CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

EV battery degradation forecasting based on high dimensional data

Master's thesis in Computer science and engineering

PONGSAKORN CHANCHAIPOL

LEELAWADEE SIRIKUL

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

EV battery degradation forecasting based on high dimensional data

PONGSAKORN CHANCHAIPOL
LEELAWADEE SIRIKUL

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

PONGSAKORN CHANCHAIPOL
LEELAWADEE SIRIKUL

Supervisor: Ashkan Panahi, Data Science and AI division, Department of Computer Science and Engineering.
Advisor: Christian Fleischer, Volvo Cars
Examiner: Devdatt Dubhashi, Data Science and AI division, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2020

PONGSAKORN CHANCHAIPOL
LEELAWADEE SIRIKUL
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This study is made to investigate the benefit of using high-dimensional data from sensors in electric vehicles (EVs) in State-of-Health (SOH) forecasting. By the term "High dimensional data", it means the data has a considerably large dimension that the computation on the data is very difficult or very time consuming without a proper handling. EV is one of the biggest trends in the automobile industry in the past few years. One of the biggest concerns about the EVs is their lithium-ion battery. The lithium-ion batteries will degrade over time based on several factors such as calendar ageing, accumulated charge, temperature, and etc. These factors can lead the Li-ion batteries to degrade, which can lead to several problems for the EVs. The study is made on the Diagnostic Read Out (DRO) dataset from real-world Volvo Cars' customers. Several machine learning models were used to predict and analyzed to find the significant causes of battery degradation. Principal Component Analysis (PCA) was used to reduce the dimension of data before fitting it into the prediction models. For the results, this study found that Long Short-Term Memory (LSTM) is the most suitable machine learning model for SOH forecasting from all machine learning models considered in this research. Moreover, applying PCA to the data significantly improves the models' performances than using the whole data without dimensionality reduction. Furthermore, adding a suitable number of lag features to the input also increases the models' performance considerably. Last but not least, SOC and the temperature while cranking and starting the engine of the hybrid EVs are considered to be important to the battery degradation problem.

Keywords: Lithium-ion Battery degradation, Electric Vehicle, State-of-Health forecasting, Machine learning, Linear Regression, Recurrent Neural Network, Random Forest Regression, Long Short-Term Memory, Principal Component Analysis.

Acknowledgements

This project is one of the best opportunities for us to have a chance to work with Volvo Cars and Chalmers University of Technology. The project itself is very challenging and interesting at the same time. We would first like to thank our thesis advisor Christian Fleischer at Volvo Cars for giving us such an opportunity for this amazing project and always provided great answers to our questions regardless of how long they were. Many advice and support from him are some of the most important things that help us complete this project. A huge thank you to our supervisor Asst. Prof. Ashkan Panahi of the Department of Computer Science and Engineering at Chalmers for providing us with a lot of support and guidance on this thesis. We would also like to thank Herman Johnsson for helping us with the data preparation and several useful recommendations on the project. We also want to thank our examiner Prof. Devdatt Dubhashi and our thesis coordinator Birgit Grohe for all the support you gave us. Last but not least, we want to thank all the Volvo Cars employees, Chalmers staffs, and other master thesis students for being part of our study at Chalmers and Volvo Cars.

Pongsakorn Chanchaipol and Leelawadee Sirikul, Gothenburg, October 2020

Contents

List of Figures

List of Tables

1
Introduction

1.1 Overall

In these past few years, electric vehicles (EVs) have been one of the world hottest trends in the automotive industry. EVs replace the need for a combustion engine with single or multiple electric motors which yield better power efficiency [1] and ease of maintenance due to the lower number of complex components in the vehicle. Apart from the electric motor, another major difference of EVs from the regular combustion engine car is their energy source. EVs use batteries as their source of energy which can be charged up via any renewable energy source, even from the brake mechanism from the car itself. Moreover, any combustion engine car with its engine running will always consume energy even when the car is stationary. On the other hand, the electric motor in EV will consume energy only when the car is accelerating. This reason alone makes EV technically more power-efficient and environmentally friendly than most of the combustion engine cars on the market.

The lithium-ion battery is a device that stores electrical energy through electrochemical reactions to power other electrical appliances such as electric vehicles (EVs), smartphones, laptops, and etc. At Volvo Cars, lithium-ion batteries are one of the most essential components that makes the Plug-In Hybrids Electric Vehicles (PHEVs) and Battery Electric Vehicles (BEVs) come to life. One main problem of batteries is that they naturally degrade over time due to the uncontrollable chemical reactions of the components inside the battery. This results in lower capacity and also increases the chance of having battery problems such as overheating or even explosion in the worst possible case. In response, Volvo Cars employs several sensors in their EVs to monitor and help to analyze the battery degradation of each car, specifically State Of Health (SOH) or the measurement that reflects the current capacity of the battery. Unfortunately, it is very difficult to have an accurate and reliable estimation of the current battery condition due to the limited capacity of online calculation, the high complexity of models, and various kinds of uncertainties.

In the past few years, Machine Learning and Artificial Intelligence (AI) have proven to be two of the most powerful analytics tools in Computer Science. It can be used for analyzing customers' behavior, improving traffic management, increasing power efficiency of mobile phones, or even forecasting future such as stock market prices, weather, and etc [2, 3, 4]. Focusing on the future prediction, the data is collected in time-series format due to the ability to be represented in graphs/plots based on

the time axis. However, these kinds of future predictions are not always accurate or reliable. In fact, most of the time, they are only used as a guide for the experts to further analyze the target event due to the instability of long-term predictions.

In the recent literature on battery degradation, most of the researchers tend to analyze only battery-specific parameters such as its input/output voltage, current, and temperature due to the ease of data collection. This data can be considered to be low dimensional since not many parameters can be collected from just the batteries [5, 6, 7]. On the other hand, this thesis proposes a way to work alternatively on high dimensional data collected from all of the sensors in Volvo EVs. This high-dimensional sensor data may contain some useful information help us forecast the state of health of the battery in a more reliable way compared to the method using only low-dimensional battery-specific data.

The main contribution of this project is to build a machine learning solution to forecast the battery degradation of EVs, specifically the State of Health (SOH) parameter, based on high dimensional sensor data. The outcome of this thesis can be further utilized to help Volvo Cars draw a conclusion on the remaining useful life of the battery in their EVs, lower the risk of customers having battery problems, and also to help Volvo Cars find out the reasons behind these battery problems caused by the degradation of these batteries.

1.2 Research questions

Below is the list of research questions that this thesis aims to answer.

- Does the additional high-dimensional data from EV sensors improve battery degradation forecasting performance of the prediction models?
- Does the usage of the Principal Component Analysis (PCA) improve the performances of the prediction models compared to using the whole dataset features?
- Does the additional lag features improve State-of-Health forecasting performance of the machine learning models?
- What is the most suitable machine learning model for SOH forecasting based on DRO data?
- What are the possible causes of the battery degradation in EVs?

2
Background

2.1 Lithium-ion Battery

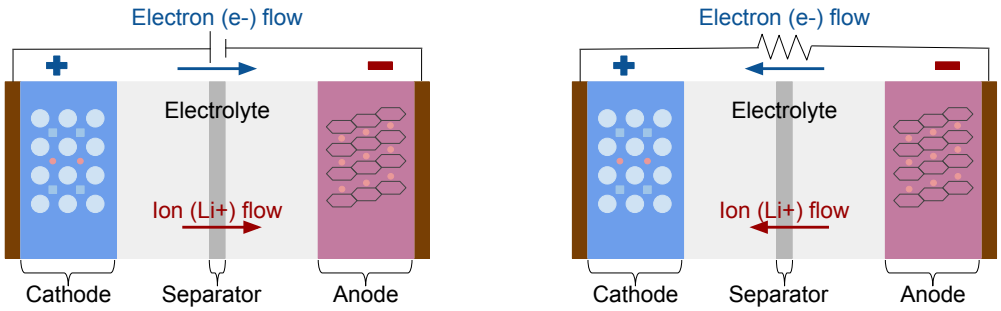
Lithium-ion batteries are one of the most commonly used energy storage devices for most of the portable electrical appliances such as smartphones, laptops, and etc. Due to their wide variety of applications, batteries can vary in size, shape, and capacity depending on the type of electrical appliances they were used for. Nevertheless, lithium-ion batteries still share the same basic structure and principle with other battery types. The below sections will explain the basic structure and the principal chemical reactions of the lithium-ion battery. Moreover, it also includes the explanation of battery degradation and some important parameters of the battery includes battery capacity, State-of-Charge (SOC), and State-of-Health (SOH)

2.1.1 Battery Structures

Lithium-ion batteries consist of 4 fundamental components including Cathode, Anode, Electrolyte, and Separator [8, 9].

The cathode is a battery component that is made from lithium-oxide. This lithium-oxide will act as active material that allows lithium to break into lithium-ions and electrons. For the anode, graphite is used as a stable structure to store lithium-ions when the battery is charged. While charging, electrons will break themselves off the lithium-oxide in the cathode and move toward the anode through the charging circuit. This induces the anode to be negatively charged which attracts the positively charged lithium-ions. In order for the lithium-ions to move from the cathode to anode, the electrolyte is used as a medium transferring the ions from one side to the other side internally but does not allow the electron to move through it. After the battery is fully charged and the charging circuit is removed, most of the lithium-ions and electrons are now residing in the anode side which creates the difference in voltage between cathode-anode. To prevent the cathode-anode from touching, the separator is inserted in between to act as a barrier. Preventing cathode and anode from touching is essential since the touching between the two can cause all of the electrons to instantaneously move from anode back to the cathode which can create electricity and heat. In the worst case, this could lead to a battery explosion.

The main principle of the lithium battery to produce electricity is the movement of electron. As mentioned before, after the battery is fully charged, the lithium-ion and electrons are trapped at the anode. Since the electrons are negative, they tend to travel back to the cathode which is positively charged. However, the electrolyte does not allow the electrons to pass through, so the electrons will need another way to go back to the cathode. In this state, when we connect an electrical appliance to the battery, the electrons are able to travel back to the cathode through the electrical appliance. This flow of the electron produces electric current for the electrical appliance until the battery runs out of energy. In order to recharge the battery, we just need to create the reverse flow of the electron by using a charging adapter that allows electrons to flow back from the cathode to the anode once again.



(a) Battery Charging **(b)** Battery Discharging

Figure 2.1: These figures show how lithium-ions and electrons flow in a lithium-ion battery during charging and discharging states.

2.1.2 Battery Parameters

2.1.2.1 Capacity

Battery capacity measures how much electric charge can be stored in a battery. The common unit of the battery capacity is Ampere-hours (Ah) which is a product of the possible output current multiplied by time. For example, 1 Ah means that the battery can release 1 A current at its specific voltage for one hour before it is out-of-charge. The equation of battery capacity is given by:

$$C = I \times t \tag{2.1}$$

where C is capacity (Ah), I is current (Ampere), and t is time (hours).

2.1.2.2 State of Charge

State of Charge (SOC) is the measurement that shows how much the energy is left in the battery [7]. It shows the percentage of the remaining releasable capacity over the rated capacity given by the manufacturer. For example, 60 % SOC for a 1000Ah battery means that the battery has 600Ah remaining capacity left. The equation of SOC is given by:

$$SOC_t = \frac{C_t}{C_{rated}} \times 100\% \tag{2.2}$$

where $C(t)$ is the remaining capacity of the battery (Ah), C_{rated} is the rated capacity given by the manufacturer (Ah).

2.1.2.3 State of Health

State of Health (SOH) is one of the most important parameters to measure the ability to store and release the energy or power of a battery compared with a new battery from the manufacturer [7]. SOH can be defined into two specific forms: SOH_e and SOH_p, which are computed based on the degradation in terms of energy and power, respectively [10].

SOH_e is defined as the maximal current capacity of the battery ($C_{max,t}$) divided by the total capacity of a new battery as specified by the manufacturer (C_{rated}), which its equation is given by:

$$SOH_{e,t} = \frac{C_{max,t}}{C_{rated}} \times 100\% \tag{2.3}$$

For SOH_p, it is defined as the current ohmic resistance ($R_{o,t}$) divided by the ohmic resistance specified by the manufacturer ($R_{o,rated}$). The equation is given by:

$$SOH_{p,t} = \frac{R_{o,t}}{R_{o,rated}} \times 100\% \tag{2.4}$$

2.1.3 Battery degradation

Battery degradation is an inevitable chemical phenomenon that continuously reduces battery performance over time. Not only does it reduce the battery capacity, but also increases the resistance of the battery which can lead to higher operating temperatures and lower output current. After a certain amount of time, any degraded lithium-ion battery will not be able to keep its performance up enough to run its system reliably. So, EV batteries need to be checked during the car maintenance to make it run perfectly and prevent any unintentional problems caused by the degraded batteries.

The amount of battery degradation depends on several factors such as battery application, usage, temperature, age, and etc. From [7, 11], the causes of lithium-ion battery degradation can be summarized to 5 main factors; time, high temperature,

high/low state-of-charge, high current input/output, and energy cycles. Firstly, battery degradation will still occur whether the battery is in-used or not. Even if a customer did not use his EV for a year, the battery of that EV will still degrade over time. Another obvious factor is temperature. The extremely hot or cold temperature can affect battery performance and even degrade them significantly. Other than the temperature and age of the battery, fully charging or fully discharging batteries will also degrade them too. Moreover, fast-charging or fast-draining can cause the chemical components inside the battery to change quickly. This drastic change can increase the battery temperature and degrade the battery even more. Lastly, the energy usage cycle is the number of SOC cycles that the battery has been used. The more a battery is used, the higher its energy usage cycle is. So, a heavily-used battery is likely to be degraded more than a lightly-used one.

2.2 Related research

From the literature [12, 13, 14], for approximately 500 cycles or equivalently 50000 percent of accumulated charge, the State-of-Health degradation can be considered linear. In other words, at the beginning to middle lifetime of the battery, the battery is expected to degrade in a linear trend. Moreover, in [13], several battery degradation plots are shown on multiple Li-ion battery variations that the degradation trends of these batteries can be linear up to a very high precision for 800 charging cycles, which is considered to be quite old for battery in real-world usage. These reasons support the benefit of using linear prediction models such as univariate or multivariate linear regressions up to a shallow artificial neural network structure [15, 16] to reserve the prediction trend as close to a linear line. Therefore, the prediction model does not need to be very complex since the trends of battery degradation are considerably linear especially in the beginning to a middle lifetime of the battery.

Another concern for battery degradation forecasting is that long-term prediction is hardly accurate due to the unpredictable degradation pattern of the battery. From [17], the ceiling of accurate SOH forecasting is approximately only 1 year ahead. From our consideration, a one-year ahead prediction should be enough for Volvo Cars to prevent the battery problem to occur to the customer since every car is expected to be checked and repaired at the maintenance center at least once a year.

In the research conducted by Yang [18], the writers directly compared the performance between the LSTM network and the unscented Kalman filter (UKF) [19] on battery SOC estimation. The results of this paper show that their LSTM network significantly outperformed UKF by comparing their root mean squared error (RMSE) and mean absolute error (MAE) on several battery datasets. From their SOC estimation result, we found that their SOC decreasing patterns are very similar to SOH, that is they are both very close to the linear line with slight fluctuations every now and then. Their LSTM network structure also consisted of only three hidden layers which can also be considered as a shallow neural network as well. This also supports the idea of using only shallow neural networks for any time series sequences that are quite linear.

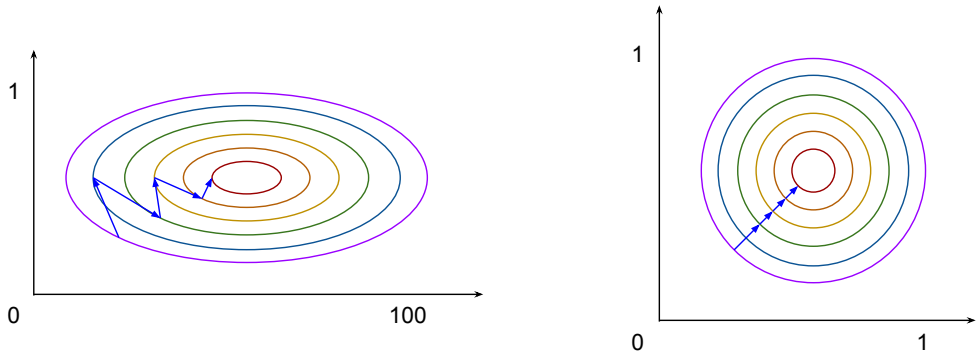
In the literature [20], the authors compared the performance between Autoregressive Integrated Moving Average (ARIMA) [21] and Long Short-term Memory (LSTM) based on their financial time-series data. The research found that their LSTM significantly outperformed the ARIMA model. Furthermore, LSTM appeared to capture the fluctuation of the patterns better than ARIMA and its variations too. So, our research will focus mainly on comparing the performances of several machine learning models including Linear Regression, Random Forest Regression, RNN, and LSTM to see which machine learning model performs the best on our dataset.

3

Theory

3.1 Standardization (Z-score Normalization)

Standardization or Z-score Normalization is an important tool for manipulating numerical features. Standardization was used to scale all numerical features so that their means are shifted to zeros (zero-means) and all have unit standard deviations [22]. This normalization technique will help the prediction models to train faster due to easier gradient descent. See Figure 3.1 for the visualization of how normalization can improve the model training process.



(a) Before normalization (imbalanced axes)

(b) After normalization (balanced axes)

Figure 3.1: An example visualization of gradient descent patterns comparing before and after applying normalization. In the figure (a), it is harder to find a path to the local minimum (center) due to the imbalanced axes (size 1:100). In the figure (b), after applying normalization, the pattern is well-balanced on both x and y-axes (size 1:1). Therefore, it is easier to find the local minimum.

The equation of the standardization is given by:

$$x' = \frac{(x - \bar{x})}{\sigma} \tag{3.1}$$

where x is the original feature, x' is the scaled feature, $\bar{x}$ is the mean of the feature, and σ is the standard deviation of the feature.

The sample mean is given by:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{3.2}$$

where x_i is the i^{th} value of variable in the dataset, n is the number of variables in the dataset.

Standard deviation is estimated using the formula given below:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{3.3}$$

3.2 Principal Component Analysis (PCA)

Principal component analysis (PCA) is a dimensionality reduction technique that transforms data to a set of linearly uncorrelated features called "Principal components" [22, 23]. The purpose of any dimensionality reduction technique is to reduce as much dimensionality from the data as possible, but also reserve as much information of the original data as possible [24]. The main point of doing this is to reduce the curse of dimensionality from the high dimensional dataset which is very problematic for any statistical models for prediction and machine learning models to learn or capture the pattern from these high dimensional data.

The main concept of PCA is to transform the data into a new coordinate system that maximizes the variance of each principal component on the new coordination system. In other words, PCA is a technique that rotates the data around and finds the viewing angle that maximizes the variance of the data, so we can see the pattern of the data from the best angle and see the distribution of the data clearer. By rotating data, it means to linearly transform the data to a new coordinate system based on eigenvalues of the principal components that are computed to maximize the variance of the transformed data. Each principal component is a linear combination between the data features, weighted by the eigenvectors of the data covariance matrix. For example, principal component 1 (PC1) can be represented like this:

$$PC_1 = w_{11}X_1 + w_{12}X_2 + w_{13}X_3 + \ldots + w_{1p}X_p \tag{3.4}$$

where $[w_{11}, w_{12}, \ldots, w_{1p}]$ is the weight or eivectors of PC_1 and $[X_1, X_2, ..., X_p]$ are the features that represent the data

The equation (3.4) of PC1 represents the linear combination between p features that maximize the variation which can be variance or sum of squared distances between each data point and the eigenvector that represents PC1. Therefore, each principal component will have its own eigenvector and eigenvalues. Moreover, these eigenvectors must be orthogonal with each other so that the transformed data are linearly uncorrelated to each other. To compute the eigenvector and eigenvalues of each principal component, we first start by computing the $p \times p$ covariance matrix

from the p features of the data. The equation for the covariance between features X and Y is defined as in equation (3.5):

$$Cov(\mathbf{X}, \mathbf{Y}) = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{X})}{N} \tag{3.5}$$

where $Cov(\mathbf{X}, \mathbf{Y})$ is the covariance of the feature X and Y,
N is the number of data points,
X_i and Y_i are the i^{th} items of each feature X and Y,
$\bar{X}$ and $\bar{Y}$ are the means of each feature X and Y

After the covariance matrix is computed, we can use it to compute the eigenvalues and eigenvectors of each principal component which we will not go into the detail since there will be too much information here. Then, after calculating all of the eigenvalues and eigenvectors of each principal component, we can now use them to transform our data.

Applying PCA, we will get new data that represents the original data but on different coordination due to the transformation and every feature is now linearly uncorrelated. At this point, we have already got p principal components from the new data to represent the old data. Now, we can sort these principal components of the new data by its eigenvalues and choose only p' highest-eigenvalues principal components, where $p' < p$, to reduce the dimensionality. Now, we will have a lower-dimensional data while reserving as much information as possible by using PCA.

3.3 Machine Learning Models

This section describes the machine learning models that were used to capture the pattern of input data and predict the future State-of-Health in detail. It includes *Linear Regression*, *Random Forest Regression*, *Recurrent Neural Network*, and *Long Short-Term Memory*.

3.3.1 Linear Regression

Linear Regression is one of the most basic prediction model in machine learning. The algorithm is based on supervised learning, where it is trained on the known pairs between features X and target y. Normally, a simple linear regression model receives only one input feature x to predict one output value $\hat{y}$. The goal of this simple linear regression is to find the relationships between them by fitting the linear equation (3.6) to a given input data.

$$\hat{y} = mx + c \tag{3.6}$$

where $\hat{y}$ is an estimated dependent variable value. x is an independent variable value or input feature. m is a coefficient that shows the slope of the line. c is the Y intercept (the value of $\hat{y}$ when $x = 0$).

In the case of multivariate linear regression or linear regression model that receives more than one input feature, the model is based on the given linear combination equation (3.7)

$$\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_{p'} x_{ip'} \tag{3.7}$$

where i is the index of the observation X_i, and p' in the number of input features. $x_{i1}, ..., x_{ip'}$ is input features. $\hat{y}_i$ is the prediction target, and $\beta_0, ..., \beta_{p'}$ is the coefficients of the model.

The main objective of the linear regression model is to find a line (or hyper-plane for the multivariate version) that fits the data the best. In this case, it means that the algorithm needs to find weights or the coefficients $\beta_0, .., \beta_{p'}$ that minimize the error between the prediction value $\hat{y}_i$ and the actual target values y_i. Finding these coefficients is usually done by using gradient descent.

This linear regression model is very simple yet reasonably effective for any regression problem that the data has linear pattern or at least close to a linear pattern. So, in this thesis, we used both simple and multivariate linear regression models as our baseline models for comparing the performance between these simple models and the more complex models that will be explained in the next section.

3.3.2 Random Forest Regression

Random forest regression is an ensemble learning technique that uses multiple decision trees to help increase prediction performance [25]. The model was called "Forest" basically because it can contain a lot of decision trees to help to predict the result. This model was considered to be one of the most highly used machine learning models due to its simplicity and strength when applied correctly.

The key concept of the random forest model is that the decision trees inside must be trained to be considerably uncorrelated within each other. Normally, a regular decision tree is prone to overfitting very easily. For this reason, the decision trees inside the random forest model must be diverse enough so that they do not vote for the same result, caused by overfitting to the same set of data. By using a technique called "Bagging" or "Bootstrap Aggregation", each decision tree inside a random

forest model will be trained on a different subset of data that increases the diversity of the decision trees in the random forest model [26]. Another technique that could be applied to improve the diversity of these decision trees in the random forest model is to also apply the Bagging method to the set of features. Training on a different subset of features will also increase the diversity of these trees inside a random forest. Furthermore, this bagging method can also be used in the node growing process of each tree to further increase node diversity and also reduce overfitting of each tree. By doing this, the decision tree can be grown to its fullest and need no pruning after that. ("Pruning" is a technique to reduce overfitting in decision trees by removing sub-trees out of the decision tree based on the impact of that sub-trees.) Finally, after these decision trees in the ensemble were created, trained to the fullest, and fed an input for a prediction, their output results from all trees will be combined by averaging.

3.3.3 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) is a class of neural networks in deep learning which is designed specifically for capturing patterns of sequential data e.g. time-series data, natural language processing, and etc [27, 28, 29]. The concept of the RNN that distinguishes itself from a regular neural network is its hidden state that memorizes the information of the past data in the same sequence. The Figure 3.2 shows the RNN internal structure that is used in this thesis.

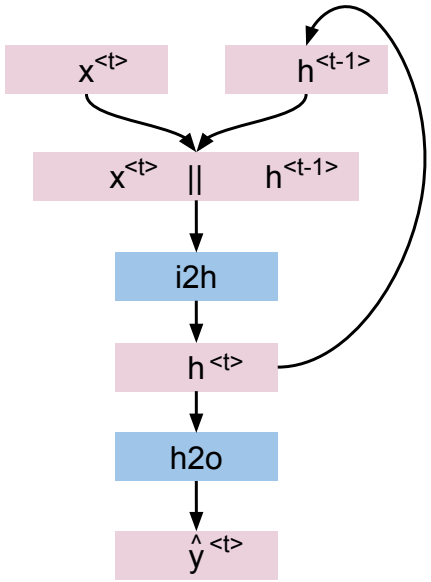


Figure 3.2: The visualization of RNN internal structure.
(Note: '||' refers to concatenation between 2 blocks of tensors)

At the start of every input sequence, we initialize the hidden state $h^{<t-1>}$ for memorizing sequential data to be all zeros (clear its memory of the previous data sequence). Then, the first item in the input sequence is concatenated with the hidden state before it is fed into the input-to-hidden (i2h) network and an activation function g_1. After that, the output hidden state $h^{<t>}$ of the i2h network is fed into the hidden-to-output (h2o) network and an activation function g_2. Moreover, the hidden state $h^{<t>}$ will be passed to the next timestamp for the next prediction too. At this point, the output of the current timestamp $\hat{y}^{<t>}$ is generated. Therefore, the model is ready to move to the next item in the sequence and keep predicting one by one until the end of the sequence. Figure 3.3 is shown to visualize on how our RNN structure receives input and predicts output for each timestamp until it finishes a sequence:

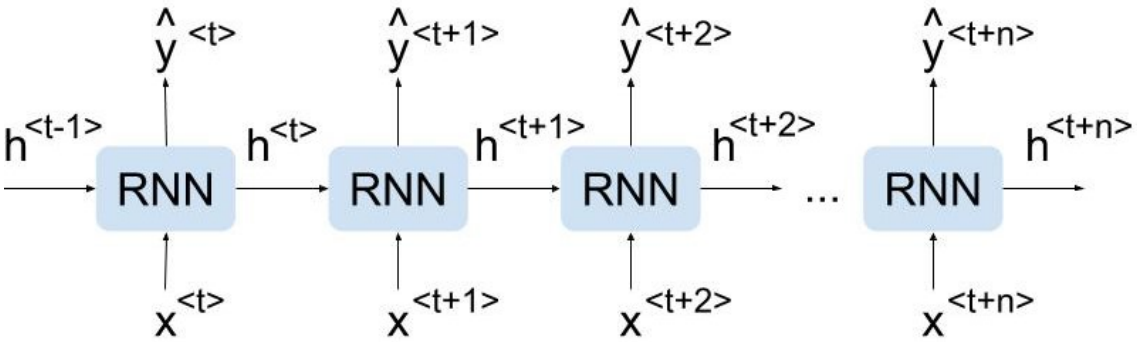


Figure 3.3: RNN feed-forward architecture.

The hidden and output layers can be computed as described in the equations (3.8) and (3.9) respectively:

$$h^{<t>} = g_1(W_{hh}h^{<t-1>} + W_{hx}x^{<t>} + b_h) \tag{3.8}$$
$$\hat{y}^{<t>} = g_2(W_{yh}h^{<t>} + b_y) \tag{3.9}$$

where $x^{<t>}$ is the input state at time t, $h^{<t>}$ is the hidden state at time t, $\hat{y}^{<t>}$ is the prediction result at time t, g_1, g_2 are the activation functions, W_{xh}, W_{hh}, b_h are the weight and bias coefficients of the i2h network that are shared temporally, W_{hy}, b_y are the weight and bias coefficients of the h2o network that are also shared temporally, $< t-1 >, < t >, < t+1 >, ..., < t+n >$ refer to the previous timestamp $< t-1 >$, the current timestamp $< t >$, the next timestamp $< t+1 >$, and the next n^{th} timestamp $< t+n >$, respectively.

At the end of each sequence, in order to start the next one, the hidden state needs to be reset to zeros again to clear the memory of the finished sequence and get ready to memorize the next one.

3.3.4 Long Short-Term Memory (LSTM)

Long Short-term Memory neural network or LSTM is one of the variations of Recurrent Neural Network (RNN) with the ability to learn long-term dependencies better than the regular RNN [30, 31]. The key components of LSTM that allow the model to learn long-term dependencies are its memory cell (c_t), forget gates (f_t) and input gates (i_t). These gates allow LSTM to adjust what to remember and what to forget that replicate how human memory work. These gates also control the information flow that means the gradients will be trapped in the memory cells longer than the regular RNN. So, this can reduce and/or prevent vanishing gradient problem that is one of the biggest problems of the regular RNN.

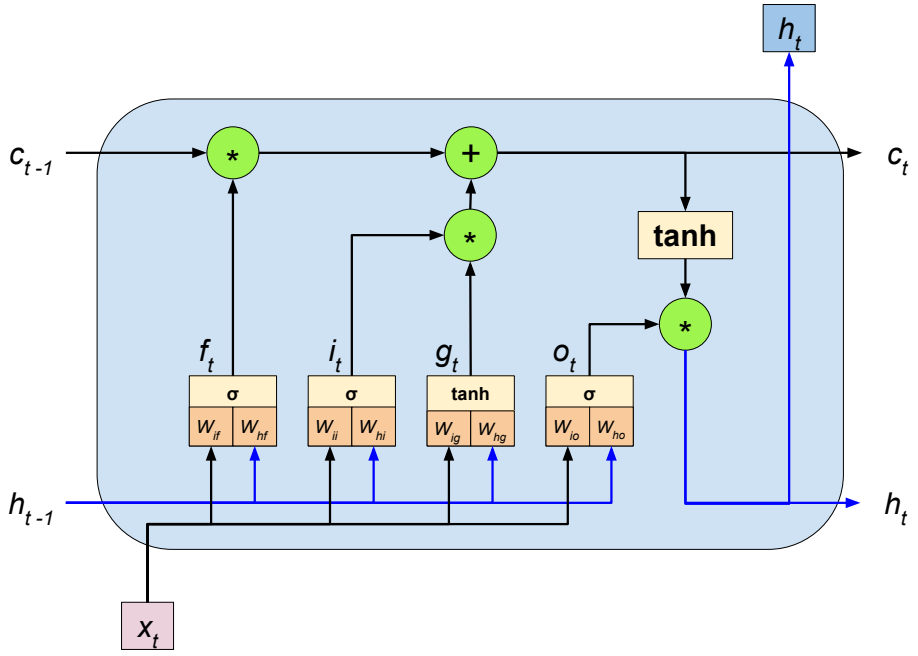


Figure 3.4: The structure of the Long Short-Term Memory (LSTM) neural network. Inputs: Current input (x_t), Memory from last LSTM unit (c_{t-1}), Output of last LSTM unit (h_{t-1}). Outputs: New updated memory (c_t), Current output (h_t). Operations: Element-wise product ($*$), Addition ($+$).

The equation of LSTM components is given by:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

where
i_t is the input gate's activation vector at time t.
f_t is the forget gate's activation vector at time t, if the values (f_t) close to zero, the cell will forget the cell state from the past (c_{t-1}), but if the value close to one, the cell will remember its history.
g_t is the cell gate's activation vector at time t.
o_t is the output gate's activation vector at time t.
c_t is the cell state vector or memory at time t.
h_t is the hidden state vector at time t and also is the output vector. h_{t-1} is the hidden state vector from the previous timestamp $t-1$ and also is the input vector.
x_t is the input vector at time t.
All W are the weights/coefficients of each component.
All b are the biases of each component.
$\odot$ is the Hadamard product or element-wise product.
σ is the sigmoid function.
tanh is the hyperbolic tangent function.

To summarize the purpose of all these components, LSTM has in total of 4 gates which are the input gate i_t, the cell gate g_t, the forget gate f_t, and the output gate o_t. The input gate i_t controls how much information we should consider from input. The forget gate f_t controls how much information we should forget from the previous cell state c_{t-1} (memory state). The cell state c_t combines the information of the processed input vector g_t and the previous cell state c_{t-1} through the i_t and f_t gates. Finally, the hidden state h_t is computed by calculating the element-wise product between the output gate o_t and the $tanh(c_t)$.

3.4 Backpropagation Through Time (BPTT)

Backpropagation Through Time (BPTT) is made specifically for the machine learning models that receive sequential or temporal data as input, such as RNN and LSTM [29, 32]. The main point of backpropagation is to calculate gradients to update the weights of the model in order to reduce the loss of the model outputs compared to the real outputs for each epoch during the training process. In the prediction of the model that uses sequential data, the information of every input state is passed to the next one over and over. Suppose that we feed an RNN a sequence of data, the first state of that sequence will be used to predict the output of that timestamp. Then, the model passes through its hidden state from the first timestamp to the next one and it will keep happening over and over until the model finished the whole sequence of input data. Therefore, the information of the first state was passed through the whole sequence via the hidden state. So, when we compute the backpropagation of the last output, it will calculate through the whole sequence due to the dependency from the hidden state that was passed since the start of the sequence. That is why backpropagation for these temporal or sequential models is called backpropagation through time. See Figure 3.5 for an example on how BPTT backtracked from $L^{<t+n>}$ to all of the previous input states of the same sequence.

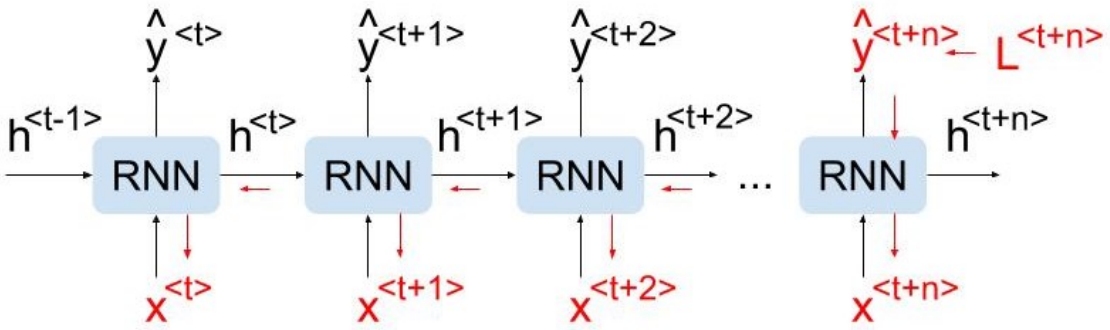


Figure 3.5: A BPTT example on the loss L^{t+n} of an RNN. In this case, the algorithm needs to backpropagate from the loss back to all input states of the same sequence including the first input state $x^{<t>}$.

4
Methods

This chapter will explain the data description and preprocessing techniques in detail. The implementations of the machine learning models will also be shown in this chapter.

4.1 Data Description

We used a data set called Diagnostic Read Out (DRO) data. The DRO data consists of tabular data collected from the sensors in the Volvo XC90 II model across 44 countries around the world. There are a total of 20,417 measurements (rows) from 7,247 cars with 253 features before feature selection. After feature selection by hand, we get a total of 237 usable features that are meaningful or relevant in some way to the battery degradation. For the sequential data of each car, the number of measurements can range from 2 to more than 60 measurements depending on each car. Mostly, the cars have only around 2-6 measurements in the period of around 1,600 days starting from their production date. So, these time gaps between each measurement can range from multiple days to multiple years depending on how frequently each customer brought their cars to a service and repair shop. Therefore, most of the cars will have long time gaps between each measurement which is typically around 180-500 days. In this study, the data were selected only from the cars that have more than 3 measurements so that we can have at least 3 sequential data points for the RNN and LSTM to learn the data patterns. For this reason, there are a total of 3,329 cars to be analyzed (with a total of 12,581 measurements).

Figure 4.1 and 4.2 indicates the scatter plot and the age distribution of the DRO data. One can clearly see that there are a lot of age gaps between these distributions, which is one big flaw of this data set. As mentioned in the previous paragraph, the DRO data was collected from customers' cars during the maintenance period of the car. A customer may bring his car to the shop once or twice a year depending on their usage. Therefore, in some age ranges, there are no customers that bring their cars to the shop at all. For this reason, these age gaps occurring on this dataset are unavoidable. This will make the sequence of data very short and has a low sampling rate (due to the gap). In this case, we think that the machine learning models for sequential data like RNN and LSTM might not benefit much from the temporal correlation of the data than other machine learning models like Random Forest due to the gap and the short sequence of the data. So, the performance of several machine learning models will be analyzed regarding this problem in the latter section of this

thesis.

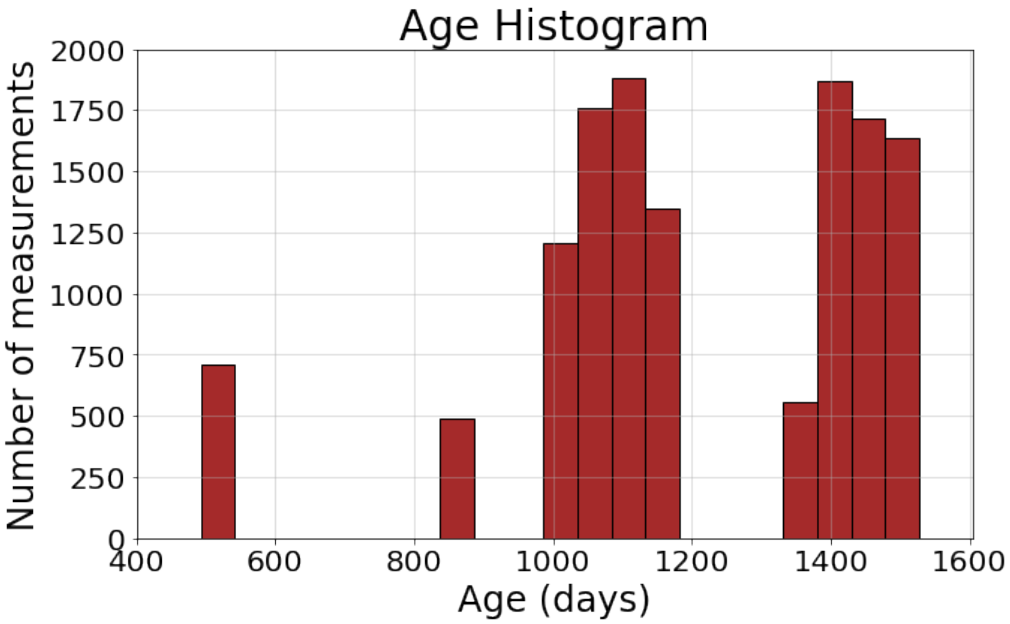


Figure 4.1: Age Distribution of DRO data

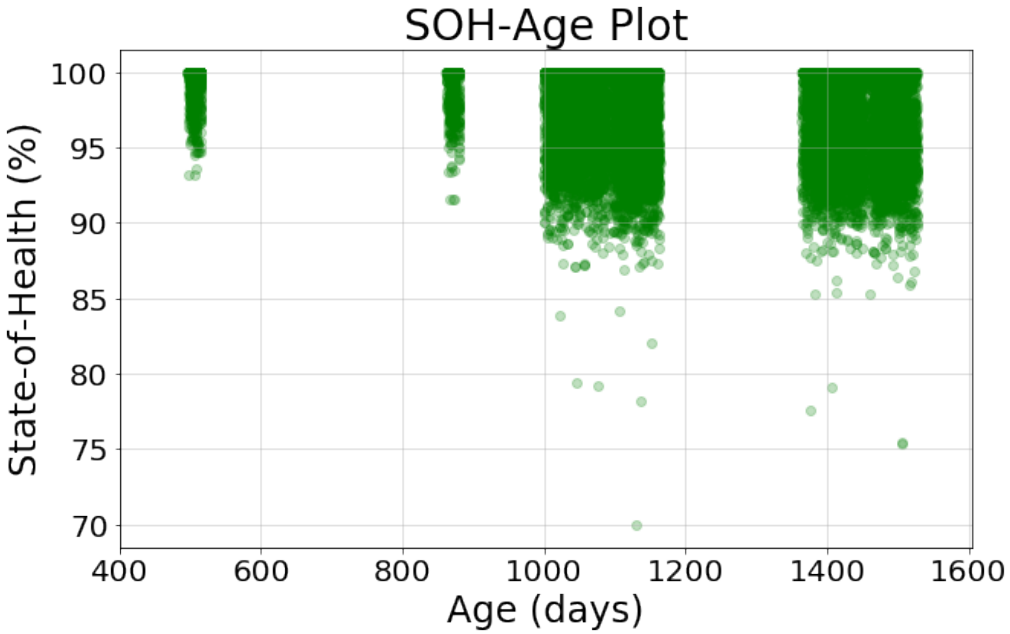


Figure 4.2: State-of-Health and Age scatter plot

4.2 Data Preprocessing

This section describes how the data is preprocessed before it is fed into the prediction models.

4.2.1 Applying Standardization and PCA

First of all, we applied the Standard Scaler to every feature in the dataset. After we normalizing the data using the Standard Scaler, we applied PCA to reduce the dimensionality of the data from 237 features down to some reasonable sizes for the prediction models. In our case, we separated the features of DRO data into 2 types; *basic features* and *histogram features*. The *basic features* are the features that are basic measurements for most of the vehicles and we considered them to be directly related to battery usage including the age of the vehicle (days), mileage, and accumulated charge. The *histogram features* are the features that are measured based on some specific vehicle events. These features are called *histogram features* because they are originally meant to be visualized into histogram plots, so they are quite high in dimension because they have to represent themselves in multiple ranges/intervals such as the number of times that the vehicle was started in some temperature ranges, maximum cell voltage output in some temperature ranges, and etc. Therefore, we decided to apply PCA only to the histogram features which are considerably high-dimensional and quite repetitive. Then, leave the low-dimensional basic features as it is since they might be useful later when we analyze our prediction models based on these non-PCA features. After applying PCA to the histogram features, we visualize their variances out as a scree plot to see how their variances distribute. Here is a sample scree plot of 50 principal components sorted by their variances.

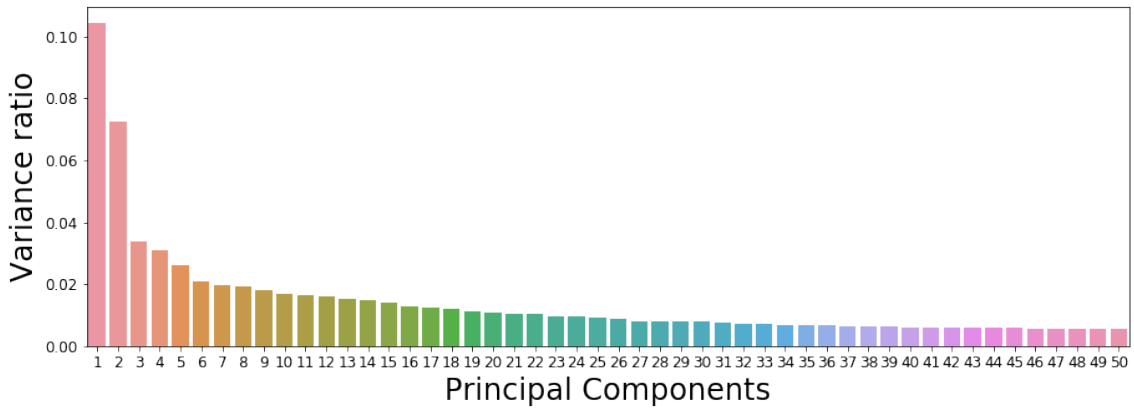


Figure 4.3: Scree plot of top 50 principal components from the *histogram features*. We can clearly see that there is an "elbow" or aggressive angle occur between PC2 and PC3.

From the scree plot in figure 4.3, we saw that there is an aggressive angle or the “elbow” occurred between PC2 and PC3. So, we considered PC1 and PC2 to be our two most significant principal components out of all principal components since the

variance ratio is significantly higher than the other principal components. Therefore, we decided to use mainly only PC1 and PC2 computed from the *histogram features* as additional features to the *basic features* that we skipped applying PCA previously.

To summarize this PCA section, we applied PCA to the histogram features and chose mainly only the PC1 and PC2 as the substitution of the whole *histogram features* due to the elbow point on the scree plot. Therefore, our main set of features after combining with the PCA result are the combination of the *basic features*, PC1, and PC2. The other principal components will be added based on our consideration during the hyperparameters-tuning step and the result will be analyzed further to find the best set of features for each prediction model.

4.2.2 Adding lag features to the input structure

From the previous part, we have already applied PCA to the histogram features and combined them with the basic features. In this section, we will restructure so that they are ready to be used to train our prediction models. The equation (4.1) is a sample designed input state from a measurement sequence of a car,

$$input\_state^{<t>} = X_1^{<t>}, X_2^{<t>}, \ldots, X_{p'}^{<t>}, X_1^{<t-1>}, X_2^{<t-1>}, ..., X_{p'}^{<t-1>} \tag{4.1}$$

where $< t >$ is the current timestamp, $X_i^{<t>}$ means the i_{th} feature at the current timestamp $< t >$, and $X_i^{<t-1>}$ means the i_{th} feature at the previous timestamp.
We design the input structure to be like this so that the prediction models can predict a result from the previous timestamp $< t-1 >$.

This structure design is expected to be especially useful for the prediction models that do not have the hidden state for sequential data, such as Linear Regression and Random Forest model in our case. This structure should still be useful for RNN and LSTM in the way that the models will focus more on the previous measurement than the further measurement in the past that they have to recall from their hidden (or memory) states.

4.2.3 Adding the Next features

The next feature is the name for a variable that contains the data from the next time steps. The purpose of this project is to predict the next SOH, not the current SOH. So, the next SOH is referred to as a dependent feature. Before predicting the next SOH, It is very important to determine a time period or the next age (next time that customer will go to the customer service center). Therefore, the next age is added into input structure and the next SOH is the target value.

4.2.4 Train/Test/Validation split

We split our dataset by different cars with an 80:10:10 ratio for training, validation, and test sets, respectively. However, splitting them randomly from the whole dataset might ruin the balance of the data between each continent. The cars from each continent might have different patterns of data due to several reasons such as different traffic conditions, climates, temperatures, cultures, and etc. Hence we decided to split the data for each continent individually. We expect this data splitting method to reserve the ratio of complex different patterns causing by reserving the ratio of data from each continent. The number of cars and measurements between each dataset will be changed due to the randomness of the splitting algorithm but the ratio between each continent will be reserved and the ratio between training, validation, and test sets is always approximately 80:10:10.

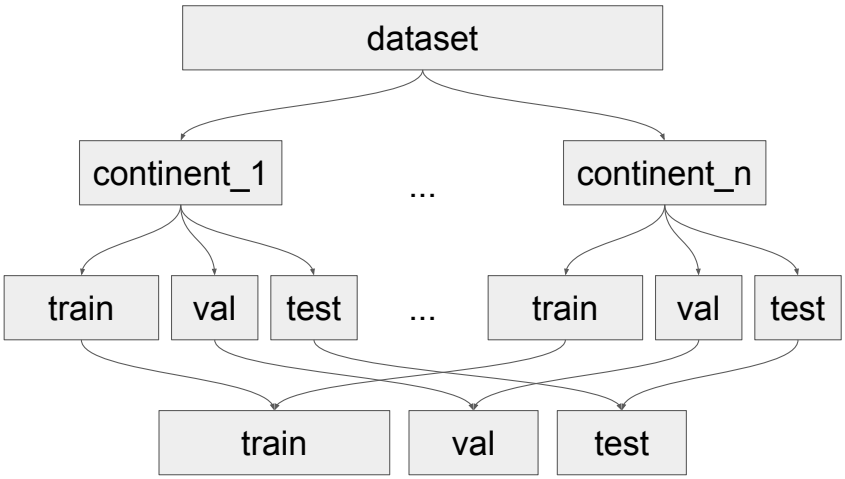


Figure 4.4: The visualization of how the dataset was split by its continental information into training, validation, and test sets.

4.3 Machine Learning Model Implementations

In this section, the implementations of the selected machine learning models will be explained. It is worth mentioning that every model except the simple linear regression model was trained on several sets of input features including basic features + none/2PC/10PC/50PC/Histogram features + none/lag features + target age.

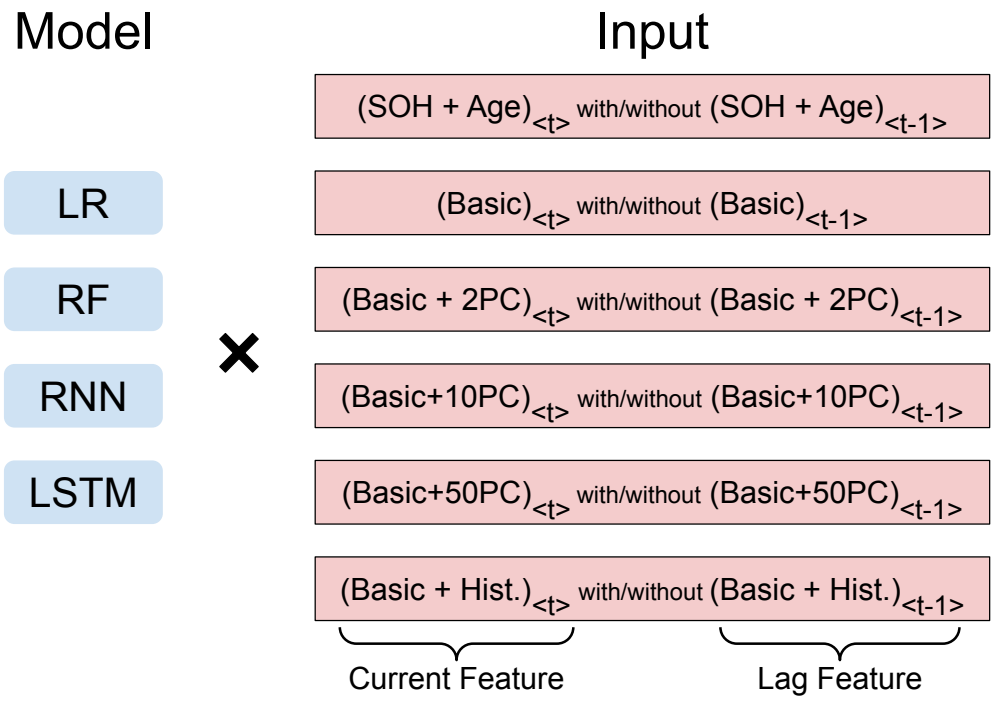


Figure 4.5: Input feature design for the SOH prediction models. Models: Linear Regression (LR), Random Forest Regression (RF), Recurrent Neural Network (RNN), Long Short-term Memory (LSTM). Features: State-of-Health (SOH), Age, Basic features (Basic), Histogram features (Hist.), while $< t >$ refers to the current timestamp and $< t - 1 >$ refers to the previous timestamp (lag features).

4.3.1 Simple Linear Regression implementation

This simple linear regression model is basically a univariate linear regression model that receives age as input and predicts SOH corresponding to the given age. As mentioned before, the equation of this simple linear regression model is given by the equation 3.6. The purpose of this model is a baseline performance for the other model, which shows how much performance can be improved by adding more input features and using higher complexity models. This model was implemented by using the linear regression model from the Scikit-learn library with age as its only input feature. After that, the model was trained using the *Fit* function given by Scikit-learn to find the slope that fits the training data the best.

4.3.2 Linear Regression implementation

This linear regression model was created in the same way as the previous simple linear regression model, that is to use the model from the Scikit-learn library [33] but more features were added. So, it is basically a multivariate linear regression model that uses the linear combination equation (3.7). The purpose of this multivariate linear regression model is to be used as another performance baseline for the performance comparison and to judge the complexity of the problem based on the effectiveness of this linear regression model. Several sets of input features were used to train the model for performance comparison. Moreover, its weights or coefficients will be analyzed to see how these features (e.g. age, mileage, temperatures) are related to SOH based on the model's weight optimizing decision.

4.3.3 Random Forest Regression implementation

For the Random Forest Regression model, we also used the model implemented from the Scikit-learn library [33]. Several Random Forest Regression models were created with several numbers of sub-trees for performance comparison with bootstrap enabled, all sub-trees are constructed from the whole set of input features until fully-grown, and unlimited maximum depth for sub-trees. The other model parameters are left as the default setting given by the Scikit-learn library. For the output calculation, the average of the prediction results from all sub-trees is computed and returned as the output of the Random Forest Regression model.

For the suitable number of trees, we constructed an experiment to observe which number of trees will give us the best result regarding the increase of model complexity and time usage. This experiment was done by training several random forest models with different numbers of trees from 1,2,4,...,8192, and 16384 trees on the validation set to observe which one gives the lowest Mean Square Error without increasing too many trees.

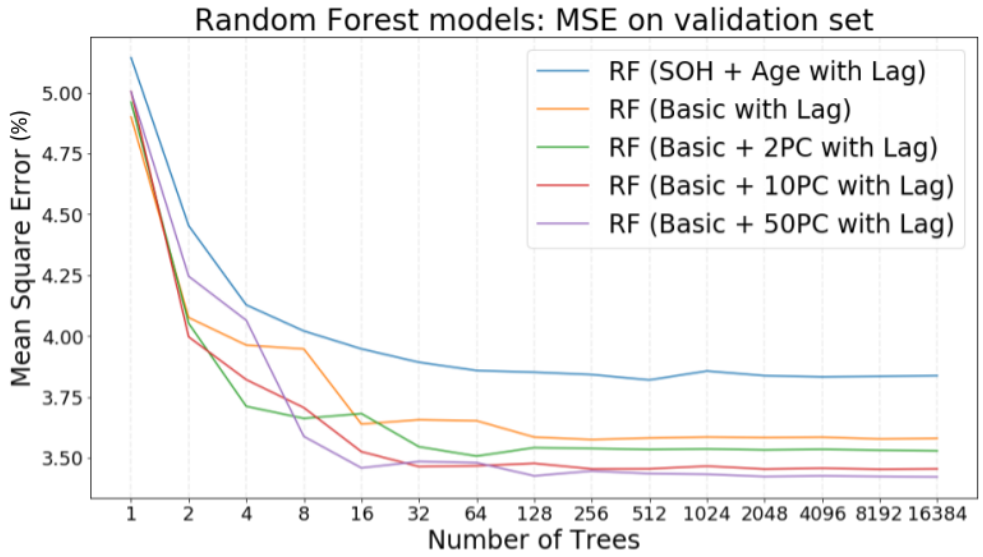


Figure 4.6: MSE of the Random Forest models on the validation set is displayed as a function of the number of trees from Random Forest Models.

From figure 4.6, it shows that after increasing the number of trees to over 512, the MSE does not reduce much for every input variations. For this reason, 512 trees seem to be the suitable number of trees for the random forest regression model. Therefore, 512 trees were applied for every Random Forest Regression models in this thesis.

4.3.4 Recurrent Neural Network implementation

The Recurrent Neural Network (RNN) was built using the Pytorch library [34]. The architecture is designed based on the equations explained in the Theory section (subsection 3.3.3). Several of RNN models were created for several sets of input features with the hidden layer size equals to the input size. For instance, RNN with input size of n features will be created with a hidden layer with size n for memorizing the equal amount of information from the sequential data of the previous timestamp. Backpropagation Through Time (BPTT) was used to update all of the coefficients inside the RNN model and train the model to fit the given training data. Adam optimizer is used here with a learning rate of 0.0002 and MSELoss as a criterion. Early stopping was also applied to prevent the model from overfitting too much on the training data by stopping the training when the validation loss is higher than the best one for more than 10 epochs.

For the training process of RNN (and LSTM), the data was first separated into 2,660 training data sequences based on the Vehicle IDs of the training set. Then, each data sequence was fed to the training algorithm one by one for the model to learn. At the end of each sequence, the data gradients will be computed and update the model based on the BPTT algorithm.

4.3.5 Long Short-Term Memory implementation

For the LSTM, the model structure was written in Python using the Pytorch library [34]. The exact structure of LSTM can be seen in the Theory section (subsection 3.3.4). The hidden layer size of LSTM was chosen to be equal to the input size, the same way with RNN on the previous sub-section. Backpropagation Through Time (BPTT) was also be used with LSTM the same way as RNN. The optimizer that we use here is Adam optimizer with a learning rate of 0.0002 and MSE as a criterion. Early stopping was also applied the same way as mentioned in the previous Recurrent Neural Network implementation subsection 4.3.4

5

Results

In this chapter, the training results of each prediction model will be shown and discussed. Several plots of the best performing model from each machine learning technique will be visualized and analyzed in detail. For the full result, please take a look at the Appendix.

5.1 Simple Linear Regression

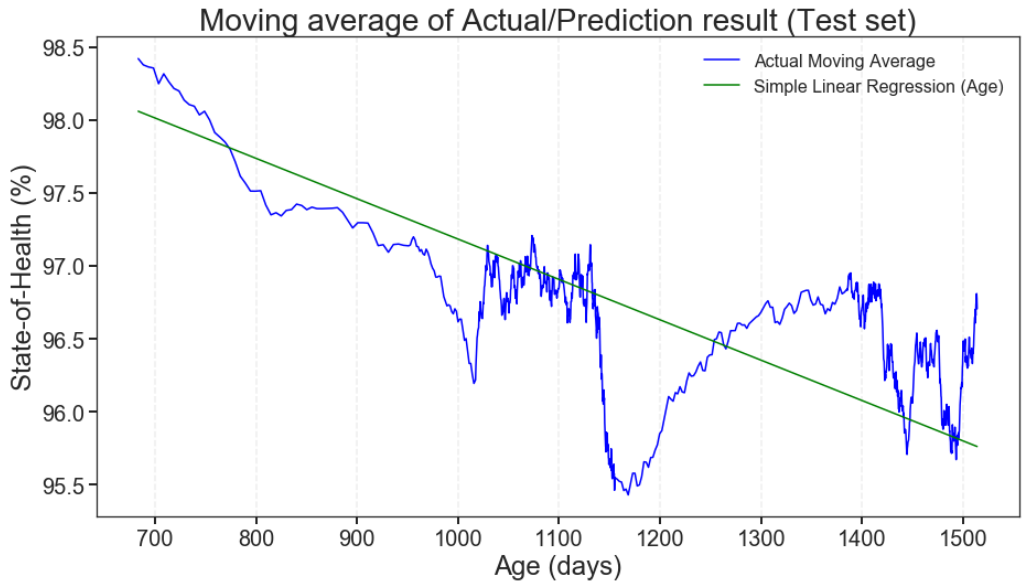


Figure 5.1: Moving average of the actual and predicted result on the test set of the Simple Linear Regression model trained with age feature

After the Simple Linear Regression model was trained with only 'age' feature as input, its MSE error (test set) is 8.4799, and the MAE error (test set) is 2.4998. From Figure 5.1, the moving average of the actual data was shown in blue color and the predicted result from the model is in green color. It can be seen that the model can represent the trend of degradation very well. However, the model cannot represent the fluctuation of the SOH due to the limitation of this simple linear regression model, which only considers the age of the vehicle as its input.

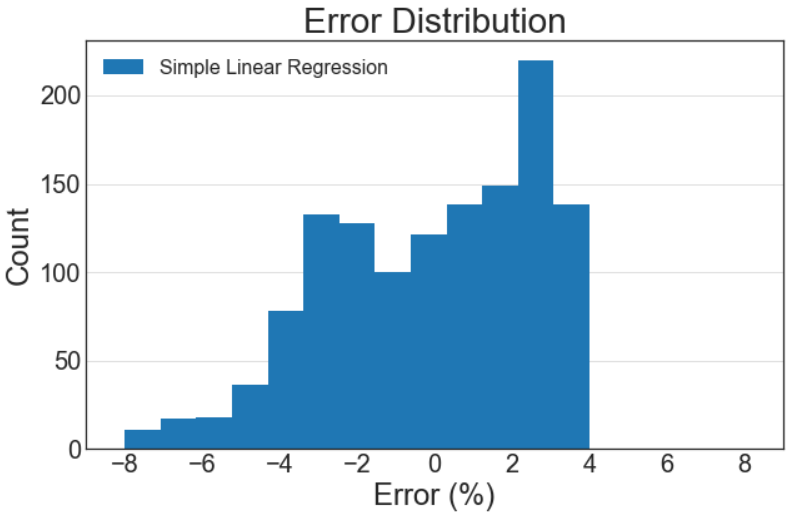


Figure 5.2: Error Distribution of the Simple Linear Regression Model

From Figure 5.2, the error distribution between the actual SOH and the predicted SOH was shown. The mean of the error is 0.0962 with a standard deviation of 2.9104. Moreover, the error can range between -8.4458 and 4.2580, which is a considerably big range of error in our case. The number of predictions with an error less than 1% is 242 predictions.

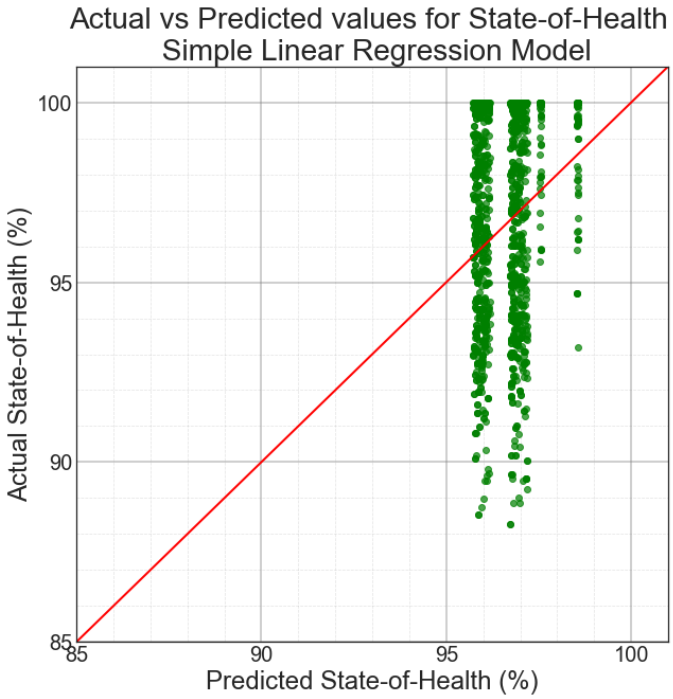


Figure 5.3: Actual vs Predicted State-of-Health values of the Simple Linear Regression Model.

Here, in Figure 5.3, the actual SOH and the predicted SOH were plotted. The perfect scenario for this plot is to have every point on the red diagonal line shown in

the figure. In this case, the simple linear regression model did not perform so well, therefore the points do not align on the red line.

5.2 Linear Regression

From table A.1, the performance of the linear regression model gives the best result when training with the basic features with lag features included. However, the performance between the model with only Basic features, Basic + 2PC and Basic + 10PC are very similar for the linear regression model with MSE loss (test set) at around 3.21-3.22 and MAE loss (test set) at 1.14-1.15.

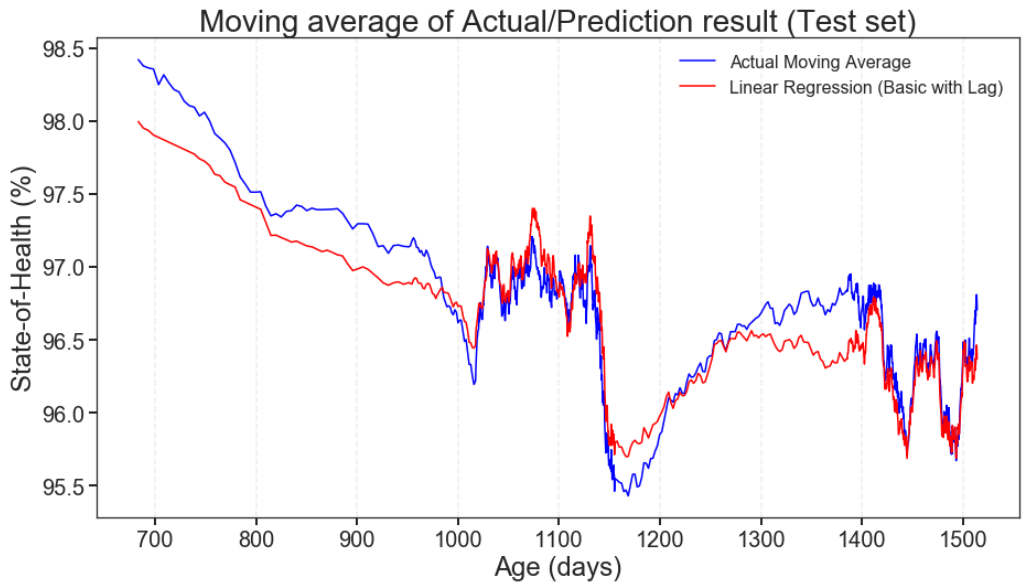


Figure 5.4: Moving average of the actual and predicted result on the test set of the Linear Regression model trained with basic + lag feature.

Here, we plot the predicted result from the linear regression model compared to the moving average of the actual SOH. From Figure 5.4, it seems this model can capture the pattern considerably well, especially between the age of 1000-1200 and 1400-1500 days that has a lot of data points.

From Figure 5.5, the error distribution plot was created from the model. When comparing this linear regression model to the previous simple linear regression model, this model has a significant improvement from the simple linear one with only the 'Age' feature considered. There are a total of 836 predictions that have an error less than 1% and the error distributes in a very balanced way between both the positive and negative side of errors. The average error of 0.0413 and a standard deviation of 1.7936. The maximum and minimum range of this error are 7.2154 and -7.6253, respectively.

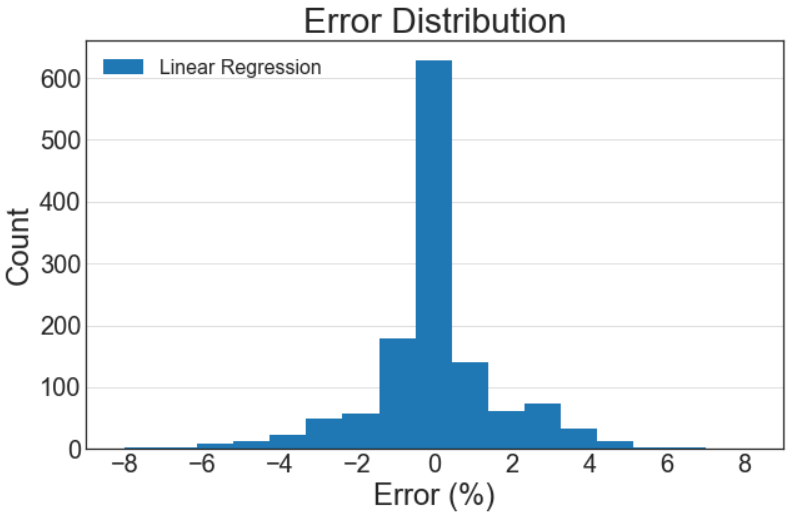


Figure 5.5: Error Distribution of the Linear Regression Model.

Figure 5.6: Actual vs Predicted State-of-Health values of the Linear Regression Model.

Here, in Figure 5.6, the actual vs predicted values were plotted to see how the error was scattered. It seems that this model can capture the pattern a lot better than the simple linear one. The scattering dots appear to be almost on the red diagonal line now.

5.3 Random Forest Regression

For the performance of the Random Forest Regression model, it performed best when training with the Basic features + 10 PC with lag features. The best model returned MSE loss at 3.1104 and MAE loss at 1.0829.

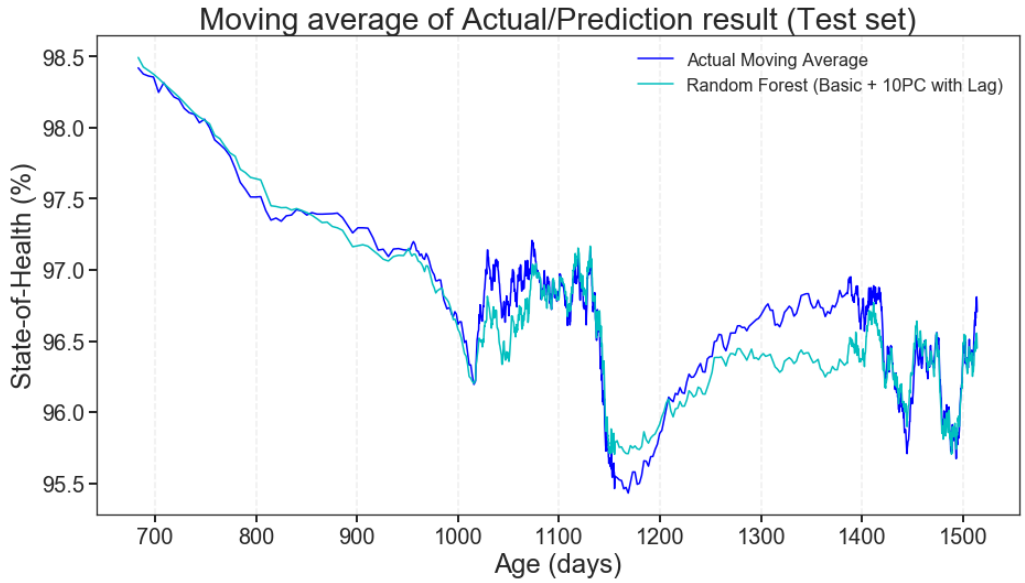


Figure 5.7: Moving average of the actual and predicted result on the test set of the Random Forest Regression model trained with basic + 10PC + lag feature.

From the moving average plot between the actual values and the predicted values in Figure 5.8, one can see that Random Forest can capture the fluctuation of the data on the beginning part a bit better than the linear regression model. The pattern is captured almost perfectly in the first 1100 days of data.

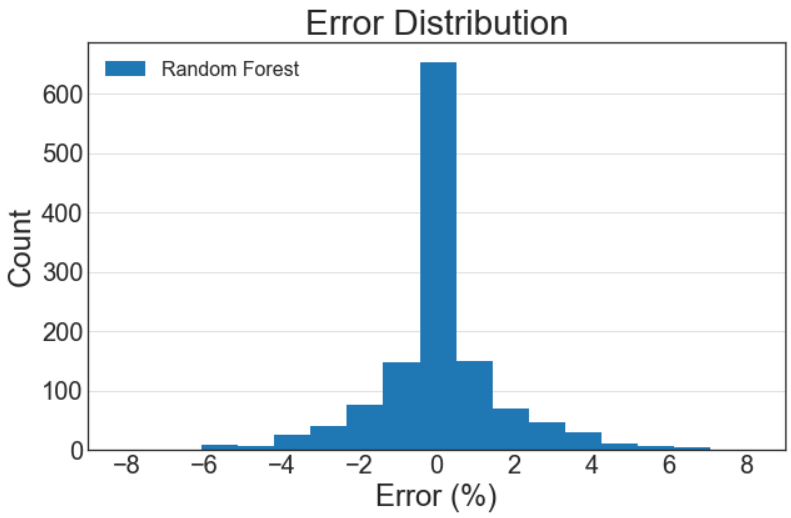


Figure 5.8: Error Distribution of the Random Forest Model.

From the error distribution chart in Figure 5.8, the error was smaller than 1 % for a total of 843 predictions. When compared to the 836 accurate predictions from the linear regression model, the random forest model seems to be a little bit better than the linear regression model. The mean error is 0.0657 with a standard deviation of 1.7612, which seems better than the best model from the linear regression models due to the smaller error variance. For this model, the maximum and minimum errors are 8.3807 and -6.9045, respectively.

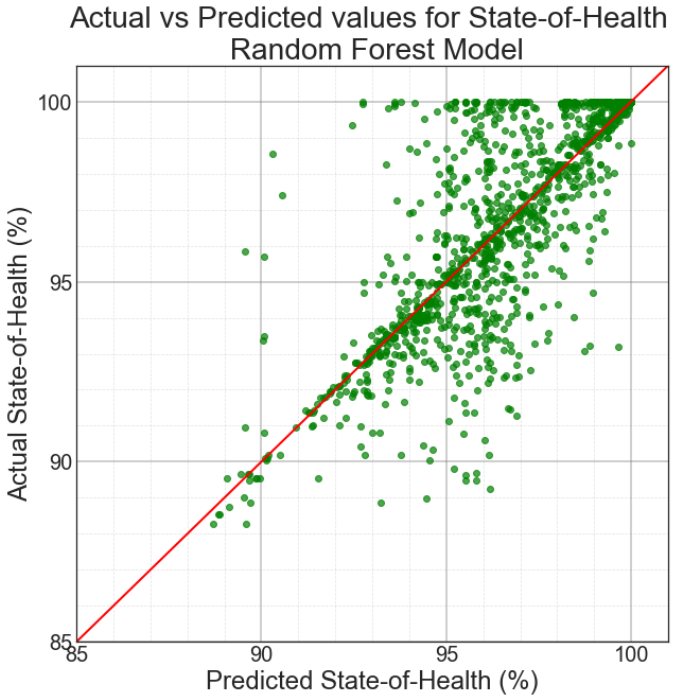


Figure 5.9: Actual vs Predicted State-of-Health values of the Random Forest Model.

Now, for the scatter plot between the actual and predicted values in Figure 5.9, the points still do not align perfectly with the red diagonal line. It seems like there is a big chunk of errors due to the unexpectedly good-SOH data points that the actual values are 100 % or almost 100 % but the model predicted them lower than it should be. However, the pattern of these scattered dots have some linearity to them, so this model did reasonably well predicting these SOH values.

5.4 Recurrent Neural Network

From Table A.1, RNN that was trained with basic features + 2PC + lag features performs the best here. It returned MSE loss (test set) of 3.0786 and MAE loss of 1.1059. However, the performance of this model is very close to the one training with basic features + lag features. So, there is not much difference here between Basic + lag and Basic + 2PC + lag. Still, RNN did perform considerably better than the Random Forest and linear regression in terms of MSE loss.

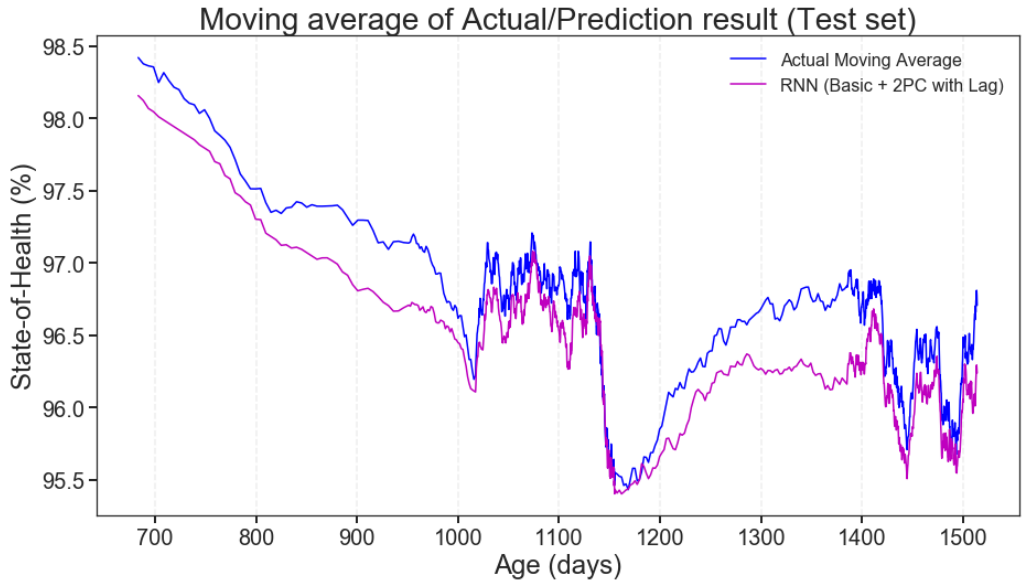


Figure 5.10: Moving average of the actual and predicted result on the test set of the Recurrent Neural Network model trained with basic + 2PC + lag feature.

For the moving average plot from Figure 5.10, RNN seems to capture excellent detail between the age of 1000-1100 days and 1400-1500 days that have a lot of data points. The overall pattern was captured very well too.

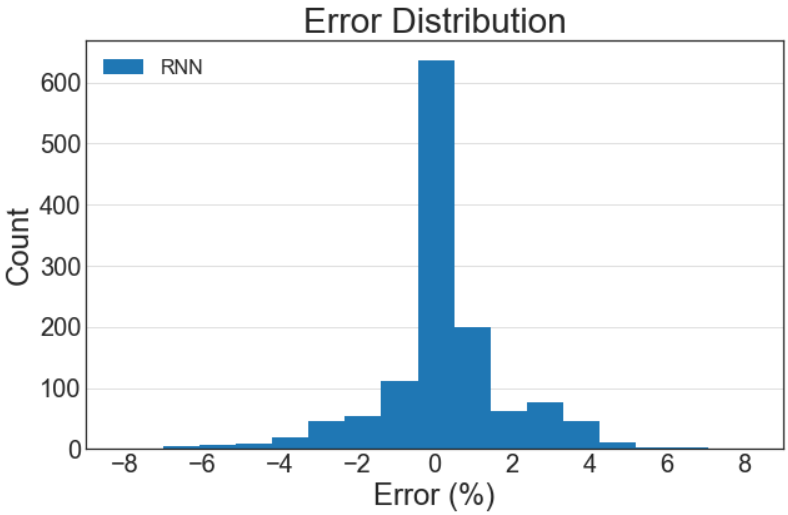


Figure 5.11: Error Distribution of the Recurrent Neural Network Model.

Here, the error distribution does not improve much from the random forest model. There are 844 predictions that have errors smaller than 1 %. The mean of the error is 0.2460 with a standard deviation of 1.7573, which means it has a smaller average error than the random forest model but with a similar standard deviation. The error can spread from -7.4973 to 7.9501, which are the maximum ranges of error we found from this model. This can be seen in figure 5.11.

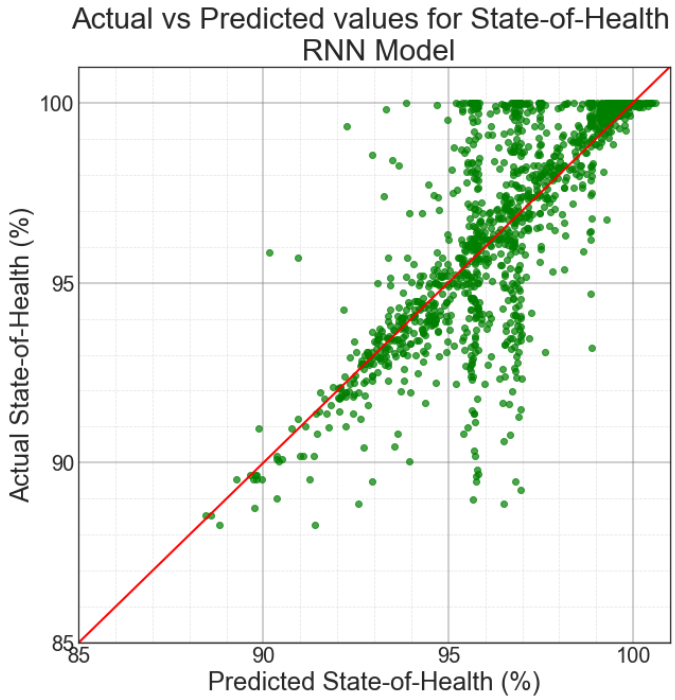


Figure 5.12: Actual vs Predicted State-of-Health values of the Recurrent Neural Network Model.

For the scatter plot of the actual vs prediction values from figure 5.12, the dots seem

to stick to the diagonal line more than the plot from the random forest model but still does not look much better in the other way.

5.5 Long Short-Term Memory

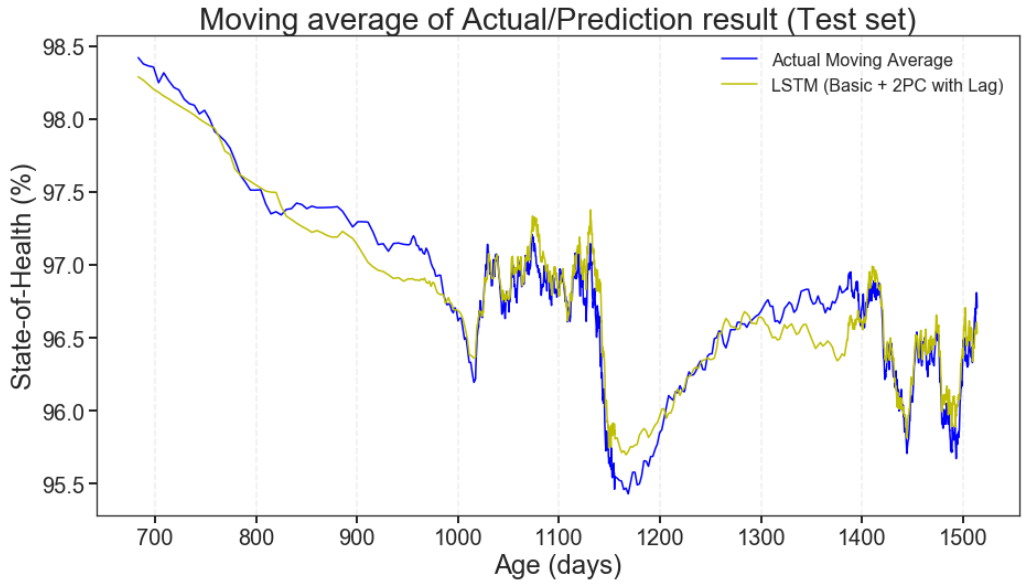


Figure 5.13: Moving average of the actual and predicted result on the test set of the Long Short-Term Memory model trained with basic + 2PC + lag feature.

The best performing LSTM model is trained with basic features + 2PC with lag features. It showed an MSE loss of 3.0200 and an MAE loss of 1.1278. Hence, this LSTM produced the best MSE and MAE losses from every model we have tried so far.

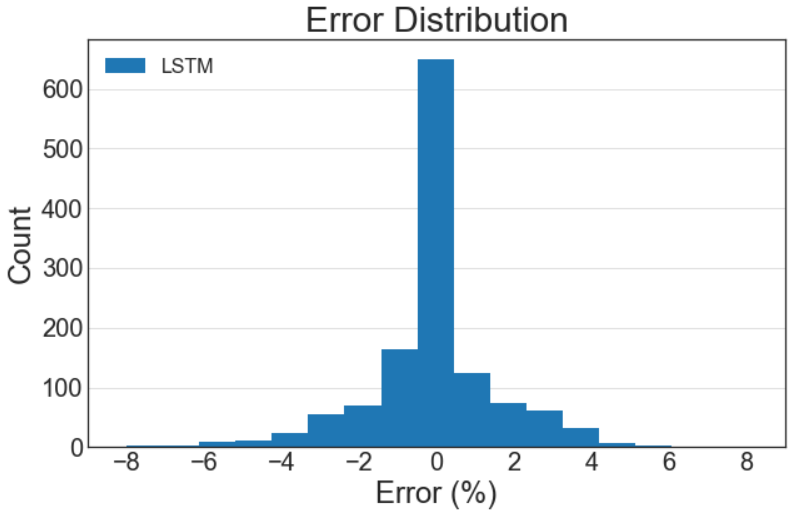


Figure 5.14: Error Distribution of the Long Short-Term Memory model.

Now, for the error distribution, LSTM got a total of 834 predictions that had errors smaller than 1%, which is a bit worse than RNN that got 844 predictions from the same criteria (<1% error). However, LSTM got the error mean of 0.0367 and standard deviation of 1.7383, which is better than the RNN that got a bigger standard deviation of the errors at 1.7573. This means the error of LSTM distributes in a smaller range than the error of RNN. Therefore, this results in a better MSE for the LSTM.

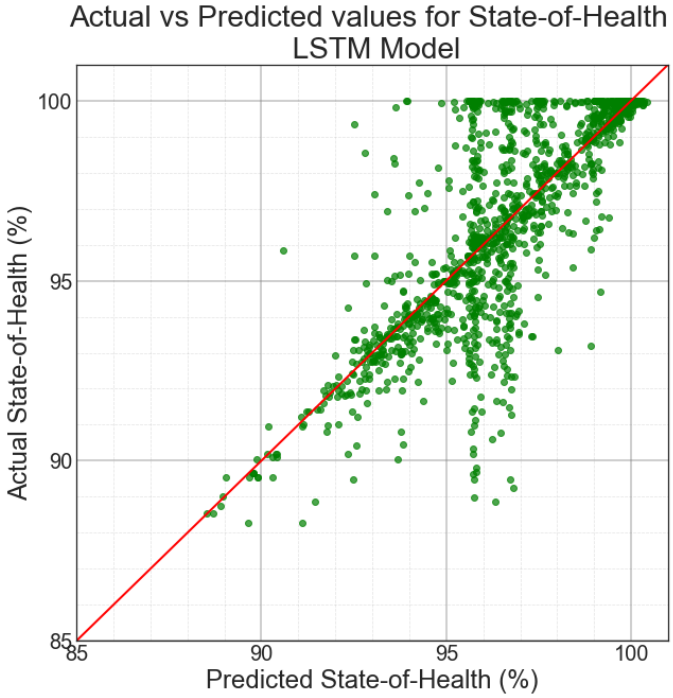


Figure 5.15: Actual vs Predicted State-of-Health values of the Long Short-Term Memory model.

For the scatter plot from figure 5.15, the pattern of these dots are very similar to the results from the previous models, that is, the pattern has some linearity to it but still not accurate enough to form a linear line. Please check table 5.1 for the numerical result of the best-performing models for each model types.

Table 5.1: MSE and MAE results on the test set of the best-performing model for each model type. See full result in the appendix chapter.

Model	**Input features**	**Lag features**	**MAE (%)**	**MSE (%)**
Simple Linear Reg.	Age	No	2.4998	8.4799
Linear Regression	Basic	Yes	1.1456	3.2189
Random Forest	Basic + 10PC	Yes	1.0806	3.0986
RNN	Basic + 2PC	Yes	1.1059	3.0786
LSTM	Basic + 2PC	Yes	1.1278	3.0200

6
Discussion

In this chapter, we will discuss and compare the performance of each prediction model. The analysis of the feature importance based on the Linear Regression model and Random Forest model will also be discussed.

6.1 Model performance comparison

This section will discuss mainly the models' performances and analyze their error distributions and their prediction patterns. Finally, the MSE of each model will be considered to conclude which prediction model is the most suitable for SOH prediction based on the DRO dataset.

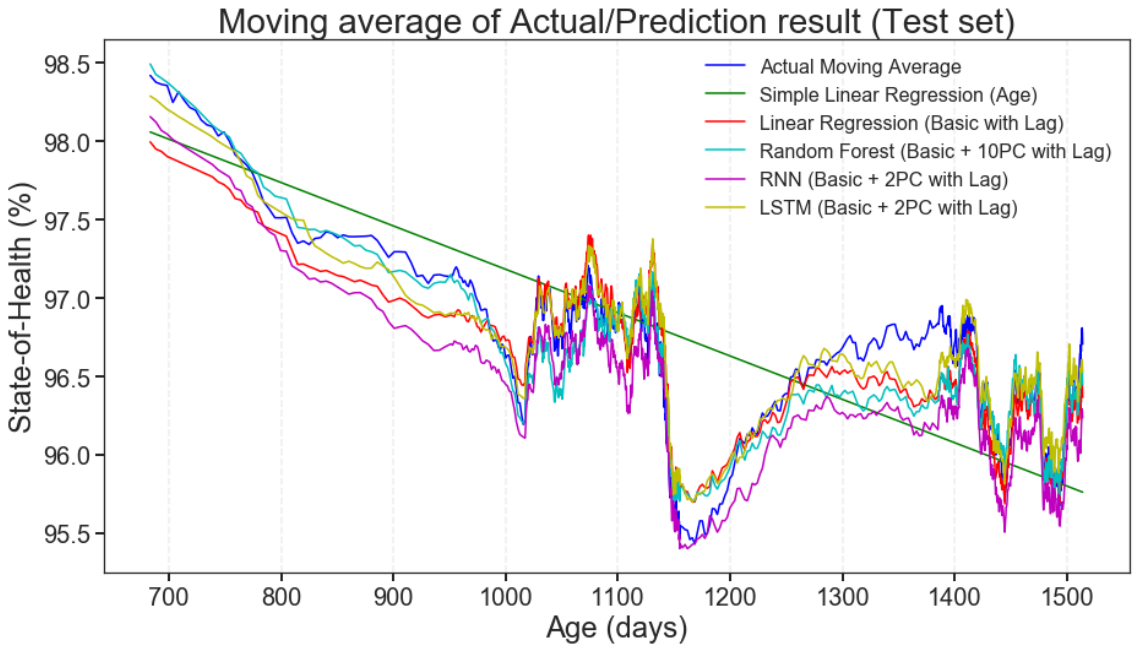


Figure 6.1: Moving average of the actual and predicted result on the test set of the machine learning models trained with different features.

From figure 6.1, the moving average of the actual data and the models' prediction results were plotted. The moving average of the actual SOH data is presented by the blue line. Apart from the green straight line from the simple linear regression model, the other models can capture the overall pattern of the actual plot very well. The hardest section that every model has the biggest errors are in between

the age range of 1300-1400 days, which is the part that the DRO data has the data gap. Therefore, the models' performances would be better if the data were more complete.

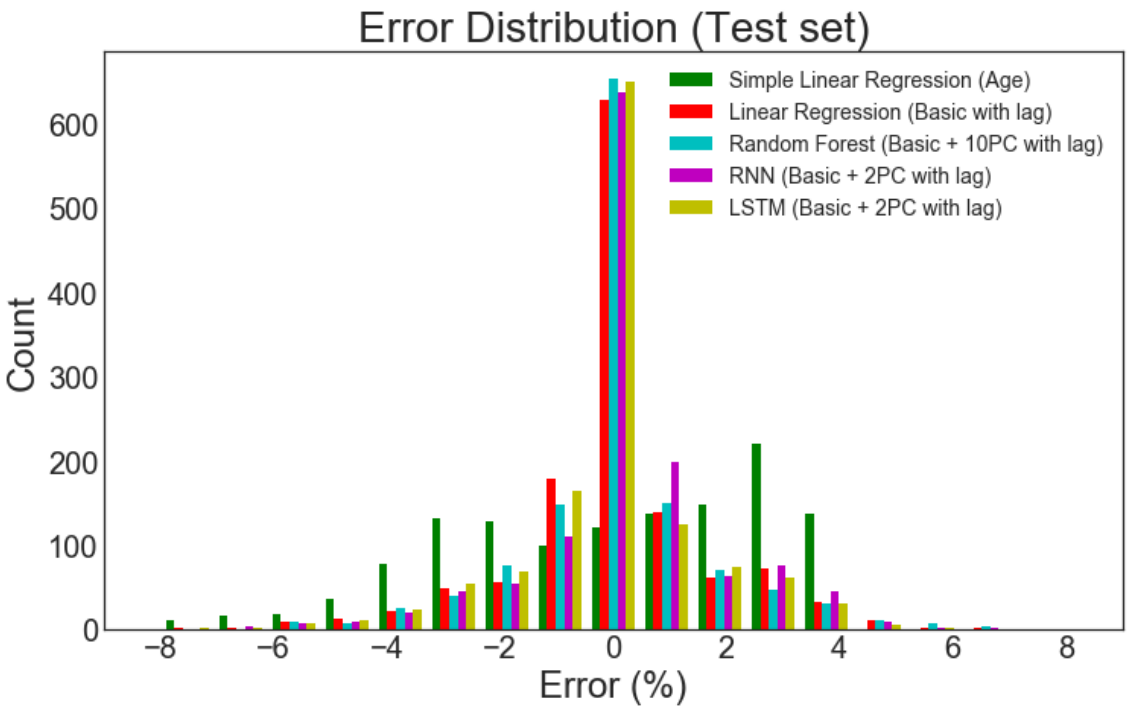


Figure 6.2: Error Distribution on the test set of the machine learning models

The Figure 6.2 above shows the error distribution. One can see that the simple linear regression model has the worst error distribution. Its error spreads widest and unbalanced. On the other hand, other models performed well. Their error distributions are very similar that they are mostly accurate and balanced. However, one cannot clearly see the differences in their performance since the patterns are very similar and might vary depending on the randomness of the training process.

Last but not least, we compare the performance of these best models numerically by their MSE. From Figure 6.3, one can clearly see that the MSE error of the Simple Linear model is significantly worse than the other models. The Simple Linear model only got an MSE of 8.4799, which is really high. Next, the Linear Regression (Basic with Lag) got an MSE of 3.2189, which improves a lot from the Simple Linear model. This shows that using only age features to predict SOH is clearly not enough to get a good MSE. Random Forest is the third best performing model with an MSE of 3.0986. Then, RNN is the second best model with an MSE of 3.0786. Finally, the most accurate model is the LSTM model with MSE of 3.0200. From our perspective, LSTM performs better than RNN due to its ability to remember longer past sequence. However, it also comes with more complexity and needs longer time to train than the RNN.

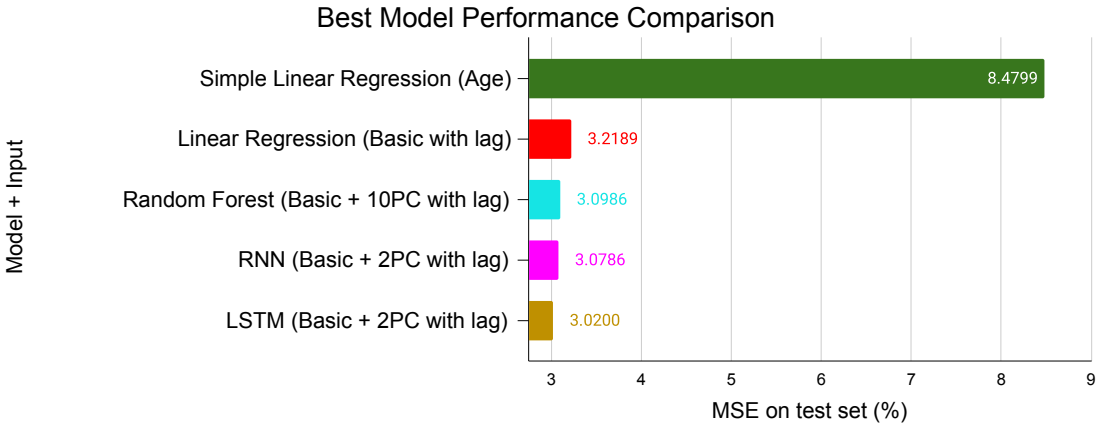


Figure 6.3: Best machine learning models performance comparison

6.2 Feature importance analysis

In this section, the feature importance analysis of Linear Regression, Random Forest, and LSTM will be discussed. The purpose of this section is to analyze the important features or sensor signals based on the characteristics of these three models.

6.2.1 Linear Regression coefficients analysis

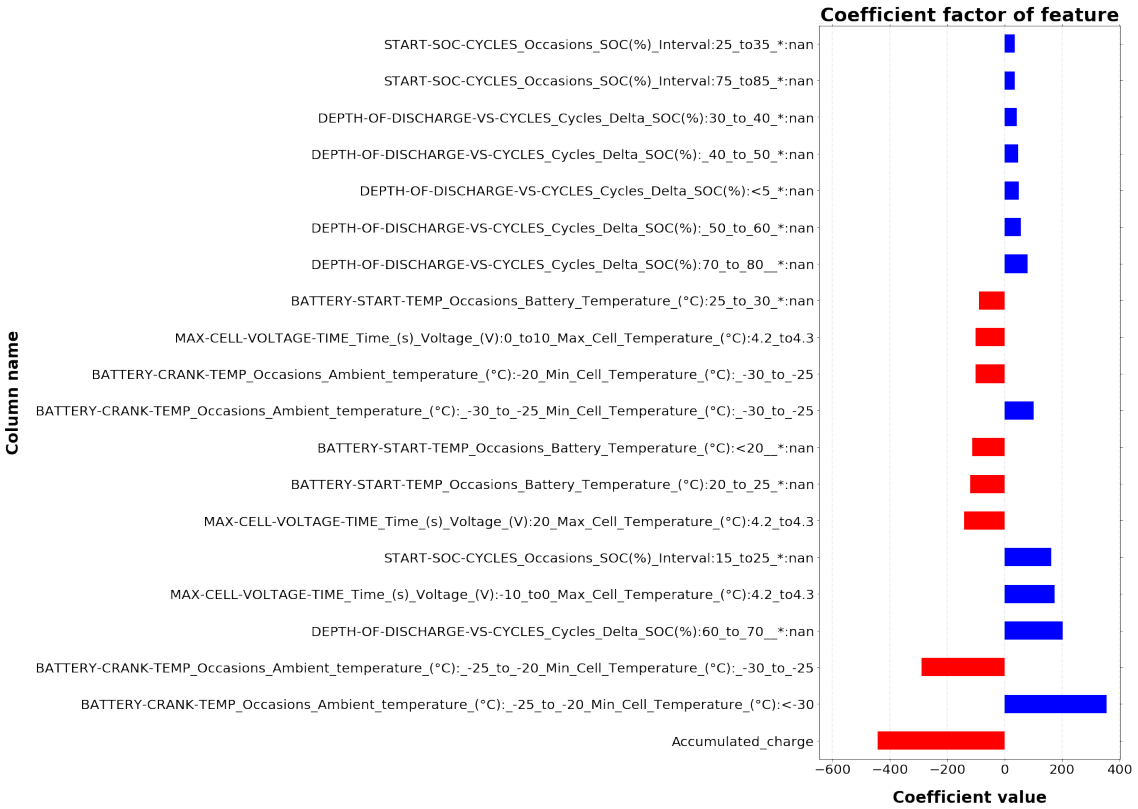


Figure 6.4: Features' coefficient values of the Linear Regression model trained with Basic and Histogram feature.

In the Linear Regression model, the input features can be ranked based on the magnitudes of its coefficients. From these magnitudes, we can see how much weight the model gave to each feature. In other words, the higher the magnitude of the coefficient, the more important the input feature was since they were standardized to have zero-mean and unit-variance. From the 6.4, it can be seen that *Accumulated_charge* has the largest negative coefficient of all input features. To be exact, the negative coefficient of it means that *Accumulated_charge* has a negative relation to the SOH. So, the higher the *Accumulated_charge* a vehicle has, the more likely that its SOH will drop lower than the other vehicles with lower *Accumulated_charge*. This makes a lot of sense since the *Accumulated_charge* refers to the number of percentages that the vehicle's battery has been charged, so it directly tells how heavy the battery usage is for each vehicle.

For the other features, it is hard to conclude whether we should follow all the coefficients or not since these coefficients might be updated in consideration of the appearances of the other features. In other words, these coefficients are all updated as a whole set of features during the training process of the model. So, it might be impossible to tell whether they are really important on their own or not. However, we can look at the whole set and see which sensor signals are frequently found in these top 20 features.

In Figure 6.4 above, one can see that the features from *'BATTERY-CRANK-TEMP'*, *'DEPTH-OF-DISCHARGE'*, *'BATTERY-START-TEMP'*, and *'START-SOC-CYCLES'* signals are found a lot in this top 20 features. So, this could mean that these signals have more effects on the SOH degradation than the other. However, it is still hard to conclude whether these signals are really important or not. Therefore, the result from the feature importance analysis of the Random Forest Regression model is needed before we can conclude which sensor signals are really important to the battery degradation problem.

6.2.2 Random Forest analysis

For the random forest model, we can analyze its feature importance property made by the Scikit-learn library based on the calculation of Gini importance [35, 36]. The random forest model that we will analyze in this section will be trained with the basic features + histogram features without lag feature included so that one can see which features are used frequently by the Random Forest model.

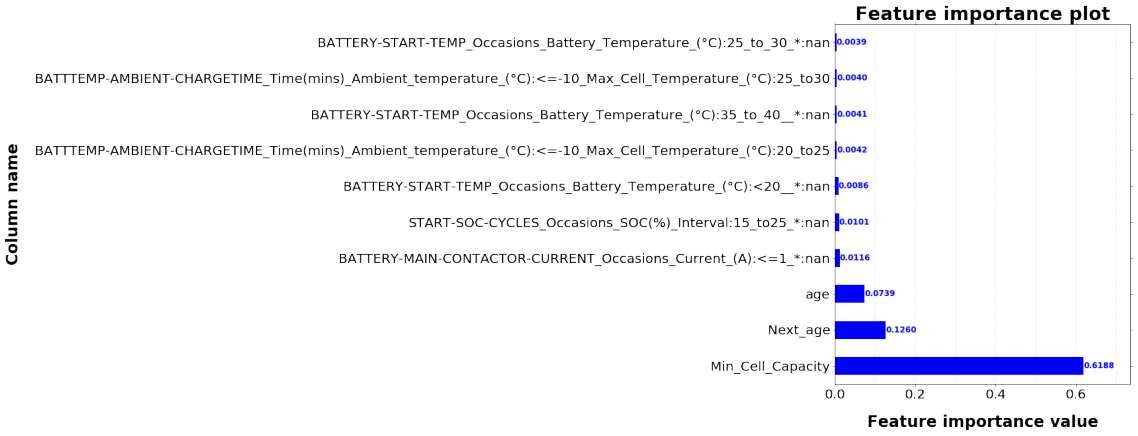


Figure 6.5: Top 10 List of the Features importance of the Random Forest Regression model trained with Basic and Histogram feature.

From Figure 6.5, it is not surprising to see that the Random Forest model used *Min_Cell_Capacity* (current SOH), *Next_age* (target age), and *age* (current age) most frequently. These parameters are surely the most important parameter to forecast SOH in the future.

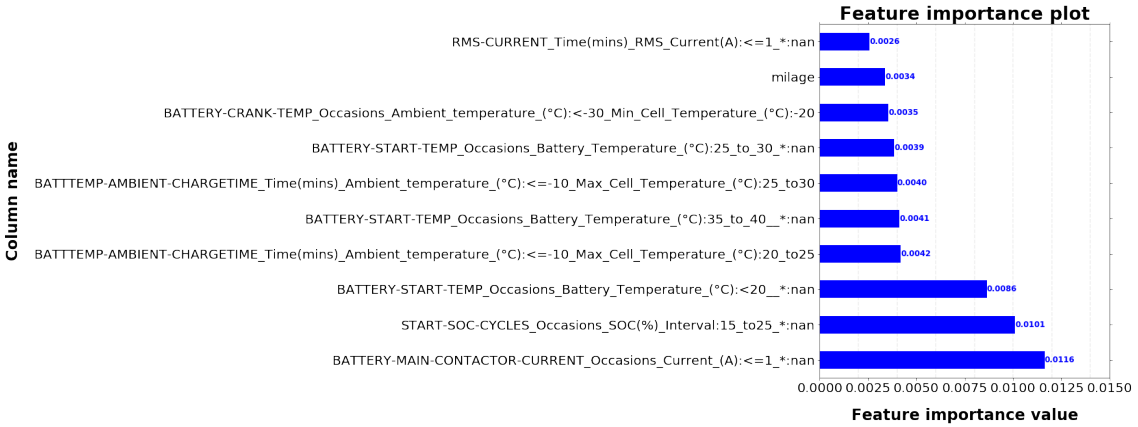


Figure 6.6: Features importance of the Random Forest Regression model trained with Basic and Histogram features that are not include current SOH (*Min_Cell_Capacity*), current age (*age*), and next age (*Next_age*) feature.

From Figure 6.6, we removed the current SOH, current age, and the target age from the chart so that we can analyze the other features more clearly. Now, one can see that *'BATTERY-MAIN-CONTACTOR-CURRENT_Occasions_Current_(A):<=1_*:nan'* (i.e. the number of times that the current of the main contactor went lower or equal to 1 ampere), *'START-SOC-CYCLES_Occasions_SOC(%)_Interval:15_to25_*:nan'* (i.e. the number of times that the car was started while its SOC was between 15 and 25%), and *'BATTERY-START-TEMP_Occasions_Battery_Temperature_(°C):<20___*:nan'* (i.e. the number of times that the car was started while the temperature was below 20°C) are the top 3 most important features from the set. Apart from these 3 features, the other features are considerably less used by the Random Forest model.

6.2.3 LSTM analysis

Feature importance analysis was done on the LSTM by measuring the improvement of the LSTM model based on the addition of specific sensor features to the training data.

Multiple LSTM models were trained on multiple set of input features constructed from the combination of Basic features and several subsets of the Histogram features. After training the models, each model was evaluated on a resampled test set to see how these sensor features improved the performance of the LSTM. The negative delta MSE means improvement, while positive delta MSE means that the performance was worsened by the added features. The result from the experiment is shown in the Table 6.1 below:

Table 6.1: The feature importance result of the LSTM model. Negative delta_MSE (%) means improvement, otherwise the performance of the model is worse than the LSTM trained with only Basic features.

Sensor features	**Delta_MSE (%)**
START-SOC-CYCLES	-7.8524
RMS-CURRENT	-7.5992
BATTERY-START-TEMP	-7.3774
DEPTH-OF-DISCHARGE-VS-CYCLES	-7.0413
BATTERY-CRANK-TEMP	-7.0321
BATTERY-MAIN-CONTACTOR-CURRENT	-6.3593
PLUG-CHARGE-CURRENT	-5.9710
MAX-CELL-VOLTAGE-TIME	-5.8650
MIN-CELL-VOLTAGE	-5.2661
MIN-SYSTEM-VOLTAGE	-4.0365
Basic features	0.0000
BATTTEMP-AMBIENT-CHARGETIME	5.1271

From the result, one can see that 'START-SOC-CYCLES' improves the performance of the model by 7.85% from using only Basic features. So, *'START-SOC-CYCLES'* is considered as the most important sensor features based on LSTM. Then, it was followed by *'RMS-CURRENT'*, *'BATTERY-START-TEMP'*, *'DEPTH-OF-DISCHARGE-VS-CYCLES'* and *'BATTERY-CRANK-TEMP'*, respectively. It is good to mention that these performance improvement are only based on a resampled test set, so the improvement might be different depending on the random distribution of the data. Therefore, these results might not be application on the other datasets.

To sum up the feature importance analysis from the Linear Regression, Random Forest, and the LSTM model, the *'START-SOC-CYCLES'*, *'DEPTH-OF-DISCHARGE-VS-CYCLES'*, *'BATTERY-START-TEMP'*, and *'BATTERY-CRANK-TEMP'* sensor features are the most frequently found signal in the top features from these models. This could mean that starting and cranking the vehicle during certain temperatures could affect the SOH degradation to some extent based on the data sample from the DRO dataset. Moreover, starting the car in some range of SOC, especially below 25%, could degrade the battery by introducing mechanical stress to the electrodes and lithium plating inside the battery (including while charging). Furthermore, the delta SOC between each usage cycle seems to have important role to the SOH degradation too. It is good to mention that these discoveries are based only on the DRO dataset collected from Volvo XC90 models, which are hybrid EVs. Therefore, more research is needed to confirm this hypothesis on the other EV models.

7

Conclusion

This chapter will conclude all the things we have done so far in this thesis. The first section will summarize all of the results and discoveries to answer all of the research questions of this thesis. After that, we will talk about future work and other possibilities for this project.

7.1 Summary

The goal of this thesis is to analyze the possibility to use the high-dimensional data from various sensors in electric vehicles to improve the State-of-Health forecasting capability and also to analyze any interesting discovery from the data and the machine learning models. Below are the main research questions that this thesis is aiming for:

- Does the additional high-dimensional data from EV sensors improve battery degradation forecasting performance of the prediction models?
- Does the usage of the Principal Component Analysis (PCA) improve the performances of the prediction models compared to using the whole dataset features?
- Does the additional lag features improve State-of-Health forecasting performance of the machine learning models?
- What is the most suitable machine learning model for SOH forecasting based on DRO data?
- What are the possible causes of the battery degradation in the EVs?

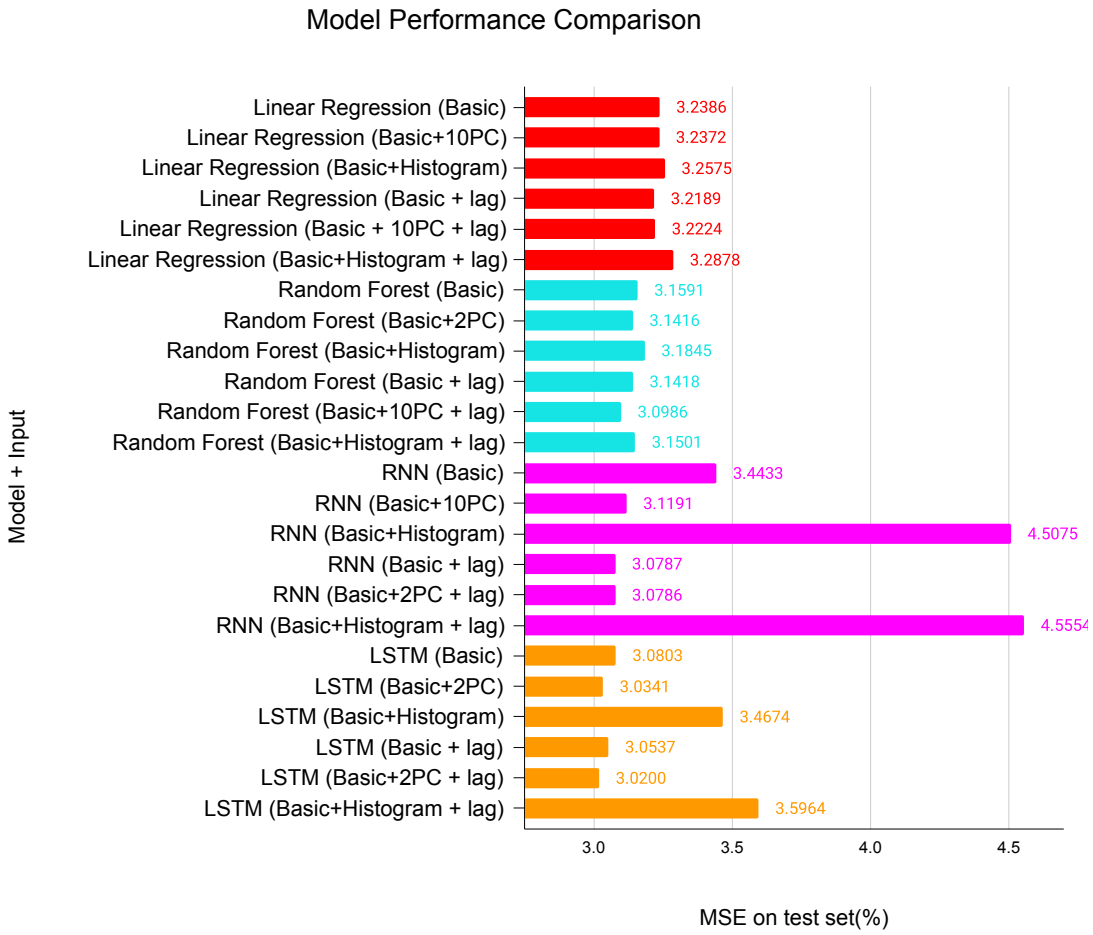


Figure 7.1: Performance comparison from different machine learning models and different inputs.

For the first research question: *"Does the additional high-dimensional data from EV sensors improve battery degradation forecasting performance of the prediction models?"*, the answer is "yes" but not very significantly on this specific DRO dataset.

In Figure 7.1 above, one can see that using basic features with additional principal components are a little bit better than using basic features to train the model. In some models, the performance difference in terms of MSE might be as low as 0.001, but in some other models, the improvement MSE might be as high as 0.05. The only exception in our case is the linear regression model that the basic feature fitted model has better MSE than the model fitted on the basic features with 10PC. Thus, the improvement will also depend on the type of machine learning models and their complexity. In conclusion, having additional features in terms of principal components will mostly improve the performance of the models.

It is worth noting that the improvement of this specific method can only be concluded on the DRO dataset since the dataset is unique due to its age gaps, in-house sensor features, and the real customer's data collection. These reasons make it impossible to guarantee that the improvement will be the same on the other datasets. Therefore, this could be another research area for future work to see whether this method also works on the other datasets or not.

For the second question: *"Does the usage of the Principal Component Analysis (PCA) improve the performances of the prediction models compared to using the whole dataset features?"*, answer is also yes and it is very significant.

From Figure 7.1, one can see that adding the whole histogram features to the training data without applying PCA results in significantly worse performance in every model. Especially in RNN and LSTM, the performance of these models is significantly worse when training the data with basic + histogram than the basic + 2PC version. Summing up, applying PCA to reduce the dimensionality of the data before training the model improves the model performance significantly.

For the third question: *"Does adding lag features help improve State-of-Health forecasting performance of the machine learning models?"*, answer is yes, but it should not increase the dimensionality of the input data too much.

From figure 7.1, the comparison between the model training with and without lag features was shown and one can see that the models trained with lag features added perform significantly better unless the input feature has Histogram features in it. Most of the models produce lower MSE when lag features were added to the input features. The improvement can range from 0.015 to 0.05. However, the models that are trained with Basic+Histogram features not only did not improve but also reduced their performance. This happened because its already high-dimensional input features were added with another set of lag features, so the dimensionality for this model is doubled. To summarize the above, lag features do help improve the model performance but it should not increase the input features' dimension too much, otherwise, it will suffer from the increased dimensionality instead.

For the question: *"What is the most suitable machine learning model for SOH forecasting based on DRO data?"*

From Figure 7.1, the best performing model for this DRO data is the LSTM trained with basic+2PC with lag features added. The model got its MSE of 3.0200, which is significantly better than the other models. Compared to the best RNN that got an MSE of 3.0786, the LSTM is still considerably better. Regarding the DRO data that is considered to be a short sequence with only around 2-6 data points in most cases, LSTM still performs better than RNN. In conclusion, LSTM is the most suitable model for SOH forecasting on the DRO data.

Last but not least, for the question: *"What are the possible causes of the battery degradation in the EVs?"*.

In the previous chapter, we have discussed on the feature importances based on the Linear Regression and Random Forest Regression models and found that the features constructed from *'START-SOC-CYCLES'*, *'DEPTH-OF-DISCHARGE-VS-CYCLES'*, *'BATTERY-START-TEMP'*, and *'BATTERY-CRANK-TEMP'* sensor signals were considered important to the models. This could mean that starting and cranking the hybrid vehicles during various ranges of temperatures could affect the battery considerably. Furthermore, starting the car below 25% SOC and delta SOC between each usage cycle could also degrade the battery to some extent too. However, these findings might not be applicable to all EV models since the study only focused on the DRO dataset. This is only some part of the possible causes of the battery degradation problem, so more research is needed to conclude on the actual causes of it.

7.2 Future work

This thesis has a lot of room for further improvement in the future. First of all, other types of dimensionality reduction techniques apart from PCA are a nice area to experiment on such as Kernel PCA [37], Linear discriminant analysis (LDA) [38], Canonical Correlation Analysis (CCA) [39], and etc. There might be a better dimensionality reduction technique than PCA that can create better new features without the limitation of linear combination like PCA. Applying soft feature selection before applying PCA to the data might be another possibility worth trying since it might get rid of some irrelevant or low-quality features before applying PCA. Another possible improvement is to experiment on other types of prediction models such as Gradient Boosting tree [40, 41], Attention-based models [42, 43], Hidden Markov Models [44], and etc. These models might yield better performances than the models that were used in this research. Moreover, in the future, it is possible to have better quality data with a longer time span, higher-sampling rate, more features and has more number of samples than the current one. The better data might help us draw a better conclusion on how the battery degrades and what leads to battery problems.

Bibliography

[1] H. J. Chae, W. Y. Kim, S. Y. Yun, Y. S. Jeong, J. Y. Lee, and H. T. Moon. 3.3kw on board charger for electric vehicle. In *8th International Conference on Power Electronics - ECCE Asia*, pages 2717–2719, 2011.

[2] Md Rafiul Hassan and Baikunth Nath. Stock market forecasting using hidden markov model: a new approach. In *5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*, pages 192–196. IEEE, 2005.

[3] Erkam Guresen, Gulgun Kayakutlu, and Tugrul U Daim. Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, 38(8):10389–10397, 2011.

[4] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[5] Vincenzo Marano, Simona Onori, Yann Guezennec, Giorgio Rizzoni, and Nullo Madella. Lithium-ion batteries life estimation for plug-in hybrid electric vehicles. In *2009 IEEE vehicle power and propulsion conference*, pages 536–543. IEEE, 2009.

[6] Jie Liu, Abhinav Saxena, Kai Goebel, Bhaskar Saha, and Wilson Wang. An adaptive recurrent neural network for remaining useful life prediction of lithium-ion batteries. Technical report, NATIONAL AERONAUTICS AND SPACE ADMINISTRATION MOFFETT FIELD CA AMES RESEARCH ..., 2010.

[7] Martin Murnane and Adel Ghazel. A closer look at state of charge (SOC) and state of health (SOH) estimation techniques for batteries. page 2, 2017.

[8] Bruno Scrosati and Jürgen Garche. Lithium batteries: Status, prospects and future. *Journal of power sources*, 195(9):2419–2430, 2010.

[9] Steven Chu and Arun Majumdar. Opportunities and challenges for a sustainable energy future. *nature*, 488(7411):294–303, 2012.

[10] Tiansi Wang, Lei Pei, Tingting Wang, Rengui Lu, and Chunbo Zhu. On-board state-of-health estimation at a wide ambient temperature range in lithium-ion batteries. *Energies*, 8(8):8467–8481, 2015.

[11] Christoph R Birkl, Matthew R Roberts, Euan McTurk, Peter G Bruce, and David A Howey. Degradation diagnostics for lithium ion cells. *Journal of Power Sources*, 341:373–386, 2017.

[12] Matthieu Dubarry, Cyril Truchot, Mikaël Cugnet, Bor Yann Liaw, Kevin Gering, Sergiy Sazhin, David Jamison, and Christopher Michelbacher. Evaluation of commercial lithium-ion cells based on composite positive electrode for plug-

in hybrid electric vehicle applications. part i: Initial characterizations. *Journal of power sources*, 196(23):10328–10335, 2011.

[13] Gang Ning and Branko N Popov. Cycle life modeling of lithium-ion batteries. *Journal of The Electrochemical Society*, 151(10):A1584–A1591, 2004.

[14] Ivana Semanjski and Sidharta Gautama. Forecasting the state of health of electric vehicle batteries to evaluate the viability of car sharing practices. *Energies*, 9(12):1025, 2016.

[15] Seungchul Lee, Harry Cui, Mohammad Rezvanizaniani, and Jun Ni. Battery prognostics: Soc and soh prediction. In *ASME 2012 International Manufacturing Science and Engineering Conference collocated with the 40th North American Manufacturing Research Conference and in participation with the International Conference on Tribology Materials and Processing*, pages 689–695. American Society of Mechanical Engineers Digital Collection, 2012.

[16] Q Badey, G Cherouvrier, Y Reynier, JM Duffault, and S Franger. Ageing forecast of lithium-ion batteries for electric and hybrid vehicles. *Curr. Top. Electrochem*, 16:65–79, 2011.

[17] Adel Mellit, M Benghanem, and Soteris A Kalogirou. Modeling and simulation of a stand-alone photovoltaic system using an adaptive artificial neural network: Proposition for a new sizing procedure. *Renewable energy*, 32(2):285–313, 2007.

[18] Fangfang Yang, Xiangbao Song, Fan Xu, and Kwok-Leung Tsui. State-of-charge estimation of lithium-ion batteries via long short-term memory network. *IEEE Access*, 7:53792–53799, 2019.

[19] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[20] Sima Siami-Namini and Akbar Siami Namin. Forecasting economics and financial time series: Arima vs. lstm. *arXiv preprint arXiv:1803.06386*, 2018.

[21] Javier Contreras, Rosario Espinola, Francisco J Nogales, and Antonio J Conejo. Arima models to predict next-day electricity prices. *IEEE transactions on power systems*, 18(3):1014–1020, 2003.

[22] Sebastian Raschka. About Feature Scaling and Normalization (and the effect of standardization for Machine Learning algorithms). *Polar Political Legal Anthropology Rev*, 30(1):67–89, 2014.

[23] Svante Wold, Kim H. Esbensen, and Paul Geladi. Principal Component Analysis. *Chemometrics and Intelligent Laboratory Systems*, 2:37–52, 1987.

[24] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

[25] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[26] Manish Kumar and M Thenmozhi. Forecasting stock index movement: A comparison of support vector machines and random forest. In *Indian institute of capital markets 9th capital markets conference paper*, 2006.

[27] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[28] A Amidi and S Amidi. Recurrent neural networks cheatsheet. Technical report, 2019. URL: https://stanford.edu/∼ shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks, Accessed: 28.05.2020.

[29] Mikael Boden. A guide to recurrent neural networks and backpropagation. *the Dallas project*, 2002.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[31] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[32] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. Accessed: 16.07.2020.

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[35] Stefano Nembrini, Inke R König, and Marvin N Wright. The revival of the Gini importance? *Bioinformatics*, 34(21):3711–3718, 2018.

[36] Bjoern H Menze, B Michael Kelm, Ralf Masuch, Uwe Himmelreich, Peter Bachert, Wolfgang Petrich, and Fred A Hamprecht. A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC bioinformatics*, 10(1):213, 2009.

[37] Sebastian Mika, Bernhard Schölkopf, Alex J Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel pca and de-noising in feature spaces. In *Advances in neural information processing systems*, pages 536–542, 1999.

[38] Suresh Balakrishnama and Aravind Ganapathiraju. Linear discriminant analysis-a brief tutorial. In *Institute for Signal and information Processing*, volume 18, pages 1–8, 1998.

[39] Bruce Thompson. Canonical correlation analysis. *Encyclopedia of statistics in behavioral science*, 2005.

[40] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pages 3146–3154, 2017.

[41] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[42] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[43] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

[44] A Poritz. Linear predictive hidden markov models and the speech signal. In *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 7, pages 1291–1294. IEEE, 1982.

A

Appendix 1

Table A.1: MSE and MAE results on the test set of the machine learning models trained with different sets of input features.

Model	**Input features**	**Lag features**	**MAE(%)**	**MSE(%)**
Simple Linear Reg.	Age	No	2.4998	8.4799
Linear Regression	SOH + Age	No	1.1710	3.4136
Linear Regression	Basic	No	1.1504	3.2386
Linear Regression	Basic + 2PC	No	1.1527	3.2418
Linear Regression	**Basic + 10PC**	**No**	**1.1524**	**3.2372**
Linear Regression	Basic + 50PC	No	1.1712	3.2558
Linear Regression	Basic + Histogram	No	1.1847	3.2575
Linear Regression	SOH + Age	Yes	1.1472	3.3429
Linear Regression	**Basic**	**Yes**	**1.1456**	**3.2189**
Linear Regression	Basic + 2PC	Yes	1.1496	3.2249
Linear Regression	Basic + 10PC	Yes	1.1503	3.2224
Linear Regression	Basic + 50PC	Yes	1.1783	3.2664
Linear Regression	Basic + Histogram	Yes	1.2060	3.2878
Random Forest	SOH + Age	No	1.1848	3.5860
Random Forest	Basic	No	1.0921	3.1591
Random Forest	**Basic + 2PC**	**No**	**1.0978**	**3.1416**
Random Forest	Basic + 10PC	No	1.1103	3.1545
Random Forest	Basic + 50PC	No	1.1333	3.1664
Random Forest	Basic + Histogram	No	1.1115	3.1845
Random Forest	SOH + Age	Yes	1.1287	3.4295
Random Forest	Basic	Yes	1.0690	3.1418
Random Forest	Basic + 2PC	Yes	1.0655	3.1077
Random Forest	**Basic + 10PC**	**Yes**	**1.0806**	**3.0986**
Random Forest	Basic + 50PC	Yes	1.1158	3.1491
Random Forest	Basic + Histogram	Yes	1.0885	3.1501

Continued on next page

Table A.1 – *Continued from previous page*

Model	**Input features**	**Lag features**	**MAE(%)**	**MSE(%)**
RNN	SOH + Age	No	1.1415	3.2979
RNN	Basic	No	1.0454	3.4433
RNN	Basic + 2PC	No	1.0324	3.2641
RNN	**Basic + 10PC**	**No**	**1.1177**	**3.1191**
RNN	Basic + 50PC	No	1.1680	3.1600
RNN	Basic + Histogram	No	1.3330	4.5075
RNN	SOH + Age	Yes	1.1475	3.3059
RNN	Basic	Yes	1.0945	3.0787
RNN	**Basic + 2PC**	**Yes**	**1.1059**	**3.0786**
RNN	Basic + 10PC	Yes	1.1529	3.0978
RNN	Basic + 50PC	Yes	1.1902	3.3410
RNN	Basic + Histogram	Yes	1.3645	4.5554
LSTM	SOH + Age	No	1.1051	3.2029
LSTM	Basic	No	1.1015	3.0803
LSTM	**Basic + 2PC**	**No**	**1.1141**	**3.0341**
LSTM	Basic + 10PC	No	1.1317	3.1168
LSTM	Basic + 50PC	No	1.2034	3.3212
LSTM	Basic + Histogram	No	1.2601	3.4674
LSTM	SOH + Age	Yes	1.1246	3.2750
LSTM	Basic	Yes	1.0881	3.0537
LSTM	**Basic + 2PC**	**Yes**	**1.1278**	**3.0200**
LSTM	Basic + 10PC	Yes	1.1400	3.1201
LSTM	Basic + 50PC	Yes	1.2820	3.5305
LSTM	Basic + Histogram	Yes	1.2959	3.5964