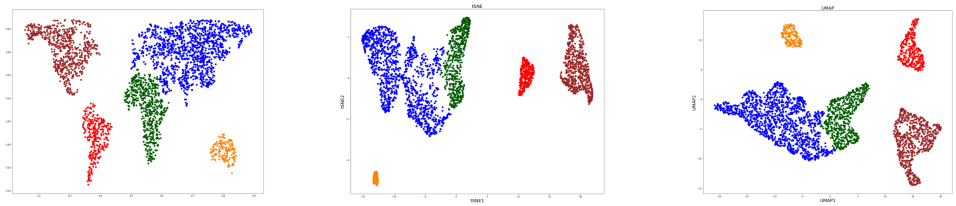
the same class to the same cluster. However, we cannot say anything regarding similarities between clusters. To illustrate this, we can look at the clusters for digits 5 and 6 (circled in the image). Digits 5 and 6 are very similar in how they look and hence also in terms of their pixel distributions. This is additionally enforced by the fact that one of the images for the digit 5 has been placed within the cluster for the digit 6. However, looking at the 2D latent space representation, the clusters for digits 5 and 6 are those that are the farthest apart.

4.5.2 UMAP

Uniform manifold approximation and projection (UMAP) is a feature embedding algorithm similar to t-SNE that can be used both for visualizing high dimensional data, and also for general dimensionality reduction.

The UMAP algorithm tries to preserve the global structure of the feature space by making use of a different cost function, compared to t-SNE [19]. A thorough comparison of the global structure preservation between the algorithms is given here [22].

Once again, we can illustrate how the algorithm handles the global structure with an example (Figure 4.3). Here, t-SNE and UMAP is applied to a dataset with a 2D representation of the world map along with labelled continents.



(a) World map **(b)** t-SNE **(c)** UMAP

Figure 4.3: Reconstruction of the world map data set using t-SNE and UMAP [22].

As is shown in Figure 4.3, t-SNE fails to preserve the global structure, since South America is placed in between Asia and North America. However, albeit rotating the data, UMAP performs better in preserving the original distances between the continents.

4.6 Model selection

The regression models in the thesis are chosen based on inspiration taken from [29] where the authors successfully prove the validity of the models on a battery cell dataset.

As explained in Section 3.2, each regression algorithm has several parameters that gets optimized during training. There are also hyperparameters that are predefined by the programmer and do not change during the training sessions. These hyperparameters are critical to tune in order for the algorithm to perform optimally. There are no hyperparameter settings that work for every case. One must tune them based on the problem and dataset at hand. In this section we will briefly explain the hyperparameter optimization done for each model.

Catboost & RF regressor

The optimal hyperparameters for the Catboost- and random forest regressors are found by using randomized search 5-fold cross validation (RSCV). Here, a fixed number of hyperparameter settings is sampled from predefined distributions. The results of the model using the sampled parameters are then cross-validated.

Gaussian process regressor

For the Gaussian process regressor, a pipeline is set up where kernels are optimized using RSCV. Kernel families investigated include radial basis function kernels, constant kernels and Matérn kernels. The inspiration for these choices are taken mainly from [29]. Additional inspiration is drawn from [8] where an overview of commonly used kernels in learning systems is given.

4.7 Feature importance

The area of measuring feature importance and model interpretability deserves a thesis project of its own. In the below section, the primary methods used in this thesis are explained. Additional theory is provided in Section 3.4.

4.7.1 Pearson correlation

Correlation quantifies the relationship between two variables. It can be measured in several ways, where the most common one is Pearson's correlation. The Pearson correlation coefficient is given by:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}, \qquad r \in \{-1, 1\}, \tag{4.2}$$

where $\bar{x}$ and $\bar{y}$ are the mean values of the two variables. If two variables have a correlation coefficient of 1, then for every positive increase in one of the variables, there is a positive increase of a fixed proportion in the other. A significant issue with Pearson correlation is that it fails to quantify nonlinear relationships [2].

4.7.2 SHAP

SHapley Additive exPlanations (SHAP) is a method for explaining model predictions using the game theoretical concept of Shapley values (explained in Section 3.4). The objective is to compute how much a specific feature contributes to the model prediction. The SHAP method models Shapley value explanations as an additive feature attribution method (a linear surrogat model). The equation for the linear surrogate model is given by:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_j z'_j. \tag{4.3}$$

Here g is the surrogate model. $z' \in \{0, 1\}$ is the coalition vector where each index j represents a feature that can either be 1 (included) or 0 (excluded). ϕ_j is the Shapley value for feature j, and M is the maximum allowed coalition size. Summing the effects of all included feature attributions yields an approximation of the output of the original regression model f, using the including features.

The Shapley values for each sample $(\boldsymbol{x}, y)$ and feature are estimated using KernelSHAP. KernelSHAP can be explained in the following 5 steps [20]:

1. Form the coalitions $z'_k \in \{0, 1\}^M$, $k \in \{1, ..., K\}$ for each sample. This can be done by sampling from a binomial distribution. In a 3-D feature space, the vector $z' = (0, 1, 1)$ means a coalition with the second and third feature. Our K coalitions become the dataset for our additive feature attribution method, where the target for the model is the original regression model's (f's) prediction for the sample and coalition.

2. For each z'_k, map the binary coalition vector back to the original feature space using a predefined mapping function h_x. Then input the result of the mapping to the original regression model f. The mapping function h_x, maps included features, $(z'_k)_j = 1$, to their respective values for that sample. Excluded features, $(z'_k)_i = 0$, get mapped to a random value from that feature value distribution.

3. Calculate the weight for each sample z', by using the proposed SHAP kernel [18]:

$$\pi_x(z') = \frac{(M-1)}{(\frac{M}{|z'|})|z'|(M-|z'|)}, \tag{4.4}$$

where $|z'|$ is the number of present features in z'. Lundberg and Lee [18] show that regression using this weighting kernel yields Shapley value regression coefficients.

4. Fit our linear model (4.3) using the dataset and weights acquired in the previous steps, by minimizing the sum of squared errors [18]:

$$L(f, g, \pi_x) = \sum_{z' \in Z} [f(h_x(z')) - g(z')]^2 \pi_x(z'), \tag{4.5}$$

yielding the same problem as we had in (3.29), (without the complexity $\boldsymbol{\Omega}$). The estimated regression coefficients, ϕ_j, are the Shapley values.

4.8 Visualizing SHAPley values

Together with the theoretical framework, Lundberg and Lee [18] have created a software toolkit called SHAP. The SHAP toolkit includes functions both for estimating and visualizing Shapley values and feature importance. One such visualization is what the authors call a summary plot, see Figure 4.4.

There is a lot going on in this plot so we will deconstruct it and look at its constituent parts. On the y-axis, we have the global feature importance, where each feature is ranked from the top (most important) to the bottom (least important). On the x-axis, we have the SHAP values for each feature for each sample in the dataset. This is easier understood by looking at the same plot for one sample only, illustrated in Figure 4.5.

A positive Shapley value for a feature means that the inclusion of that feature pulled the model's prediction upwards. Feature 2 is most important for this particular sample, impacting the model by pulling the prediction upwards. Feature 4 is next most important, impacting the model by pulling the prediction downwards.

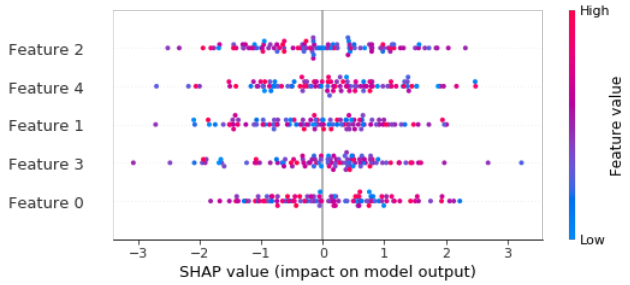


Figure 4.4: Example SHAP summary plot of generated toy data [21].

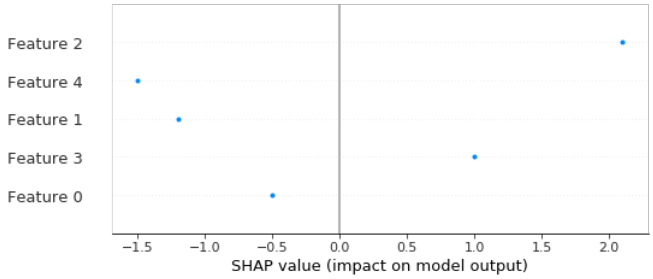


Figure 4.5: Example SHAP summary plot: single sample of generated toy data [21].

Finally we have the color grading, indicating if the particular feature values for each sample are high or low in comparison to each feature value distribution. If a feature value ranges between 0 and 10, a sample with that feature value equal to 1 would be colored blue.

Putting this all together gives an exhaustive overview of what features and feature values that impacted the model the most.

4.9 Evaluation metrics

The models are evaluated in terms of predictive power and uncertainty.

4.9.1 Predictive power

We measure the predictive power of the models using mean absolute percentage error (MAPE) and root mean squared percentage error (RMSPE). Both metrics express the average model prediction error in percentages. RMSPE squares the errors before averaging, yielding a higher weight to large errors. The formulas for MAPE and RMSPE are given in (4.6) and (4.7) respectively.

$$E_{MAPE} = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|, \tag{4.6}$$

$$E_{RMSPE} = 100\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2}. \tag{4.7}$$

Here, N is the number of samples. y_i and $\hat{y}_i$ are the true and predicted values for sample i. The multiplication of 100 is often omitted in the literature. However, as becomes obvious by studying the formulas, and as is indicated here [14], we do not get the errors expressed in percentages if omitted.

4.9.2 Uncertainty

In order to measure the uncertainty of our predictions, we need to calculate their standard deviations. The standard deviation for the predictions are calculated differently depending on model type. For the Gaussian process regressor, we simply calculate the standard deviations from the diagonal of the covariance matrix of the GP posterior as is explained in Section 3.3.3. For the random forest regressor, we make use of a version of the jackknife method as proposed in [35]. For the Catboost regressor, we divide our training set and train multiple Catboost models using cross-validation. Each model is tested and evaluated on the test set. The standard deviation is then calculated from the set of model predictions.

Calibration Score

Calibration score measures the fraction of our predictions that lie inside a 2σ-range from the true values. Here σ is the standard deviation for the true values. The calibration score gives us an idea of how well our models are performing in relation to the variance of the target variable. The formula is given by:

$$CS_{2\sigma} = \frac{1}{N}\sum_{i=1}^{N}[|y_i - \hat{y}_i| < 2\sigma]_I, \tag{4.8}$$

where, $[]_I$ represents an Iverson bracket:

$$[P]_I = \begin{cases} 1, & \text{if } P \text{ is true} \\ 0, & \text{otherwise.} \end{cases} \tag{4.9}$$

Sharpness

The sharpness Sh is simply the average model standard deviation for each prediction and is generally calculated in absolute terms. Since we cannot show the actual values of each target, a normalized version of sharpness is proposed. Here, the sharpness is expressed in terms of the ratio to the true value. This yields:

$$Sh_{norm} = \frac{1}{N}\sum_{i=1}^{N}\frac{\sigma_i}{y_i}, \tag{4.10}$$

where σ_i is the standard deviation for prediction i.

Additional uncertainty measures

The $\alpha_{accuracy}$ and β scores, as proposed by [29] measure the uncertainty of a model's prediction from an engineering perspective. For each target value, an accuracy zone is calculated as a

percentage error α from the true target value. This yields an accuracy zone of $\boldsymbol{y} \pm \alpha$, where $\boldsymbol{y}$ is the target value vector, and α is the chosen percentage error. Following the example of [29], α is chosen to 1.5%. We then get the $\alpha_{accuracy}$ as the fraction of predicted values $\hat{\boldsymbol{y}}$ that lies within the accuracy zone.

To calculate the β score we first construct normal distributions of each prediction using the prediction value and its standard deviation. The β score is then the average intersection of the probability mass of the normal distributions and the accuracy zone. Ideally we want a β score of 1, suggesting that the variance in our predictions is entirely encapsulated within the accuracy zone. Figure 4.6 illustrates α and β scores for the prediction of capacity.

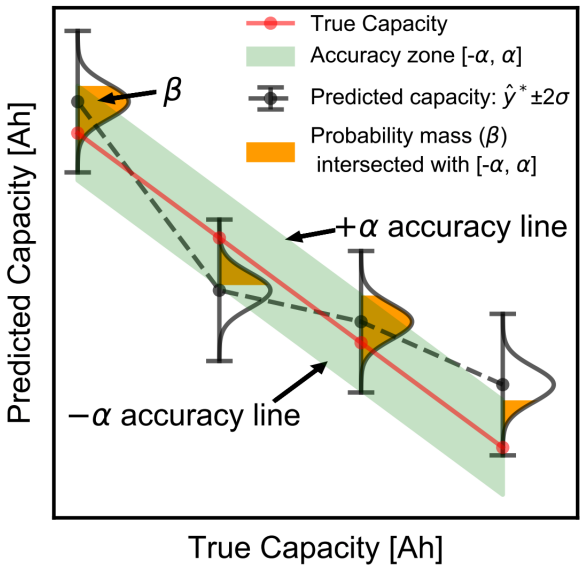


Figure 4.6: Illustration of α-accuracy and β-probability mass distribution [29].

R-squared

R-squared (R^2) measures how well a regression model can explain the variance in a target variable, given input data. It is calculated as:

$$R^2 = 1 - \frac{RSS}{TSS}, \tag{4.11}$$

where RSS, and TSS is residual- and total sum of squares, given by:

$$\begin{aligned} RSS &= \sum_{i}^{N} (y_i - \hat{y}_i)^2, \\ TSS &= \sum_{i}^{N} (y_i - \bar{y})^2, \end{aligned} \tag{4.12}$$

where $\bar{y}$ is the mean of the target and the other variables are defined as previously. R^2 by itself does not quantify the reliability of a regression model. Some datasets inherently have a greater amount of unexplainable variation.

Chapter 5
Results and discussion

In this chapter, we first test and discuss the validity of our tried methods on a synthetic dataset. Then, the results obtained on the Northvolt dataset are explained in detail for each prediction target and model.

5.1 Predicting capacity on a synthetic dataset

The synthetic dataset comes from [29], where the authors try to estimate the capacity fade of battery cells in the field. The code for the following section can be found here [34].

In Figure 5.1 we have used UMAP to find a 2D latent space representation of the feature space. The prediction target is then added as color to the samples. We see a clear gradient of the prediction target in the latent feature space, where samples with lower capacity are spread out in the upper right corner, and higher capacity samples are grouped to the bottom left.

This indicates that there is a strong signal between the features and target in the dataset. Samples that are similar in feature space, are also similar in target space.

The dataset is split into train/test, and multiple Catboost regressors are trained on different folds of the training data. We only evaluate the Catboost regressor since the goal is to show that *any* well designed regressor can fit the data arbitrarily well provided there exists a signal between the features and target. Each Catboost model is evaluated on the test data, and the aggregated results can be seen in Table 5.1.

Table 5.1: Evaluation results for Capacity synthetic dataset).

Target	$\sigma^{\%}_{target}$	Model	MAPE	RMSPE	C_{score}	Sh_{norm}	$\alpha_{accuracy}$	β	R^2
Capacity	8.322	cb	1.245	3.440	1.0	0.003	0.826	0.832	0.911

The evaluation metrics displayed in Table 5.1 are covered in detail in Section 4.9. $\sigma^{\%}_{target}$ is the percentage standard deviation of the target value distribution. It is normalized to enable

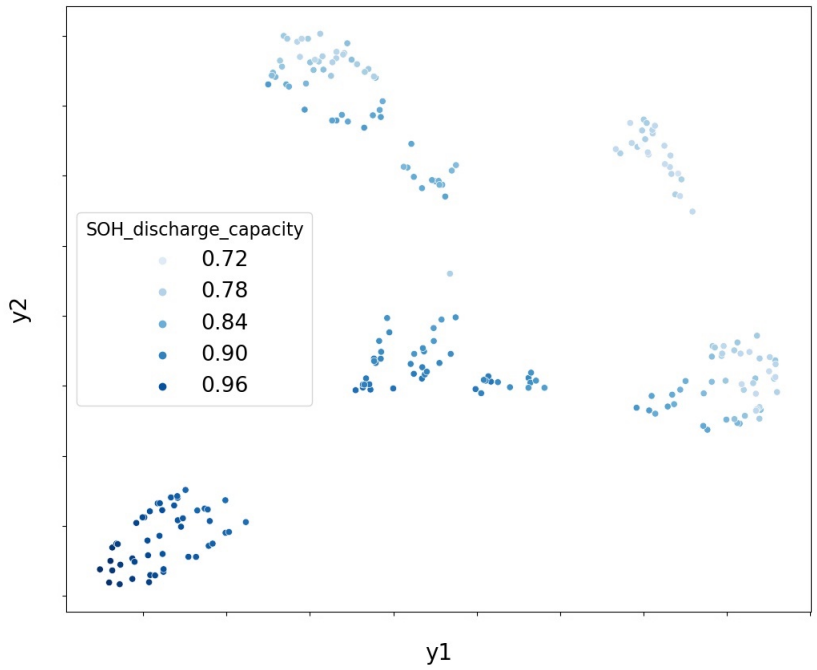


Figure 5.1: UMAP 2D representation of the synthetic-dataset feature space, with the prediction target distinguished by color. The axes $y1$, and $y2$ represent the latent space dimensions.

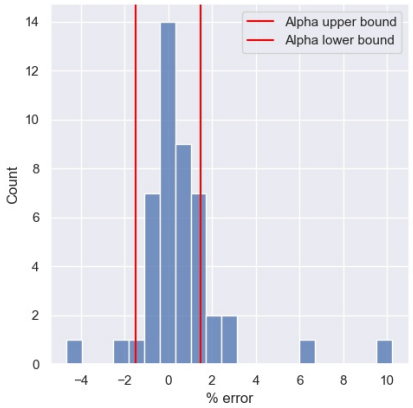


Figure 5.2: Percentage error distributions on the synthetic dataset.

easier comparisons with the target standard deviations of the Northvolt targets (that we need to normalize due to non-disclosure agreement). To make it easier to compare it to MAPE and RMSPE, it is expressed in percentages. A standard deviation of 8.3% means that there is not much variance in the target distribution. This can lead to problems as we will see in the following section. The percentage error distribution is visualized in Figure 5.2. By studying 5.1 and 5.2, we see that the Catboost model achieves great scores, both in terms of predictive power and reliability.

In Figure 5.3, we see the global SHAPley value plot for the model. Lagged Pseudo Resistance is by far the most important feature. Almost all samples either greatly pushed the models prediction downwards, or upwards. If we examine the feature values of lagged pseudo resistance of the samples that impacted the model the most, we see that samples with a very high resistance, greatly pushed the models prediction downwards. This makes sense from an electrochemical point of view. We remember from Section 2 that a higher resistance has a great impact on a battery cell's power characteristics.

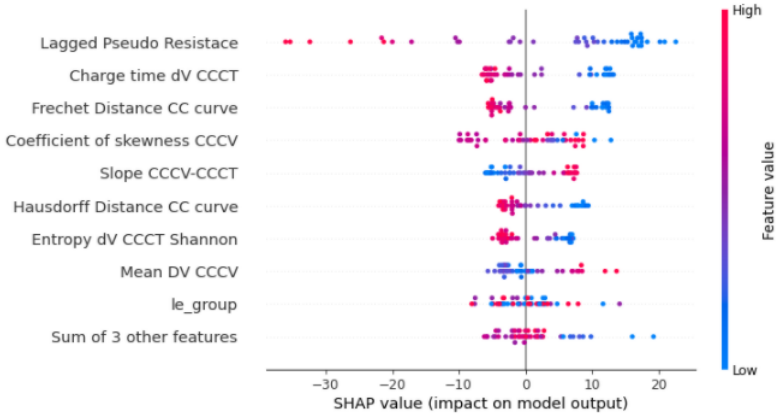


Figure 5.3: Global shapley value summary plot for synthetic dataset.

5.2 Northvolt dataset

In this section, we first investigate the target and feature spaces of the Northvolt dataset. Then we explain our results for each of our three targets. All target values are normalized due to a non-disclosure agreement.

5.2.1 Feature- & target space exploration

Linear correlations between feature and target

For each target, Pearson correlations between the available features and the target are calculated. None of the features are strongly correlated with the targets, with the highest correlation in absolute terms measuring to 0.4. This indicates one of the following: i) there simply is no signal between our features and targets in the dataset, ii) the correlations between our features and targets are nonlinear and/or interdependent. This would mean that some nonlinear combination of our features might correlate more strongly with the targets. Finding and illustrating such nonlinear correlations is difficult. In the following section, we find a 2D latent space representation of feature space to make it easier to explore the relationships between our features and targets.

Nonlinear representation of the feature space

In Figure 5.4, we see the distributions of our prediction targets, grouped on battery model. As becomes evident by studying the graphs, outlier removal is needed to get a clearer view of the distributions.

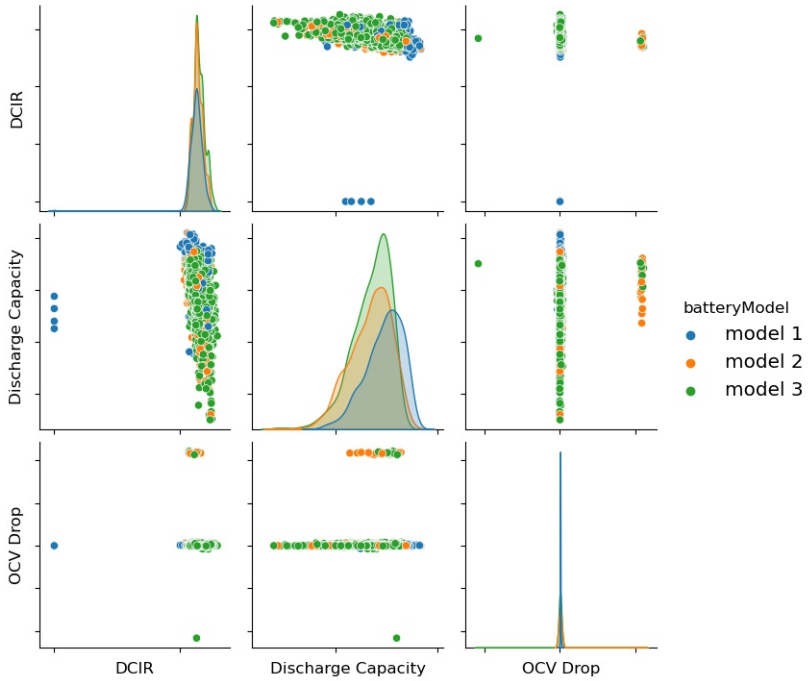


Figure 5.4: Target distribution, grouped on battery model.

Outlier removal is performed by keeping the central **95%** within each battery model sub-distribution. This results in the following target distributions illustrated in Figure 5.5.

For the internal resistance (DCIR) we see that there are multi-modal distributions for each of the battery models. For Discharge Capacity and Open Circuit Voltage (OCV) drop, the distributions for battery model 1 stand out in comparison to the other models. This is interesting since the different battery models are produced differently, with varying production parameters. To verify that this is the case, we use UMAP as described in Section 4.5.2, to create a 2D latent space representation of the feature space, without including battery model. The idea is to validate that the differences in production parameters for the different battery models are significant enough for the UMAP algorithm to clearly distinguish them into clusters. The results are illustrated in Figure 5.6.

As seen in Figure 5.6, UMAP manages to group the different battery models together. This tells us that there is a distinction between the production parameters for the different battery models, as was expected. However, what is worrying is that the two battery models that are close to each other in feature space, are not the same battery models that are close together in target space. Remember that, in Figure 5.5, the target distributions for battery model 1 stood out both for Discharge Capacity and OCV drop. In feature space, battery model 3 is the one that stands out. This is important since it indicates that there is a signal in the production data that we are currently not collecting. The phenomenon is perhaps most easily understood with the help of a 1D toy example. Say we have a regression model as in eq.

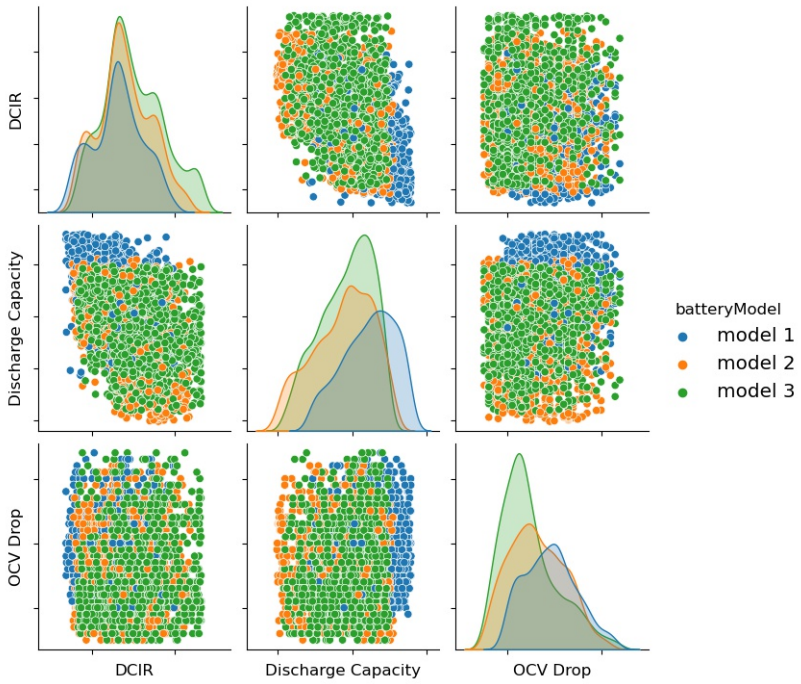


Figure 5.5: Target distribution after outlier removal, grouped on battery model.

(3.1) that tries to learn how to map an input value to a target value in 1D. It will be completely impossible for regression models, humans and super intelligences alike, to find this mapping if the data samples with the same input value have completely different target values. Then, by definition, there exists no mapping between input and output. The problem is additionally enforced by looking at Figure 5.7. Contrary to our synthetic example, the target variables are spread out over the entire 2D space. The similarities between cells in feature space, do not correlate with the similarities between cells in the target spaces.

5.2.2 Direct current internal resistance

Feature selection

The RFECV-algorithm finds 10 features as optimal for predicting DCIR. The features along with explanations are found in Table 5.2. The features are ranked according to their average impact in the feature importance section.

Algorithm performance

The models for each target are evaluated on test data that has been completely unseen during training. The evaluation results and percentage error distributions can be seen in Table 5.3 and Figure 5.8.

By just looking at the MAPE and RMSPE, it seems that all models achieve great scores, with percentual errors ranging between $2-3\%$. However, by studying $\sigma^{\%}_{target}$ we see that the

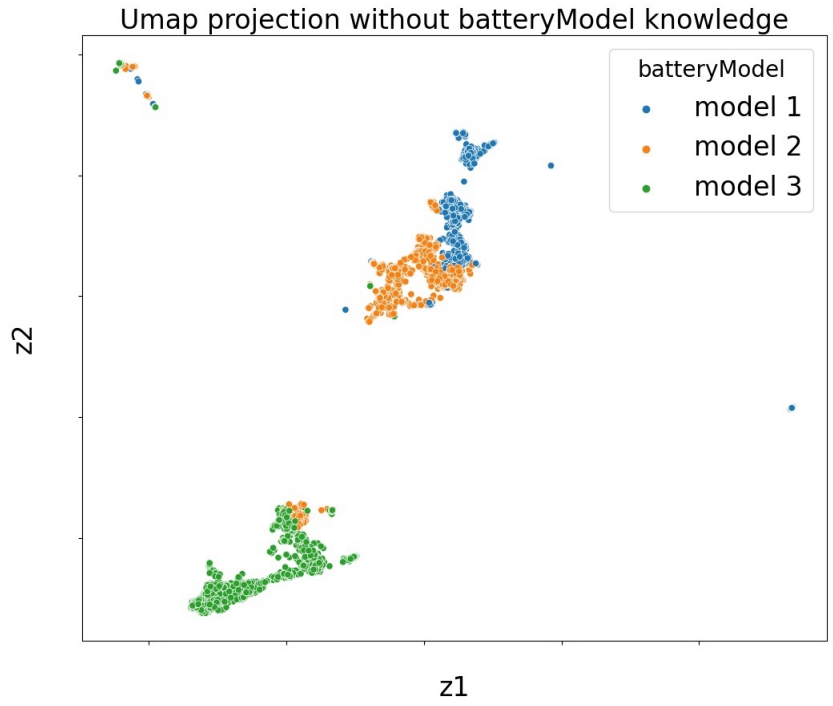


Figure 5.6: UMAP 2D representation of feature space, without inclusion of battery model. The axes $z1$, and $z2$ represent the latent space dimensions.

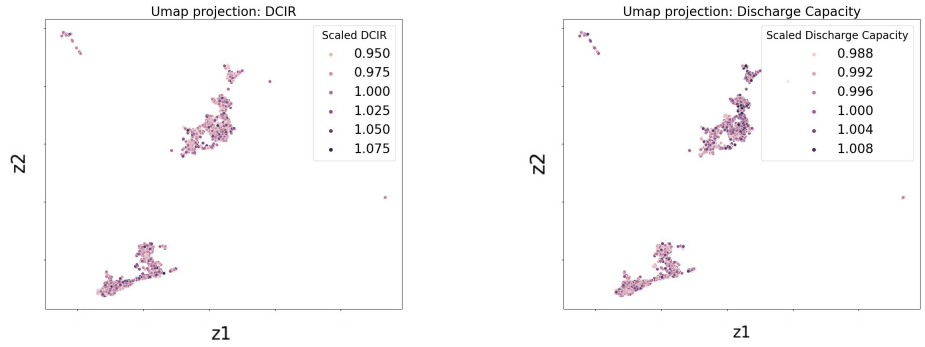
Table 5.2: Features deemed as optimal for predicting DCIR using the RFE-CV algorithm.

Feature	Explanation
ir measurement 1, 2	Measurements of internal resistance done prior in the process
pressure measurement 1, 2	Measurements of pressure within the cell
temperature measurement	A measurement of temperature within the cell
label encoded model	Label Encoded version of battery model
capacity measurement	A capacity measurement done prior in the process.
internal distance measurement 1, 2	Measurements of distance between internal components of the cell
force measurement 1	A measurement of force applied to the cell at one stage of production

Table 5.3: Evaluation results for direct current internal resistance (DCIR) for the three models; Catboost (cb), Random forest (rf) and Gaussian process regressor (gpr)

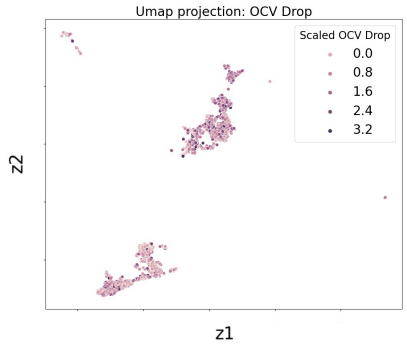
Target	$\sigma^{\%}_{target}$	Model	MAPE	RMSPE	C_{score}	Sh_{norm}	$\alpha_{accuracy}$	β	R^2
DCIR	3.167	cb	2.213	2.669	0.983	0.003	0.434	0.430	0.277
		rf	2.176	2.734	0.978	0.012	0.434	0.349	0.119
		gpr	2.358	2.941	0.961	0.005	0.390	0.386	0.239

standard deviation of the target value is also very low, meaning a 'dummy-model' only guessing the mean of the target would perform quite well. Since the MAPE is above our chosen



(a) DCIR

(b) Discharge Capacity



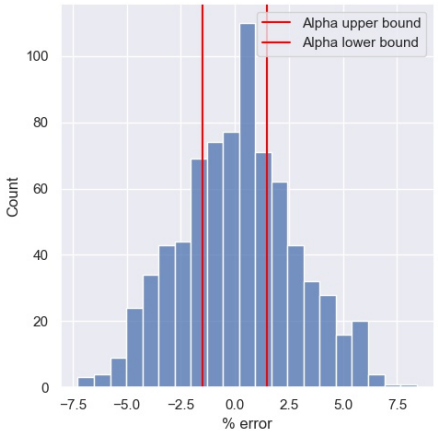
(c) OCV Drop

Figure 5.7: 2D latent space representation of feature space with the prediction targets distinguished by color. The axes $z1$, and $z2$ represent the latent space dimensions.

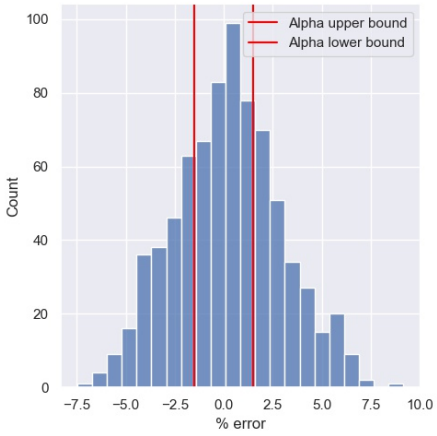
$\alpha = 1.5\%$, only around 40% of our predictions lie within our constructed α-interval. Due to the low Sh_{norm} for both the Catboost and GPR models, the β-scores are very close to the $\alpha_{accuracy}$-scores. The RF-model has a higher Sh_{norm}. This means that the normal distributions that we create out of the predictions and the standard deviations become wider. This in turn yields a lower probability mass of the distributions lying within the α-zone. All models have low R^2-scores, indicating they fail to explain the variance in the target variable.

Feature importance

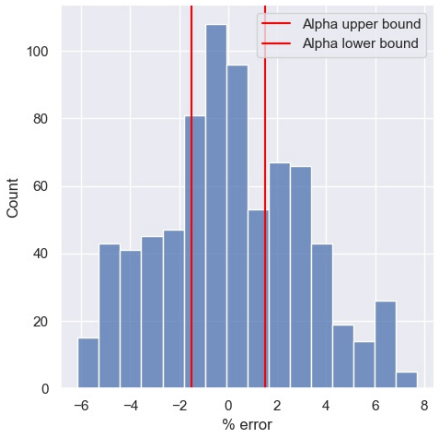
Due to the uncertainty of the model predictions, we should take the findings regarding feature importance with a grain of salt. These features have been selected using the RFE-CV algorithm described in Section 4.4. However, since the results were only slightly better after the inclusion of the features, we cannot be certain that all of these features actually play an important role in predicting our target. That being said, all three models rank *ir measurement 1* as the most important feature. This comes as no surprise since both *ir measurement 1* and our target DCIR are measurements of internal resistance. Samples with higher *ir measurement 1* pushed the model prediction upwards, which makes sense electrochemically as well.



(a) Catboost



(b) RF



(c) GPR

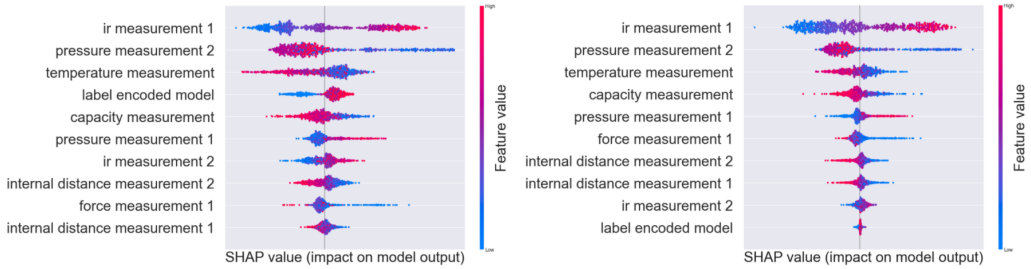
Figure 5.8: Percentage error distributions for DCIR the different models.

5.2.3 Discharge capacity

Feature selection

For predicting discharge capacity, the RFECV-algorithm finds 9 features as optimal. The features are displayed in Table 5.4.

The feature *capacity measurement* is a discharge capacity measurement similar to the target variable. The difference between the two is that the first discharge session is part of the actual production process. During this session, a solid electrolyte interphase is formed on the negative electrode particle surfaces. The second session is done mainly for quality validation. The features are ranked in the feature importance section.



(a) Catboost **(b)** RF

(c) GPR

Figure 5.9: SHAP summary plots for DCIR.

Table 5.4: Features deemed as optimal for predicting Discharge Capacity using the RFE-CV algorithm.

Feature	Explanation
capacity measurement	A capacity measurement done prior in the process
label encoded model	Label Encoded version of battery model
electrode sheet size measurement	An aggregate measurement of the sizes of electrode sheets included in the cell
internal distance measurement	A measurement of distance between internal components of the cell
weight measurement	Cell weight measurement
pressure measurement 1, 2	Measurements of the pressure within the cell
high voltage measurement	A high voltage measurement taken to determine effectiveness of insulation
force measurement	A measurement of force applied to the cell at one stage of production

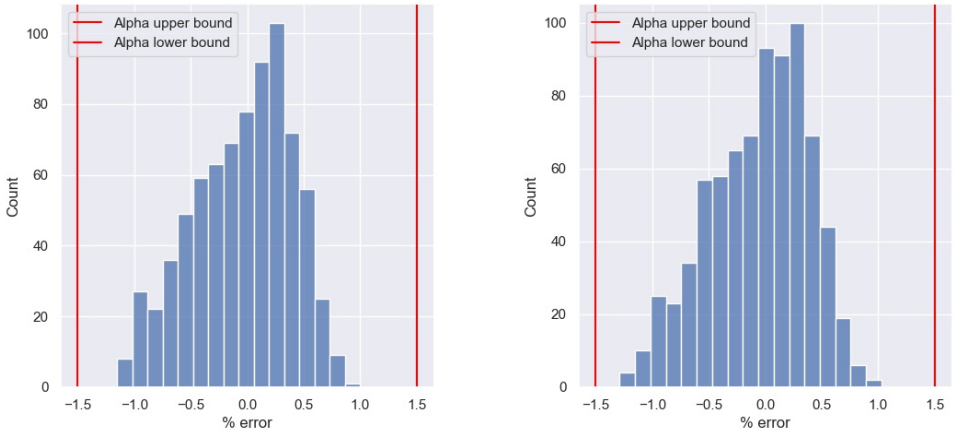
Algorithm Performance

The evaluation metrics along with the percentage error distributions are displayed in Table 5.5 and Figure 5.10 respectively.

Table 5.5: Evaluation results for discharge capacity.

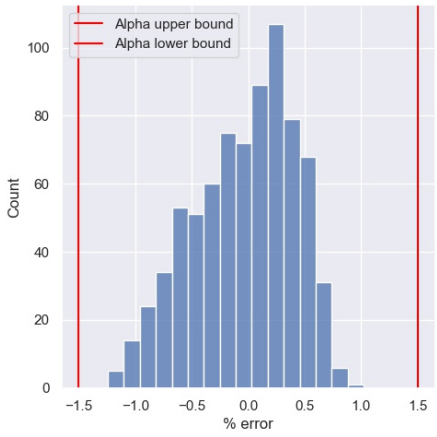
Target	$\sigma^{\%}_{target}$	Model	MAPE	RMSPE	C_{score}	Sh_{norm}	$\alpha_{accuracy}$	β	R^2
dc cap	0.492	cb	0.359	0.438	0.984	0.0004	1.0	1.0	0.209
		rf	0.359	0.443	0.974	0.0027	1.0	0.997	0.174
		gpr	0.371	0.448	0.981	0.0007	1.0	0.999	0.192

For this target variable the MAPE and RMSPE are extremely low, ranging from 0.3–0.4% for the different models. However, once again we have the case of low variance within the target value distribution as is indicated by $\sigma^{\%}_{target}$. Since the target variables are so centered around the mean of the target distribution, just guessing the mean will in this case yield impressive results. For all models, all predictions lie within our constructed α-zone. This



(a) Catboost

(b) RF



(c) GPR

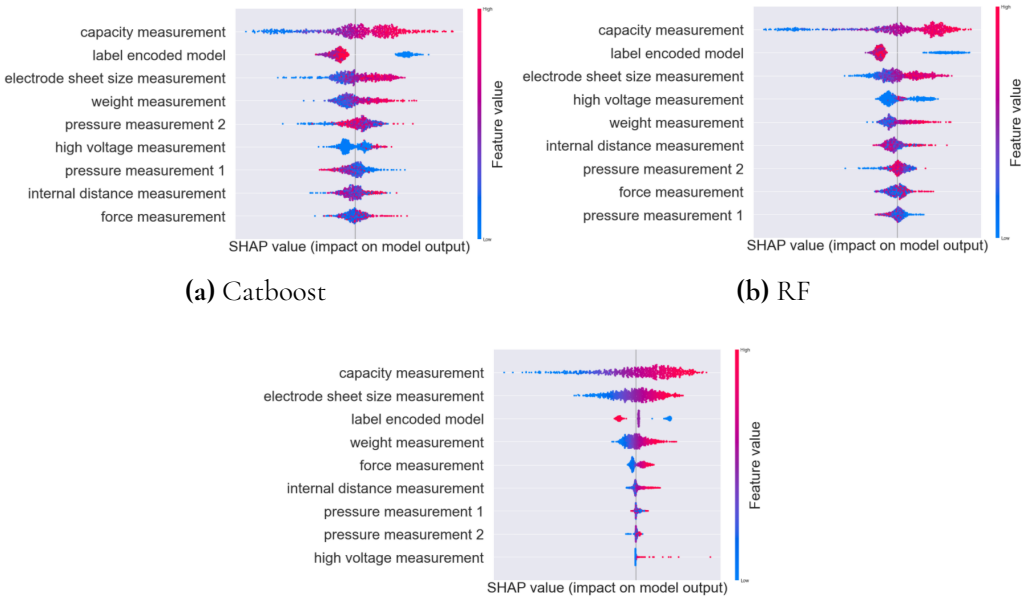
Figure 5.10: Percentage error distributions for discharge capacity the different models.

is an indicator that we should construct separate α-zones for our different targets. Using an interval of ±1.5%, as advised in [29] might be too wide for this target. Due to the low variance of the predictions, the β-scores are similar to the $\alpha_{accuracy}$-scores. Once again we have low R^2-scores for all three models. This indicates that the models struggle to explain the minimal variance that we have in the target variable. We should therefore be cautious when looking at the feature importances in the next section.

Feature importance

The SHAP summary plots for each model are displayed in Figure 5.11.

Not surprisingly, the capacity measurement corresponding to a discharge measurement done prior in the process, is deemed as most important for all models. Once again we see the correlation of high feature values and high target predictions. We also see that all three models rank the label encoded version of battery model highly. This is also not surprising since we saw in Figure 5.5, that there was a clear difference between the Discharge Capacity



(a) Catboost **(b)** RF

(c) GPR

Figure 5.11: SHAP summary plots for Discharge Capacity.

target distribution for model 1 compared to the other models. The color grading is here misleading since battery model is a categorical feature, without ordinality.

The electrode sheet size measurement corresponds to the minimum electrode sheet width of the cell. All models deem this feature important for predicting the discharge capacity. This makes sense electrochemically since the electrodes houses the ions. A cell with an electrode that is too small will likely have problems delivering optimal capacity.

5.2.4 Open circuit voltage drop

Feature selection

The RFECV-algorithm finds 7 features as optimal for predicting OCV drop. The features along with explanations are found in Table 5.6.

Table 5.6: Features deemed as optimal for predicting OCV drop using the RFE-CV algorithm.

Feature	Explanation
capacity measurement	A capacity measurement done prior in the process
label encoded model	Label Encoded version of battery model
internal distance measurement 1, 2	Measurements of distance between internal components of the cell
temperature measurement	A measurement of the temperature within the cell
force measurement	A measurement of force applied to the cell at one stage of production

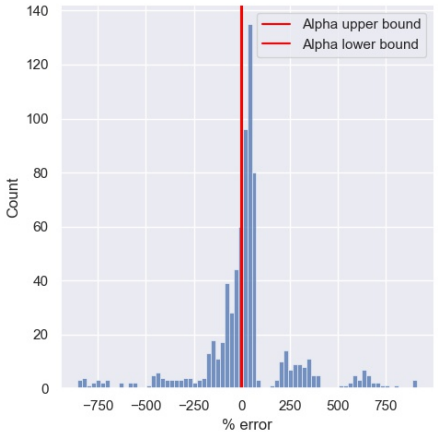
The features are ranked in the feature importance section.

Algorithm Performance

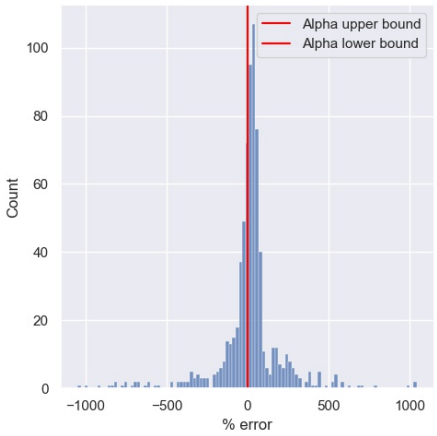
The evaluation metrics for predicting OCV drop can be found in table 5.7. The percentage error distributions are illustrated in Figure 5.12.

Table 5.7: Evaluation results for OCV drop.

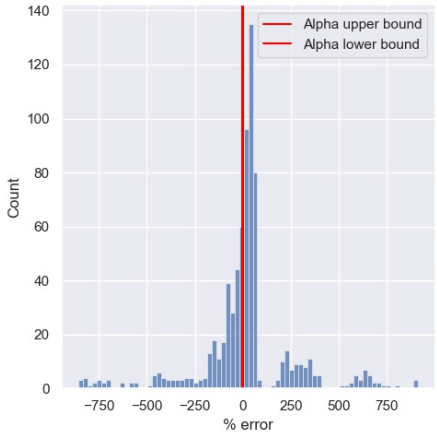
Target	$\sigma^{\%}_{target}$	Model	MAPE	RMSPE	C_{score}	Sh_{norm}	$\alpha_{accuracy}$	β	R^2
OCV drop	971.709	cb	123.463	218.316	0.978	0.092	0.022	0.015	0.294
		rf	116.263	206.281	0.971	0.434	0.017	0.015	0.127
		gpr	144.995	242.654	0.975	0.155	0.015	0.012	0.272



(a) Catboost



(b) RF



(c) GPR

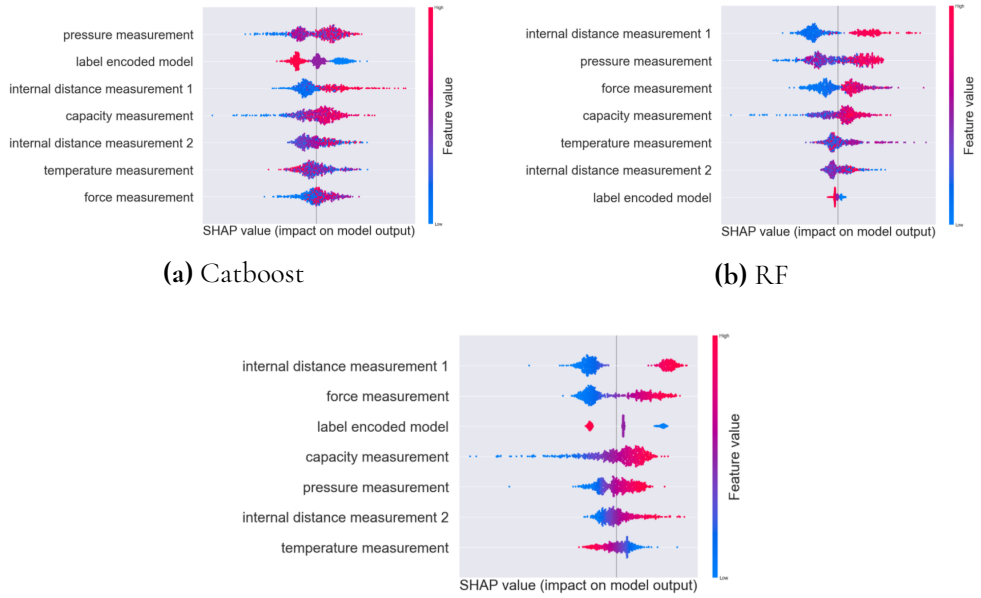
Figure 5.12: Percentage error distributions for OCV drop the different models.

By studying $\sigma^{\%}_{target}$ in Table 5.7, we see that there is high **normalized** variance in the OCV drop target value distribution compared to our other targets. This is partly due to OCV drop being a difference between two nearly equal measurements. Its distribution is hence centered close to zero, leading to higher normalized values. The MAPEs and RMSPEs

are comparatively worse than for the other targets ranging between $116 - 242\%$. However, the difference between the target standard deviation and our models' percentual prediction errors is greatest for OCV drop. So, even though our percentage errors are a lot higher than for the previous targets, the models in a sense perform better for this target. An additional indicator of the difficulty of predicting this target is given by studying the C_{score}. Even though our prediction accuracies are comparatively low, our C_{scores} are still close to 1 due to the high normalized variance within the target distribution. Studying $\alpha_{accuracy}$ and β, we once again get an indicator of the problems of using the same accuracy zone for all three targets. For this particular target, we should probably widen the accuracy zone. Similar to the previous targets, the R^2 scores are low for all three models.

Feature importance

Just as for the previous targets, due to high uncertainties and low R^2-scores, one has to be cautious while studying the results regarding feature importances. The SHAP summary plots for OCV drop are illustrated in Figure 5.13.



(a) Catboost **(b)** RF

(c) GPR

Figure 5.13: SHAP summary plots for OCV drop.

An additional indicator of the unreliability of the models' results is that they rank the importances of the features differently. In a setting where all features are strongly correlated to the target, this is not necessarily a problem. In our case we know of the inherently low correlations between our features and target and we thus need to be careful. The Catboost model ranks the force measurement the lowest, whereas the same feature is ranked top three for the random forest and Gaussian process regressors. The feature label encoded model is also spread across the rankings for the different models. This could very well be an indicator that those features are simply noise.

In the following section, we summarize our findings and propose future work to improve the models.

Chapter 6
Conclusion and future work

6.1 Conclusion

In this thesis we created a machine learning pipeline for estimating battery cell quality using production line data. The proposed pipeline consists of feature selection, data cleaning and preparation, hyperparameter tuning along with an exhaustive evaluation procedure focusing on both predictive accuracy and uncertainty. The main objectives were to find out whether we could use a subset of production line data to predict quality indicators that are measured at the end of production line, as well as to investigate what production parameters that are most important with regards to cell quality.

Judging by the results presented in Section 5 we are not able to reliably predict the EOL quality indicators using cell assembly and formation and ageing data. A common denominator for all targets is that there simply does not seem to be enough of a signal in the data.

For DCIR, our prediction accuracy is acceptable, but only slightly better than the accuracy one will get by predicting the mean of the target population. The low R^2 scores indicate that the models, using the input features, fail to explain the variance in the target variable.

For discharge capacity, the results look great at first glance. However, we here have a target variable with even less variance than for DCIR. Our models mainly perform as well as they do by simply predicting the mean, rather than using the variability in the feature space. However, the slight improvement gained from the features could partly be due to the minimum electrode sheet width within the cells. This is an interesting find that will be investigated further.

For OCV drop, we on the other hand have a lot of normalized variance in the target distribution. As a result, our prediction accuracy is way worse in comparison to the other targets. However, although we are comparing normalized performance metrics, we cannot draw one-to-one comparisons between the results of the different targets. It could be the case that for one target, a MAPE of 2% is way above the required limits, whereas 50% for another can be considered well within the bounds. It all depends on target and problem type. Hence,

the results for our three targets will be presented internally to get opinions from cell design experts.

6.2 Future work

For the targets with low variance, a reasonable idea to improve the performance of the models is to refrain from filtering out outliers. One could imagine a scenario where some of the signal lies within the outliers. However, this was not the case for the dataset used in this thesis. Here the outliers mainly consisted of obvious machine mismeasurements.

Another point of improvement could be to incorporate feature engineering in the training pipeline. As is shown here [29], feature engineering can have a great impact on model performance. Incorporating feature engineering would require close collaboration with battery cell engineers, to get inspiration for how one could manipulate the features we have into something more useful for the models.

For end of line validation, one could also try to include time as one of the features of the dataset. It is reasonable to believe that faulty cells come in batches. This aspect is completely overlooked in the current setup.

The project of using all cell assembly data to predict a few quality indicators in the end of the line might introduce too much noise. Instead one could try to break up the quality estimation into subparts during the line. So that with data collected in one process, we try to say something about the results of the coming process. There are however some clear disadvantages with such an approach. As [15] suggests, modelling processes individually is simply not enough to capture the complexity of the entire process chain. Another is that there most definitely exist sparse correlations within the measurements between processes. This is due to many of the measurements done during the process line not really being measurements in the pure sense of the word, but rather machine settings that an engineer has chosen for that day or battery model recipe. We can explain the phenomenon with an example. Say we measure a strong correlation between oven temperature at process p_{i-1} and electrode sheet width at process p_i. This correlation might solely be due to a process setting change of the sheet cutting width at process p_i, rather than to the oven temperature at p_{i-1}.

Another likely explanation for the inconclusive results is that we need to go further back in the production line to capture the true signal. As is shown in [6], defects during electrode coating seem to play an important role with regards to battery cell quality. Moving forward, additional analysis will be done including data from electrode coating.

A final point of future work is to investigate general outlier detection in the production line. This could be done by multidimensional distribution estimation as is proposed here [9].

References

[1] Juan Carlos Antón, Paulino Jose Garcia Nieto, Cecilio Blanco, and J. Vilan. Support vector machines used to estimate the battery state of charge. *Power Electronics, IEEE Transactions on*, 28:5919–5926, 12 2013.

[2] Richard Armstrong. Should pearson's correlation coefficient be avoided? *Ophthalmic and Physiological Optics*, 39, 08 2019.

[3] Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Peter Deisenroth. Matérn gaussian processes on riemannian manifolds, 2020.

[4] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[5] Wen-Yeau Chang. The state of charge estimating methods for battery: A review. *ISRN Applied Mathematics*, 2013, 07 2013.

[6] Mohanti D et. al. Electrode coating defect analysis and processing nde for high-energy lithium-ion batteries. `energy`.

[7] Jerome Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29, 11 2000.

[8] Marc Genton. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 01 2001.

[9] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation, 2015.

[10] Frank E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[11] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 01 2002.

[12] Graeme Halford, Rosemary Baker, Julie Mccredden, and John Bain. How many variables can humans process? *Psychological science*, 16:70–6, 02 2005.

[13] Z. He, G. Yang, L. Lu, and H. Wu. Battery dc internal resistance test method based on the constant current external characteristics and soc. -, 55:532–537, 05 2015.

[14] Sungil Kim and Heeyoung Kim. A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, 32(3):669–679, 2016.

[15] Thomas Kornas, Edgar Knak, Rüdiger Daub, Ulrich Bührer, Christoph Lienemann, Heiner Heimes, Achim Kampker, Sebastian Thiede, and Christoph Herrmann. A multivariate kpi-based method for quality assurance in lithium-ion-battery production. *Procedia CIRP*, 81:75–80, 2019. 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.

[16] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[17] Gregory L.Plett. *Battery Management Systemts, Volume 1: Battery Modelling*. Artech House, 2015.

[18] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

[19] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3:861, 09 2018.

[20] Christoph Molnar. *Interpretable Machine Learning*. Lulu.com, 2020.

[21] Wai On. Visualizing ai. -, -, 2020.

[22] Nikolay Oskolkov. tsne vs. umap: Global structure. *Mathematical Statistics and Machine Learning for Life Sciences*, 15, 2020.

[23] Jong In Park, Seong Kim, and Myong Jeong. A new tolerance design method for a secondary rechargeable battery using design of experiments with mixture. *Quality and Reliability Eng. Int.*, 24:543–556, 08 2008.

[24] Kedar Potdar, Taher Pardawala, and Chinmay Pai. A comparative study of categorical variable encoding techniques for neural network classifiers. *International Journal of Computer Applications*, 175:7–9, 10 2017.

[25] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features, 2019.

[26] Linden D. Reddy T B. *Linden's handbook of batteries. 4th ed.* McGraw-Hill, 2011.

[27] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.

[28] Oded MaimonLior Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer, 2005.

[29] Darius Roman, Saurabh Saxena, Valentin Robu, Michael Pecht, and David Flynn. Machine learning pipeline for battery state of health estimation, 2021.

[30] GOZDE Sahinoglu, Milutin Pajovic, Zafer Sahinoglu, Yebin Wang, Philip Orlik, and Toshihiro Wada. Battery state of charge estimation based on regular/recurrent gaussian process regression. *IEEE Transactions on Industrial Electronics*, PP:1–1, 10 2017.

[31] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[32] Edward R. Tufte. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[33] Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

[34] Filip Vitéz. Methodology validation on a synthetic dataset. `https://github.com/filipvitez/Thesis-SOH-Dataset`, 2021.

[35] Stefan Wager, Trevor Hastie, and Bradley Efron. Confidence intervals for random forests: The jackknife and the infinitesimal jackknife, 2014.

[36] Robert E. Schapire Yoav Freund. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14:771–780, 1999.

[37] Taimoor Zahid, Kun Xu, Weimin Li, Chenming Li, and Hongzhe Li. State of charge estimation for electric vehicle power battery using advanced machine learning algorithm under diversified drive cycles. *Energy*, 162, 08 2018.

[38] Zhengming (John) Zhang and Premanand Ramadass. *Lithium-Ion Batterylithium-ion batterySystems and Technologylithium-ion batterytechnology*, pages 6122–6149. Springer New York, New York, NY, 2012.

[39] Yanyan Zhao, Oliver Pohl, Anand I. Bhatt, Gavin E. Collis, Peter J. Mahon, Thomas Rüther, and Anthony F. Hollenkamp. A review on battery market trends, second-life reuse, and recycling. *Sustainable Chemistry*, 2(1):167–205, 2021.

Appendices

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2021-06-03

EXAMENSARBETE Predicting Lithium-Ion Battery Cell Quality Indicators
STUDENT Filip Vitéz
HANDLEDARE Kenan Šehić (LTH), Marcus Ulmefors (Northvolt AB)
EXAMINATOR Elin Anna Topp (LTH)

Kan maskininlärning användas för att prediktera batterikvalitet?

POPULÄRVETENSKAPLIG SAMMANFATTNING **Filip Vitéz**

För att möta den ökade efterfrågan på batterier krävs en explosionsartad produktionsökning av battericeller. I detta arbete presenteras en metod för att med hjälp av produktionsdata prediktera kvalitetsindikatorer för battericeller som mäts vid slutet av produktionslinan.

När fler länder ger sig in i kampen för en grönare framtid ökar efterfrågan på konsumtion och lagring av elektricitet. Det har i sin tur lett till en explosionsartad efterfråga på litiumjonbattericeller som i dagsläget utgör den mest skalbara tekniken. Battericellstillverkning är komplicerad, och består av ett trettiotal delprocesser där mycket kan gå fel. För att säkerställa att de producerade cellerna håller önskad kvalitet, utförs traditionellt en uttömmande valideringsprocess i slutet av produktionslinan där ett antal kvalitetsindikatorer mäts. En ökad förståelse för hur dessa kvalitetsindikatorer hänger ihop med produktionsprocessen skulle ha flertalet positiva följdeffekter. En sådan kan vara att optimera produktionsparameterarna för att i sin tur möjliggöra en produktionsökning av celler med hög kvalitet.

I detta examensarbete har således en maskininlärningspipeline tagits fram för att med hjälp av produktionsdata från den senare delen av linan prediktera de kvalitetsindikatorer som mäts vid slutet av linan. Pipelinen jämför flertalet maskininlärningsmodeller och innehåller även en omfattande analys av vilka produktionsparametrar som verkar ha mest betydelse för att nå önskade kvalitétsnivåer.

Resultatet av forskningen visar att modellerna har hög träffsäkerhet för några av kvalitetsindikatorerna. Det är emellertid svårt att avgöra hur mycket av detta som beror på att modellerna faktiskt har lärt sig kopplingar mellan produktionsdatan och kvalitetsindikatorerna. Många av indikatorerna varierar väldigt lite cellerna emellan, och att bara gissa medelvärdet ger således förhållandevis hög träffsäkerhet.

En rimlig slutsats är att data från tidigare i produktionsprocessen behöver inkluderas för att nå bättre resultat. Pipelinen är byggd så att ny data enkelt kan läggas till, samt att fler maskininlärningsmodeller kan jämföras.