**REPUBLIC OF TURKEY**
**YILDIZ TECHNICAL UNIVERSITY**
**GRADUATE SCHOOL OF SCIENCE AND ENGINEERING**

# BATTERY STATE OF HEALTH (SOH) PREDICTION USING MACHINE LEARNING

**Kıray GÜNER**

MASTER OF SCIENCE THESIS
Department of Mechanical Engineering
Machine Theory and Control

Supervisor
Assoc. Prof. Dr. Meral BAYRAKTAR

July, 2023

**REPUBLIC OF TURKEY**
**YILDIZ TECHNICAL UNIVERSITY**
**GRADUATE SCHOOL OF SCIENCE AND ENGINEERING**

# BATTERY STATE OF HEALTH (SOH) PREDICTION USING MACHINE LEARNING

A thesis submitted by Kıray GÜNER in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 06.07.2023 in Department of Mechanical Engineering, Machine Theory and Control.

Assoc. Prof. Dr. Meral BAYRAKTAR
Yildiz Technical University
Supervisor

**Approved By the Examining Committee**

Assoc. Prof. Dr. Meral BAYRAKTAR, Supervisor
Yildiz Technical University _____

Assoc. Prof. Dr. Kadir Erkan, Member
Yildiz Technical University _____

Assoc. Prof. Dr. Osman Taha Şen, Member
Istanbul Technical University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of Battery State of Health (SOH) Prediction Using Machine Learning supervised by my supervisor, Assoc. Prof. Dr. Meral BAYRAKTAR. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Kıray GÜNER

Signature

# ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to all those who have contributed to the successful completion of this thesis.

First and foremost, I would like to thank my supervisor, Meral Bayraktar, for her guidance, support, and invaluable insights throughout the entire research process.

I am grateful to the faculty members of Yıldız Technical University and Bilkent University and all the teachers I have had during my education life for providing me with the background that allowed me to complete this thesis.

I extend my appreciation to all the people that spent their time and effort to gather the data used for this study. Without their cooperation, this research would not have been possible. I am deeply thankful for their willingness to be part of this endeavor.

I would like to acknowledge the assistance and support received from my family and friends thoroughout my life.

Kıray GÜNER

# TABLE OF CONTENTS

**LIST OF FIGURES** vii

**LIST OF TABLES** viii

**ABSTRACT** ix

**ÖZET** xi

**1 INTRODUCTION** 1

**2 Literature Review** 2

2.1 Lithium-Ion Cell Basics . . . 2
2.2 Components Of A Lithium-Ion Battery . . . 3
2.3 Battery Degradation and End of Life . . . 4
2.4 Degradation in cell components . . . 5
2.5 Battery State of Health (SOH) . . . 8
2.6 Machine Learning . . . 9
2.7 Artificial Neural Networks . . . 10
2.8 Performance Metrics . . . 11
2.9 Related Studies in Literature . . . 12

**3 Methodology** 14

3.1 Data Collection and Preprocessing . . . 14
3.2 Feature Extraction . . . 15

**4 Results and Discussion** 18

4.1 Model Results . . . 18
4.2 Comparison Of The Models . . . 25

**5 Conclusion and Future Work** 27

5.1 Conclusion . . . 27
5.2 Future Work . . . 28

**REFERENCES** 29

**A Python Code Used with NASA Dataset** **32**

**PUBLICATIONS FROM THE THESIS** **44**

# LIST OF FIGURES

**Figure 2.1** Lithium-Ion Battery Operation [4] . . . . . 2
**Figure 2.2** Lithium Plating [14] . . . . . 6
**Figure 2.3** Categories of machine Learning [22] . . . . . 10
**Figure 3.1** Lab Data Voltage vs. Duration . . . . . 15
**Figure 3.2** NASA Data Voltage vs. Duration . . . . . 15
**Figure 3.3** Lab Data Correlation Heatmap . . . . . 16
**Figure 3.4** NASA Data Correlation Heatmap . . . . . 16
**Figure 4.1** Linear Regression Model on Lab Data . . . . . 19
**Figure 4.2** Linear Regression Model on NASA Data . . . . . 19
**Figure 4.3** Decision Tree Model on Lab Data . . . . . 20
**Figure 4.4** Decision Tree Model on NASA Data . . . . . 21
**Figure 4.5** Random Forest Model on Lab Data . . . . . 21
**Figure 4.6** Random Forest Model on NASA Data . . . . . 22
**Figure 4.7** FNN Model on Lab Data . . . . . 22
**Figure 4.8** FNN Model on NASA Data . . . . . 23
**Figure 4.9** RNN Model on Lab Data . . . . . 23
**Figure 4.10** RNN Model on NASA Data . . . . . 24
**Figure 4.11** LSTM Model on Lab Data . . . . . 24
**Figure 4.12** LSTM Model on NASA Data . . . . . 25

# LIST OF TABLES

**Table 3.1** Dataset Test Conditions . . . . . 14
**Table 4.1** Comparison of Model Performance by Root Mean Squared Error . . 25
**Table 4.2** Comparison of Model Performance by Mean Absolute Error . . . . . 26
**Table 4.3** Comparison of Model Performance by $R^2$ score . . . . . 26

# ABSTRACT

## Battery State of Health (SOH) Prediction Using Machine Learning

Kıray GÜNER

Department of Mechanical Engineering
Master of Science Thesis

Supervisor: Assoc. Prof. Dr. Meral BAYRAKTAR

This thesis explores prediction of state of health (SOH) of Li-On batteries using various machine learning techniques. SOH of a battery is a critical parameter that indicates its remaining capacity and performance. Accurate SOH prediction is important for optimizing battery usage, performance and management.

The thesis begins with a comprehensive review of battery, battery aging, machine learning and SOH estimation methods. This review forms the basis for the research conducted. In the research 2 different datasets have been used. One is the NASA battery dataset available online. The other is data collected within our labs. These data was processed and normalized for usage. It is important to note that all the data used in this study was collected in laboratory environment and may not fully reflect real world data. Further study may be needed with wider datasets to train models more fitting real world usage.

In the thesis six different algorithms have been used. These are linear regression, decision tree, random forest, feedforward neural network, recurrent neural network and long-short term memory. The performance of these algorithms were evaluated using mean absolute error, root mean square and r-squared metrics. The evaluated models have been compared to identify the most suitable model for usage on SOH prediction. The results demonstrate that machine learning techniques can accurately predict battery SOH. Thus, machine learning methods can be alternative to traditional methods for SOH prediction.

Overall, this thesis contributes to the field of battery maintenance and management. The models used in this study demonstrate great potential for accurate SOH prediction. With further research and validation, this would greatly impact many fields using batteries.

**Keywords:** Battery state of health, machine learning methods, neural networks, predictive maintenance, battery management

# ÖZET

## Makine Öğrenmesi ile Batarya Sağlık Durumu Tahmini

Kıray GÜNER

Makine Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Doç. Dr. Meral BAYRAKTAR

Bu tez, çeşitli makine öğrenimi teknikleri kullanarak Li-On akülerin sağlık durumunun (SOH) tahminini araştırmaktadır. Bir akünün SOH'u, kalan kapasitesini ve performansını gösteren kritik bir parametredir. Doğru SOH tahmini, akü kullanımını, performansını ve yönetimini optimize etmek için önemlidir.

Tez, akü, akü yaşlanması, makine öğrenimi ve SOH tahmin yöntemlerinin kapsamlı bir incelemesi ile başlamaktadır. Bu inceleme, yürütülen araştırmanın temelini oluşturmaktadır. Araştırmada 2 farklı veri kümesi kullanılmıştır. Bunlardan biri çevrimiçi olarak sunulan NASA batarya veri setidir. Diğeri ise laboratuvarlarımızda toplanan verilerdir. Bu veriler işlenmiş ve kullanım için normalize edilmiştir. Bu çalışmada kullanılan tüm verilerin laboratuvar ortamında toplandığını ve gerçek dünya verilerini tam olarak yansıtmayabileceğini unutmamak önemlidir. Gerçek dünya kullanımına daha uygun modelleri eğitmek için daha geniş veri kümeleriyle daha fazla çalışma yapılması gerekebilir.

Tezde altı farklı algoritma kullanılmıştır. Bunlar doğrusal regresyon, karar ağacı, rastgele orman, ileri beslemeli sinir ağı, tekrarlayan sinir ağı ve uzun-kısa vadeli bellektir. Bu algoritmaların performansı ortalama mutlak hata, ortalama karekök ve r-kare metrikleri kullanılarak değerlendirilmiştir. Değerlendirilen modeller, SOH tahmininde kullanılmak üzere en uygun modeli belirlemek için karşılaştırılmıştır. Sonuçlar, makine öğrenimi tekniklerinin akü SOH'sini doğru bir şekilde tahmin edebildiğini göstermektedir. Böylece, makine öğrenimi yöntemleri SOH tahmini için geleneksel yöntemlere alternatif olabilir.

Genel olarak, bu tez akü bakımı ve yönetimi alanına katkıda bulunmaktadır. Bu çalışmada kullanılan modeller, doğru SOH tahmini için büyük potansiyel göstermektedir. Daha fazla araştırma ve doğrulama ile bu, akülerin kullanıldığı birçok alanı büyük ölçüde etkileyecektir.

**Anahtar Kelimeler:** Akü sağlık durumu, makine öğrenmesi yöntemleri, sinir ağları, kestirimci bakım, akü yönetimi,

# 1
# INTRODUCTION

Lithium-ion (Li-On) batteries are a widely used technology due to their high energy density and long cycle life. They can be found everywhere from cell phones to electric vehicles. However, over usage Li-On batteries deteriorate. Being able to predict the level of this deterioration is a critical task. An accurate prediction can optimize the usage to reduce the cost by manufacturers and users.

Predicting SOH can be complex and challenging due to non-linear degradation aging mechanisms behind it. Traditional models based on physical models cannot accurately portray these mechanisms. Therefore, machine learning (ML) methods have been proposed as an alternative approach.

The objective of this thesis is to develop and evaluate ML models for Li-ion battery SOH prediction using discharge duration as the indicator of battery aging. Regression models and neural networks will be used to that end. Namely linear regression, decision random forest regressors. For neural networks feed-forward and recurrent neural networks will be used.

Two different datasets have been used in this thesis to train and test the models. One is Li-On Battery Aging Dataset available online [1]. The second dataset have been collected at our labs. Both datasets have been collected at labaratory conditions and may not represent real world conditions.

The thesis provides an overview of the literature in Lithium-Ion batteries, machine learning and state of healt (SOH) prediction. The methodology describes the data and the features. The results section discusses the performance of each model and their comparison. Finally the conclusion section summarizes the findings and future work directions.

# 2
# Literature Review

## 2.1 Lithium-Ion Cell Basics

Lithium-ion batteries have ever rising importance in today's world with the rise of EVs. Their ability to store regenerated power [2] allows them to hold this importance. Main components of a lithium-ion battery are: anode, cathode, electrolyte and separator. The electricity is produced via chemical potential difference between negative electrode (anode) and positive electrode (cathode). This potential difference determine what the open circuit voltage (OCV) of the cell will be. During charging and discharging lithium ions passes through the electrolyte between the anode and cathode [3]. The final main component, separator, serves to sever contact between the negative and positive electrodes [3].

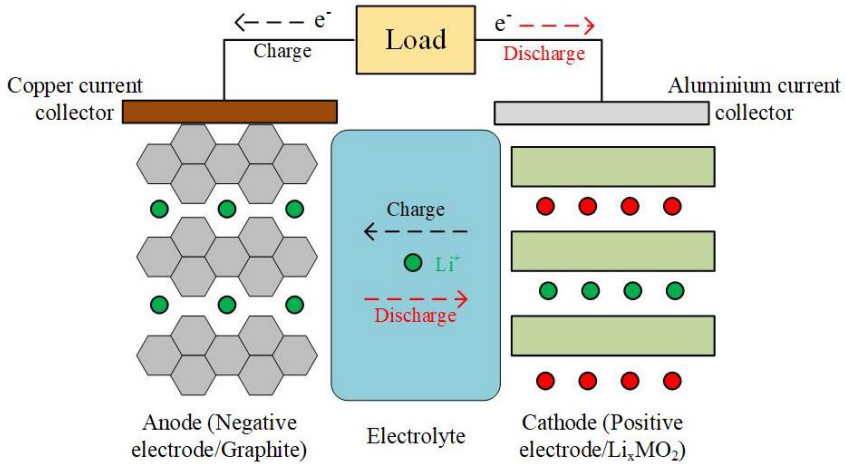



**Figure 2.1** Lithium-Ion Battery Operation [4]

Externally connecting the two electrodes creates a path for electrons to move between anode and cathode. During charging this movement is towards to anode and vice versa for discharging. While the electrons are moving through the external circuit, lithium-ions move through the electrode, creating a balanced potential. This way chemical energy can be converted to electrical energy in a reversible fashion [3]. This

working principle is also illustrated in the figure 2.1 [4].

## 2.2 Components Of A Lithium-Ion Battery

This section will give more detail about each main components of a lithium-ion battery. These components are electrodes, electrolyte, and separator.

### 2.2.1 Electrodes

Electrodes are made up of active materials, additives and binders. Active materials are used to ensure prolonged cycling by accommodating lithium. [5]. During charging and discharging lithium ions move between electrodes. Additive are used to increase conductivity to improve the efficiency of this movement. [4]

The anode is covered by a homogeneous and thin layer of active material such as copper. This cover provides essential organic electrolytes. [3], [5]. As a result, the manufacturing process of anode such as mixing and covering determine the battery properties. Capacity and ageing mechanism of the cell is among these properties. [4]. Minor differences during the manufacturing process can lead to variation among battery cells. Carbon based materials tend to be used for anodes with graphite being the most popular choice.

The cathode is made up of layered oxides. Some of the oxides used commercially are: lithium magnase oxide (LiMn2O4), lithium iron phospate (LiFePO4). Alternatively mixed metal oxides like nickel manganese cobalt oxide (NMC) are also used. Compared to lithium based cells, cobalt based cells have higher capacity however they are toxic and expensive. A cheaper option is manganese based cells but they have capacity fade issue. As a result LiFePO4 cells are the cheap and eco-friendly solution [4].

### 2.2.2 Electrolytes

An electrolyte is a mix of liquid solvent with lithium salts. The most common lithium salt is lithium hexafluorophosphate (LiPF6). Electrolytes need to have the following qualities [6]:

i) Good ionic conductivity

ii) Good electronic insulation

iii) Electrochemically does not oxidize or lose electrode surface

iv) No reactivity with other components of the cells

Recently manufacturers have been trying to replace liquid electrolytes with solid state ones. The solid-state electrolytes are safer as the are non-flammable and they improve energy density as well. [7]. However solid-state electrolytes still have som way to go before becoming commercialized as they lack of ionic conductivity and coulombic efficiency. They are also more reactive to environmental exposure [7], [8].

### 2.2.3 Separators

The purpose of the seperator component of the battery cell is the seperate the anode and the cathode. Seperator achieves this by creating a physical barrier that stops the cell from short cicuiting. While the seperator stops short circuiting it allows lithium ions to pass through during charging and discharging. A seperator must have the following qualities [3]:

i) Thermal, chemical and mechanical stability

ii) Good electronic insulation

iii) Uniform thickness

iv) Low electrical resistance and high porosity

The most used type of seperator is polyolefins. Some polyolefins types are ployethylene and polypropylene [3]. Seperator are defined as monolayer or multilayer based on the amount of layers they have [9]. TMonolayer seperators is that they have higher stabily but at the cost of short circuiting the cell at higher temperatures. Multilayer do not have this problem since they melt at higher temperatures, thereby blocking any conductivity that can lead to a short [9]

## 2.3 Battery Degradation and End of Life

Over their lifespan batteries lose their capacity, power or both. This is also seen with the decline of functionality and reliability of the battery. As batteries tend to be used as part of a system aging of the battery can accelerate aging of the other components within the system [10]. Since there are multiple factors that affect battery degrradation, the mechanism behind can be complex. However these mechanisms can be summarized by chemical and mechanical degradation.

### 2.3.1 Chemical Degradation

Main chemical degradation types in li-on batteries are [10]:

i) Electrolyte decay

ii) Solid electrolyte interface (SEI) growth

iii) Binder decay

iv) Active material loss

v) Gas venting

### 2.3.2 Mechanical Degradation

Movement of the lithium ions within the cell cause volume changes. This repeated volume increase and decrease cause stress for the cell. The tensile stress then leads to cracks within the cell, loss of contact between anode and cathode or with current collectors [11]. Another type of mechanical degradation reason is the loss of porousness on the seperator resulting in loss of lithium mobility.

## 2.4 Degradation in cell components

This section will give detail on degradation mechanism of each battery cell component. These are anode, cathode and separator degradation.

### 2.4.1 Anode degradation

One of the main reasons of anode degradation is the relations between the anode and electrolyte [12]. The anode deteriorates as it reacts to electrolyte. This deterioration leads to formation of a thin layer called solid electrolyte interface (SEI). The SEI layer initially helps protect the anode from corrosion. However the growth continues over time and since the SEI layer grows by consuming lithium ions, the growth of this layer leads to increase of resistance [4]. Although the growth slows down compared to initial stages of the battery's lifespan [13]. SEI growth is considered to be the main reason behind anode degradation and increase in internal resistance of the battery.

Other types of anode degradation are lithium plating and current collector corrosion. Lithium plating occurs on the surface of the anode when potential of the lithium is equal or higher than anode as can be seen on figure 2.2 [14]. On the other hand,

current collector corrosion occurs when the current collector reacts to electrolyte components or if voltage of the anode is too high [15].

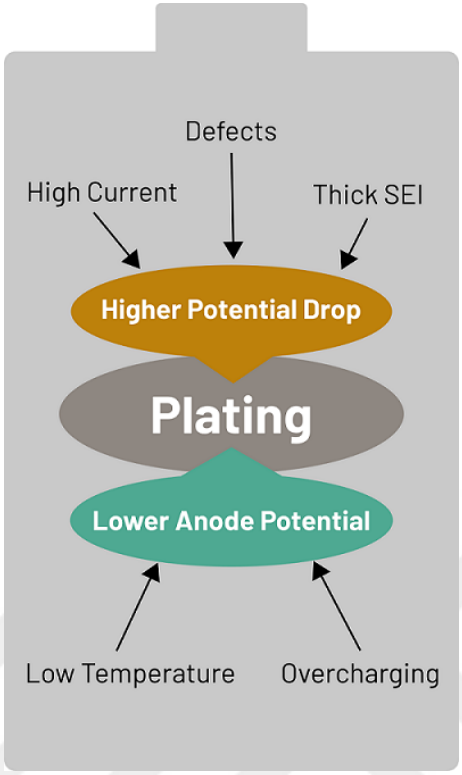



**Figure 2.2** Lithium Plating [14]

### 2.4.2 Cathode degradation

Several factors can cause cathode degradation of lithium-ion batteries, such as [15]:

i) Aging of active material

ii) Degradation of binder or conductor agents

iii) Current collector corrosion

iv) Electrolyte oxidation

It's important to note that these reactions are not necessarily independent. One or more of them can happen at the same time. Other root causes of capacity reduction are cycling leading to structural changes in active materials, surface film on cathode undergoing modifications or chemical reactions such as decomposition or dissolution.

Degradation in the cathode electrode, much like the anode electrode, depends on the state of charge (SOC) and the cycling conditions. When lithium ions are inserted or

extracted, it causes increase or decrease to the volume of the positive active material, which in turn can cause strain on the components of the cathode.

The process of inserting or extracting lithium ions can result in a phase shift within the cathode oxide, causing distortion in the crystal lattice and generating mechanical strains [13]. These strains are particularly pronounced at the boundaries between different phases, leading to a lack of coherence and potentially leading to cracking of nanoparticles within the cathode.

Capacity degradation in the cathode can also occur due to the dissolution of active materials, especially within cathodes based on Mn (manganese) [10]. The dissolution of Mn into the electrolyte has a two effects on cell aging. Firstly, it leads to the loss of active material, reducing the overall capacity of the cathode. Secondly, the dissolved active material can migrate to the anode and deposit on its surface. This in turn, increases the resistance of the anode. This deposition can further contribute to the degradation and overall performance decline of the battery.

The isolation of active particles within the cathode electrode is identified as one of the primary reasons for battery deterioration [16]. This mechanism can be summarized with two main factors. The active material in the cathode electrode can experience formation of cracks or the binder can undergo fracture [10].

As a result, the isolation of the active material is caused by mechanical degradation in binder and in active material also by chemical degradation in the binder itself. These are contributing factors to the overall degradation of the battery.

### 2.4.3 Separator degradation

The deterioration of the separator in lithium-ion batteries can be attributed to several factors, such as [10]:

i) Lithium dendrite growth

ii) Structural reduction

iii) Blocking of passageways

The chemical properties of separator materials used in lithium-ion batteries are crucial to ensure their inertness and non-interference with electrical output. However, the physical properties of separators have a significant impact on cell performance and

safety [17]. To maintain optimal functionality, separators should exhibit uniformity in their porous structure, low shrinkage, and low resistance.

The resistance of the separator to penetration, as well as factors such as separator thickness, porosity, and toughness, may vary depending on the desired cell characteristics and functions [17].

One approach to increase cell capacity is to use thinner separators. However, thinner separators have certain drawbacks, such as reduced electrolyte volume and diminished mechanical strength. Thinner separators may not be able to accommodate as much electrolyte, potentially affecting battery performance.

One of the reasons for power fading originating from the separator is the inherent increase in separator ionic impedance [18]. This increase in ionic impedance is highly dependent on the number of cycling and temperature. The rise in ionic impedance is caused by the clogging of separator pores, which occurs due to the presence of electrolyte decomposition products. The acceleration of this process at high temperatures further exacerbates the clogging and leads to an increase in ionic impedance, contributing to power fading.

## 2.5 Battery State of Health (SOH)

The definiton of state of health (SOH) does can depend on battery company, control company or the application. However all these definitions refer to current state health of the battery with respect to initial state of health of the battery [19]. This relationship can be formed on the basis of battery's internal resistance, capacity, voltage and so on.

In this study, SOH was defined based on full capacity of the battery, based on the formula 2.1.

$$SOH = \frac{C_{fullcharge}}{C_{design}} \tag{2.1}$$

On equation 2.1, $C_{fullcharge}$ refers to maximum charge that can be stored in the battery at any given time. $C_{design}$ refers to the maximum charge capacity of the battery at it's brand new condition.

It is worth mentioning that SOH is a measure that affects the battery functionality. As such the calculation of SOH is also built into battery management systems [19].

## 2.6 Machine Learning

In today's digital era, computers play a crucial role in our daily lives, encompassing various devices such as mobile phones and supercomputers. Researchers have devoted significant efforts to developing new techniques that leverage the computational power and speed of computers to simplify routine tasks and reduce human effort. This has given rise to the field of Artificial Intelligence (AI), which aims to enable machines to perform tasks that were previously carried out by humans [20].

Machine Learning (ML), a subset of AI, focuses on the study of algorithms and techniques that utilize computers' ability to recognize patterns. Training machine learning algorithms is what allows them to recognise these patterns. These patterns recognized by machine learning algorithms may or may not be visible to a human eye. After training phase is complete, these algorithm can then be used for decision making with minimum human involvement. To verify the ability of ML algorithms "test data" is used.

The learning done in machine learning happen in 3 different types [21]:

i) Supervised learning

ii) Unsupervised learning

iii) Reinforcement learning

Supervised learning algorithms, require the availability of explicit labels in the training data [21]. These algorithm recognise patterns in the data by looking at the labels and finding an input output relationship based on these labels. Some of the most common areas supervised learning is used are regression and classification. Common examples of supervised learning can be listed as linear regression, naive bayes, decision tree algorithms.

On the other hand, unsupervised learning algorithms, as described in the same reference [21], do not rely on labeled instances in the data. Instead, they aim to identify significant features and hidden patterns within the data. These algorithms can be utilized for various tasks such as finding anomalies and outliers in the data. An example of an unsupervised learning algorithm can be K-means clustering.

Reinforcement learning (RL), is different from supervised learning method by not needing labeled data. Instead it learns within an environment by receiving rewards or penalties based on its performance. This way the algorithm learns about

the environment by striving to maximize its rewards. RL algorithms have found applications in fields such as robotics and autonomous driving.

Reinforcement learning differs from supervised learning in not needing labelled input/output pairs to be presented, an

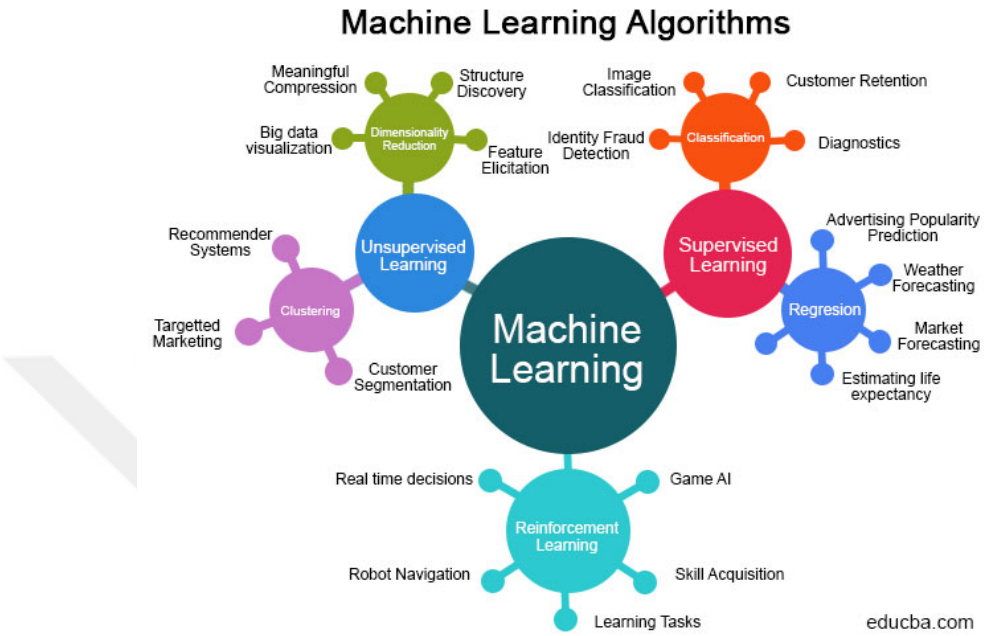



**Figure 2.3** Categories of machine Learning [22]

Figure 2.3 [22], provides an overview of the different types of learning types and the tasks that can be accomplished with them. This study follows regression path of supervised learning since we have we have labeled data to train on.

## 2.7 Artificial Neural Networks

Artificial Neural Networks (ANN), are a popular class of machine learning models used for handling nonlinear data [23]. Neural networks are made up of nodes called "neuron"s which applies a mathematical function to the input it receives, transforms it, and produces a scalar output. These neurons are named after their inspiration that is the biological neurons in the human brain. The concept of artificial neurons was first introduced by McCulloch and Pitts in their work [24], followed by Frank Rosenblatt's contributions in 1957 [25], where he demonstrated that artificial neurons could learn from data. Rosenblatt also introduced the idea of Perceptron, a supervised learning algorithm that allowed artificial neurons to determine correct weights from training data automatically. Neural networks consist of interconnected neurons arranged in layers, and information flows starts from input layer and ends at the output layer

[23]. The neurons are connected by numerical connections also called weights. These weight are initialized randomly and are than determined during training. As a result, a neural network can be reimagined as a graph of weighed lines connecting nodes of artificial neurons. These weights act to direct the flow. Neural networks can be classified in two types:

Feed-Forward Neural Networks: This type of network, such as the popular Multi-Layer Perceptron (MLP), consists of layers of neurons with unidirectional connections. Each layer outputs scalar values to the next layer, and the layers are independent of the previous layer's state.

Recurrent Neural Networks (RNN): RNNs are used to analyze sequential data. They have the ability to produce sequences. Due to their feedback mechanism neurons in an RNN layer can adjust weights based on the previous layers and their states. In artificial neural networks (ANNs) the weight values are determined over iterations. With every iteration modifications are made as the neural network recognizes patterns in the data. Long-short term memory (LSTM) is a special type of RNN that can process entire sequence of data rather than a single point. LSTM also has a short and long term memory thanks to it's memory lane and forget gate [26].

The developments seen in multi-layer feed-forward networks and the back-propagation algorithms [27], [28], ANN models have gained noteworthy popularity. They have been successfully applied to solve complex linear and nonlinear problems, leading to the emergence of a sub-field called Deep Learning.

## 2.8 Performance Metrics

Three common performance metrics have been used during this thesis. They are root mean squared error (RMSE), mean absolute error (MAE) and R-squared. Each of them have been explained in this section.

### 2.8.1 Root Mean Squared Error

Mean squared error (MSE) is a method to assess average of the squared error between observed and predicted values. For a model with zero errors mean squared error is also zero. The formula for mse is given in 2.2.

$$MSE = \frac{\Sigma(y_i - \hat{y}_i)^2}{n} \tag{2.2}$$

Where $y_i$ is the $i^th$ and $\hat{y}_i$ is the predicted value. n being the number of observations

Root mean squared error (RMSE) is simply square root of MSE. It is commonly used because the units of RMSE matches the units of the values in the dataset.

### 2.8.2 Mean Absolute Error

In forecasting a value with regression, an error is the unpredictable difference between the data and the observation. Errors in forecast is in the same scale with the data. As a result accuracy can be measured by error rate. Mean absolute rate is one of the most used methods to determine error rate. Mean absolute error is calculated by equation 2.3 where $e_t$ is the error [29].

$$MAE = mean(|e_t|) \tag{2.3}$$

### 2.8.3 R-Squared

R-squared is used to measure how well prediction fits of a regression model. It tells the relationship between dependent and independent variable. It tells this by evaluating comparing data to the fitted regression line. R-squared is calculated by equation 2.4 [30].

$$R^2 = \frac{\text{Variance explained by the model}}{\text{Total variance}} \tag{2.4}$$

## 2.9 Related Studies in Literature

Battery Health prediction is an emerging field with many various research published. It is possible to find studies that predict remaining useful life (RUL) with deep belief network and relevance vector machine with acceptable accuray [31]. In another study support vector regression-particle filter (SVR-Pf) was used to predict RUL [32]. In this study it is shown that capacity is reversely proportional to charge and discharge cycle count. The study presents a robust algorithm but one which can only work in a narrow 75% to 85% limit. Moreover, a study done by Liu et al. [33] shows a data-driven Gaussian process regression (GPR) for SOH estimation. The GPR model is trained with cycle number and capacity information. The predictions does not take varying conditions into account. On another study using GPR [34], charging curve is used to train the model instead. The study extract four features from charging curves. These features are time in constant current mode, duration in constant voltage, constant

charging slope and constant current slope. Model predictions are accurate except that error increases during testing. This may be due to low optimization or over-fitting.

# 3
# Methodology

## 3.1 Data Collection and Preprocessing

In this section how the data was collected and prepared for usage will be explained.

### 3.1.1 Data Sources

In this study, two different datasets have been used. One of the datasets comes from NASA Li-On battery aging datasets [1]. Dataset relating to battery no. 6 has been used due to its similarity to the second dataset. The second dataset was collected within our labs. The test conditions for both of the datasets are given in table 3.1

| Data Source | Constant Charge Current(A) | Charge Cut-Off Voltage(V) | Discharge Cut-Off Current(A) | Discharge Voltage(V) |
|---|---|---|---|---|
| NASA Battery Dataset Battery No.6 | 1.5 | 4.2 | 2.0 | 2.5 |
| Lab Data | 0.8 | 4.1 | 1.35 | 3.0 |

**Table 3.1** Dataset Test Conditions

As it can be seen in table 3.1, batteries were charged at constant charge current until cut-off voltage was reached. Afterwards charging continued at constant voltage at the cut-off voltage. During discharge different constant current was used but the principle was the same. Discharge with constant voltage until the cut-off voltage is reached.

### 3.1.2 Data Cleaning

To clean the data outlier removing has been performed. Outliers was removed to remove their effect on the analysis, improve the quality and reliability of the data. To remove the outliers, data between 1 and 99 percentile has been selected for processing with rest being removed. The jumps that can be seen on lab data on results section has been kept to protect authenticity. These jumps were not present on NASA data.

## 3.2 Feature Extraction

### 3.2.1 Feature Selection

For this study voltage vs. duration have been selected as a feature as there are examples [35], [34] in literature that showed it to be promising candidate. Upon analysis, there was clear connection with SOH and the duration for voltage to have a certain change.

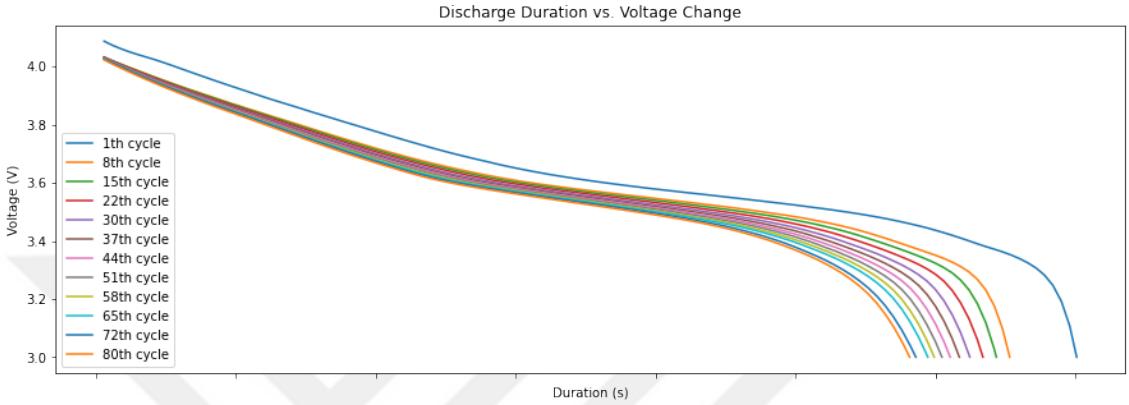



**Figure 3.1** Lab Data Voltage vs. Duration

On figure 3.1 and 3.2 it can be seen that as the cycle number increases (i.e. the battery ages) the duration for voltage to drop gets shorter. This is inline with the theory because the capacity of the battery is reduced as it ages, thus it takes less amount of time to charge or discharge. It is worth noting that this drop in duration is not necessarily linear. For example on NASA data $16^{th}$ cycle takes less time to discharge than $31^{s}t$ cycle as it can be seen in figure 3.2.

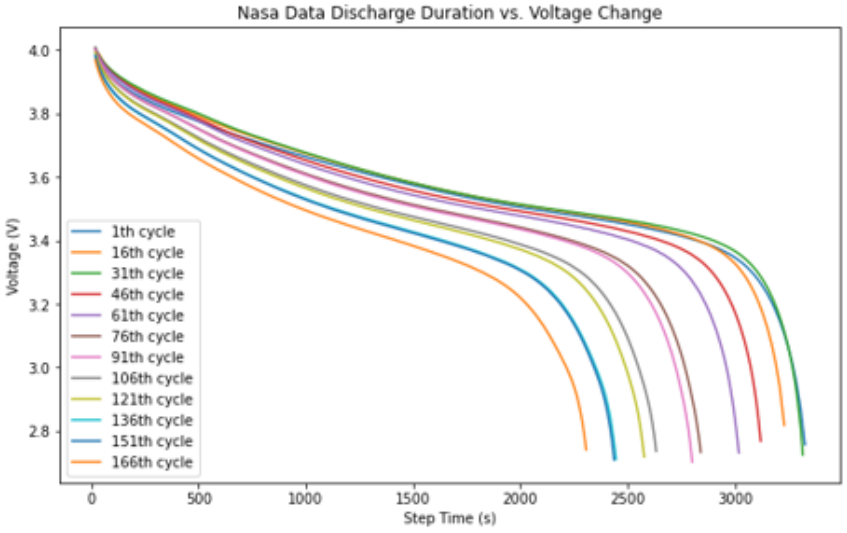



**Figure 3.2** NASA Data Voltage vs. Duration

However this change in duration follows change in SOH. THis can be seen in figures 3.3 and 3.4. These heatmaps show that the change in duration is correlated with the change in SOH and duration of voltage drop is a suitable feature.

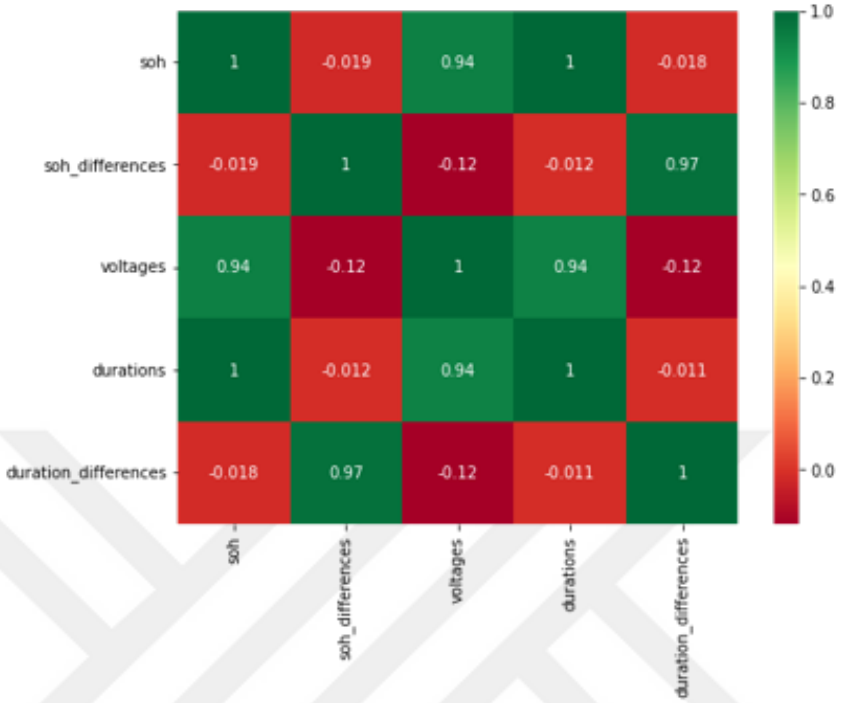



**Figure 3.3** Lab Data Correlation Heatmap

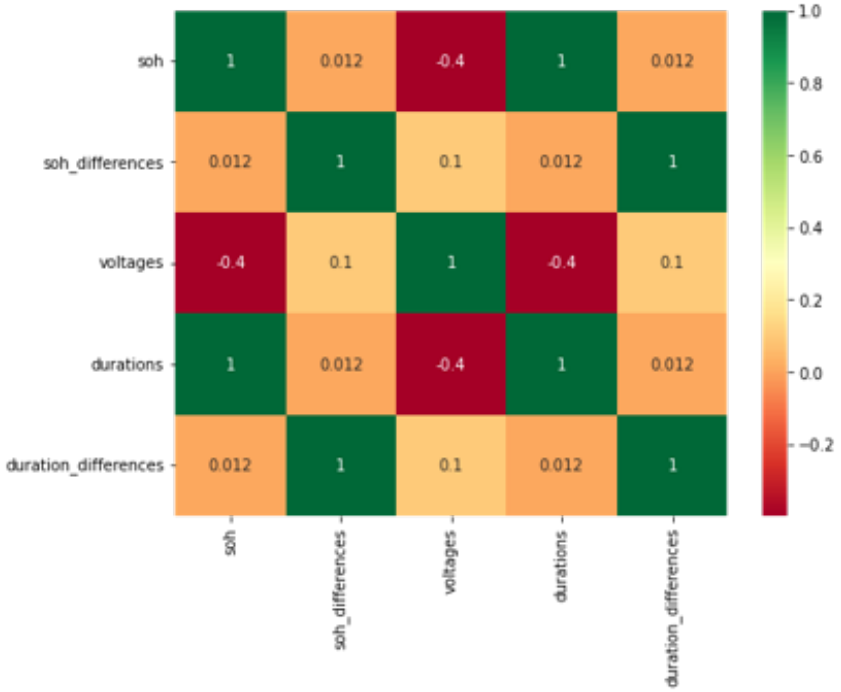



**Figure 3.4** NASA Data Correlation Heatmap

### 3.2.2 Feature Engineering

To use the selected feature for training, they were normalized first. To do the normalization, minmaxscaler function of Sklearn library was used [36]. The formulation behind the normalization has been given in equations 3.1 and 3.2.

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

$$X_{scaled} = X_{std} * (max - min) + min \tag{3.2}$$

Here X is defined as a dataframe. In this study a column of discharge duration per cycle has been used. As a result minimum and maximum values of that column correspond to $X_{max}$ and $X_{min}$.

# 4
# Results and Discussion

This chapter discusses the model performance on both of the datasets used and how the models compare to each other.

## 4.1 Model Results

In this section six different models have been trained and tested on the two datasets available. The models are linear regression, decision tree, random forest, feed-forward neural network, recurrent neural network and long-short term memory. For each model the resulting visualisation of how the models performed on each prediction on each dataset is given.

### 4.1.1 Linear Regression

In this case study, linear regression model was used on both datasets. The results can be observed at figures 4.1 and 4.2. To achive these result the input feature (discharge duration) and output SOH data was used to train the model until the specified cycle. After the model is trained it was asked to predict the future SOH based on the change in discharge duration.

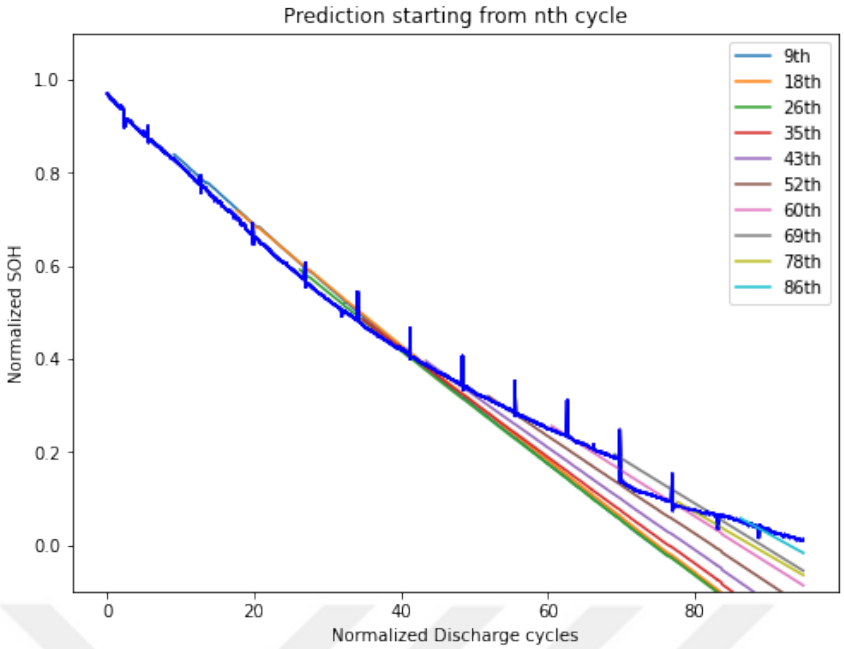



**Figure 4.1** Linear Regression Model on Lab Data

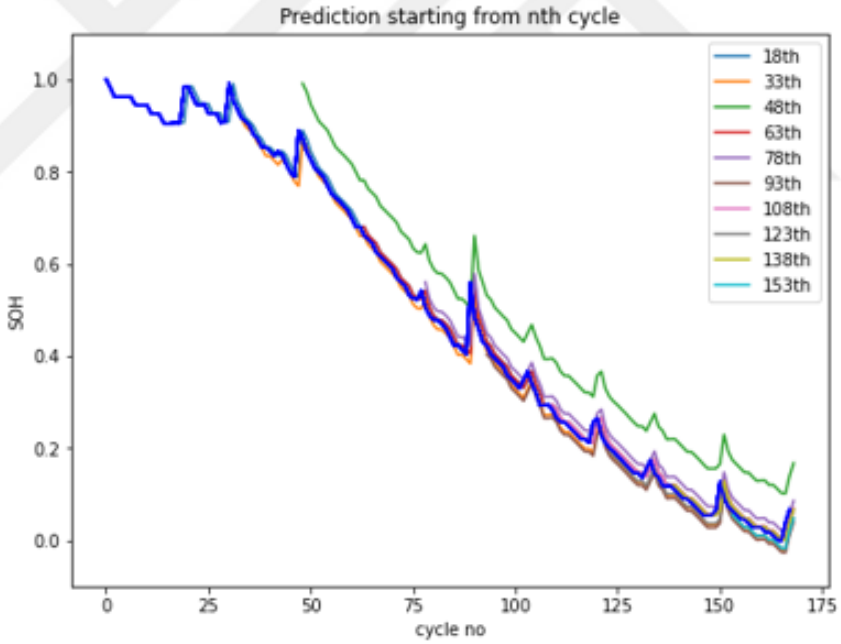



**Figure 4.2** Linear Regression Model on NASA Data

Figures 4.1 and 4.2 show that the same model was able to capture the trend of the data better with NASA dataset. This may be due to test conditions. In our lab data testing was automated for certain number of cycles. However in some cases the automation would end but restarting the test would take some time. This caused polarization within battery and the jumps that can be seen on the lab data. Another possible reason for prediction on NASA dataset to be able to follow the trend better is the trend in the

data itself. In conclusion, why there is a difference in the performance of same model on two different datasets needs further research.

### 4.1.2 Decision Tree

In this case study the same process with linear regression has been applied with a decision tree model as it can be seen on figures 4.3 and 4.4. Unlike linear regression decision tree needs higher amount of cycles before starting to perform at acceptable levels.

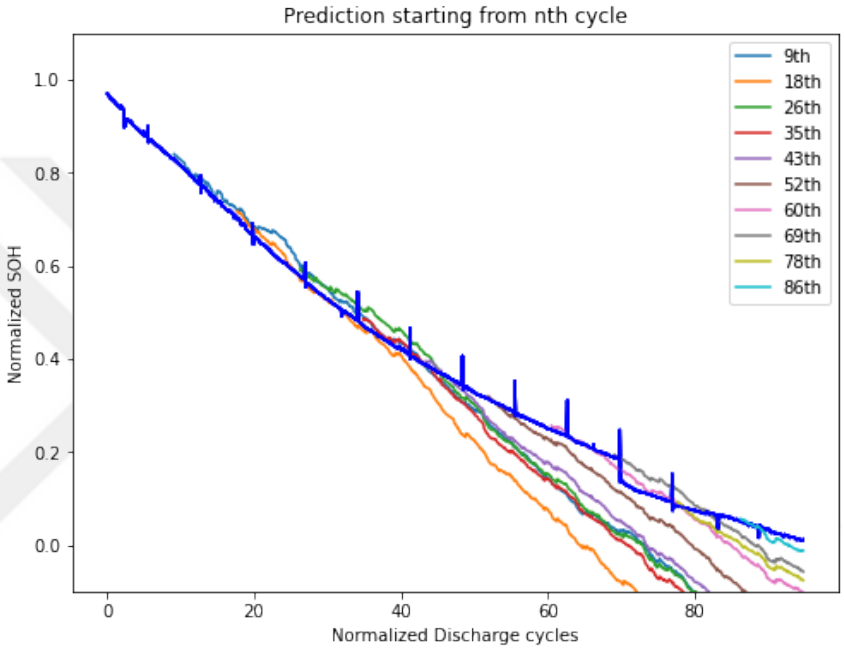



**Figure 4.3** Decision Tree Model on Lab Data

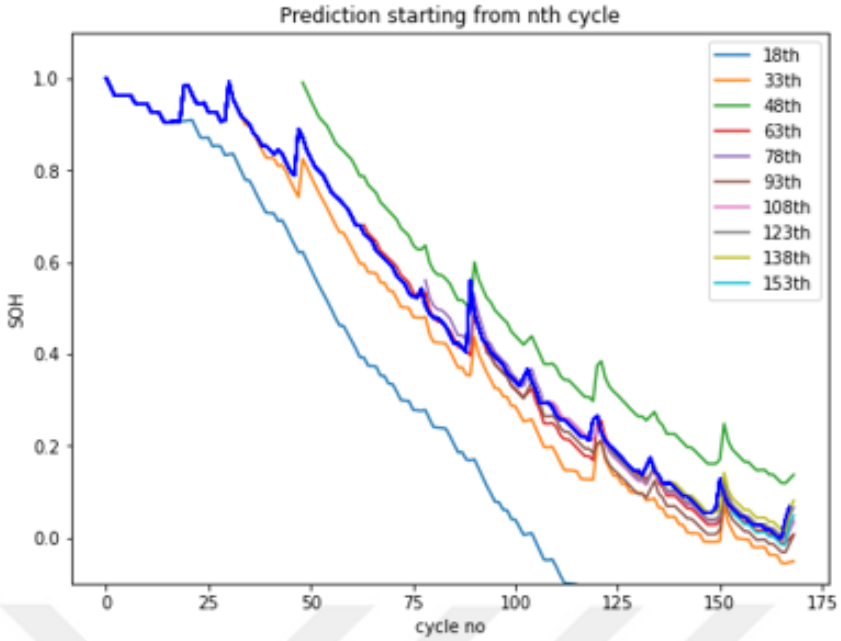



**Figure 4.4** Decision Tree Model on NASA Data

### 4.1.3 Random Forest

On figures 4.5 and 4.6 the results for random forest model can be seen. The results are similar to decision tree but converges more effectively.

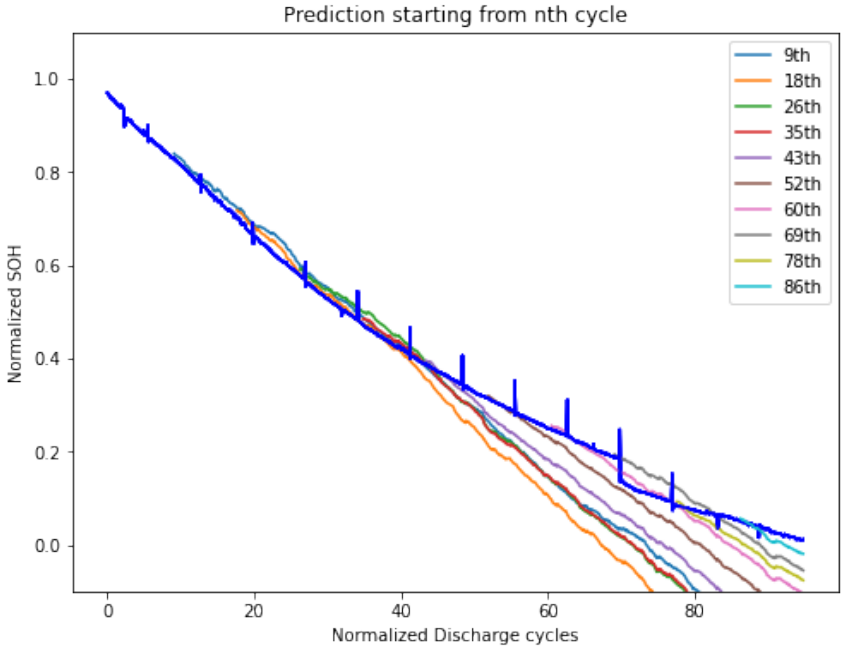



**Figure 4.5** Random Forest Model on Lab Data

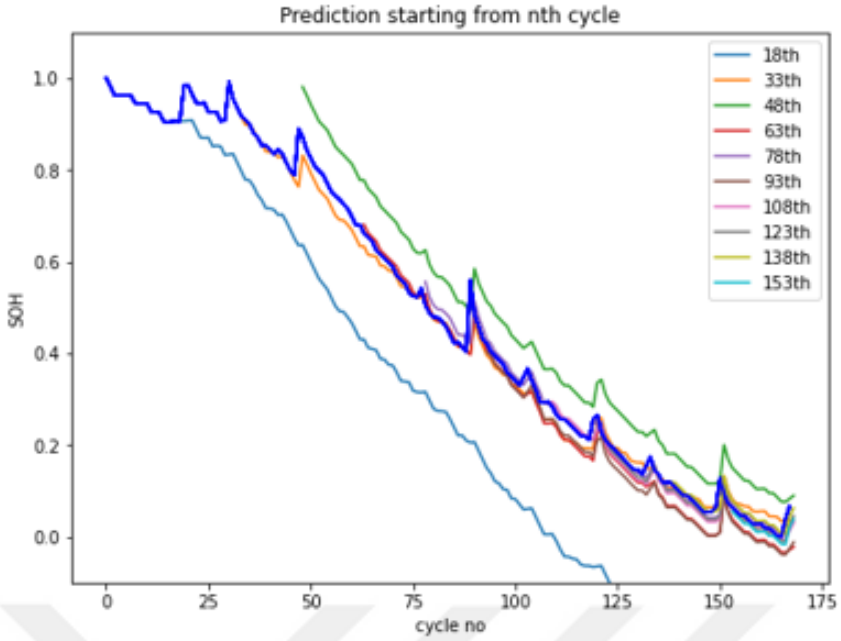



**Figure 4.6** Random Forest Model on NASA Data

### 4.1.4 Feedforward Neural Network

Results for 64 noded feedforward neural network performance is shown on figures 4.7 and 4.8. The results show that it takes longer to train FNN

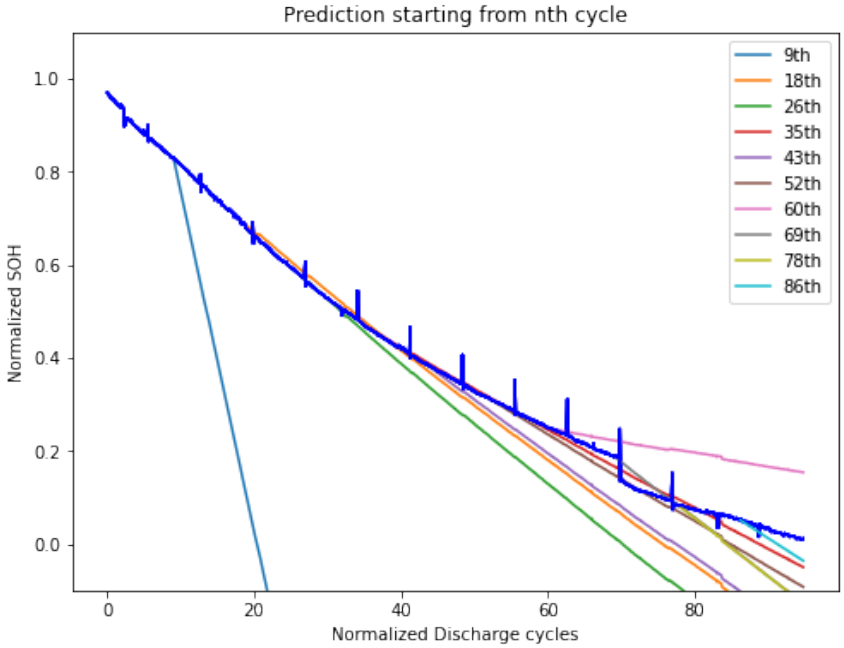



**Figure 4.7** FNN Model on Lab Data

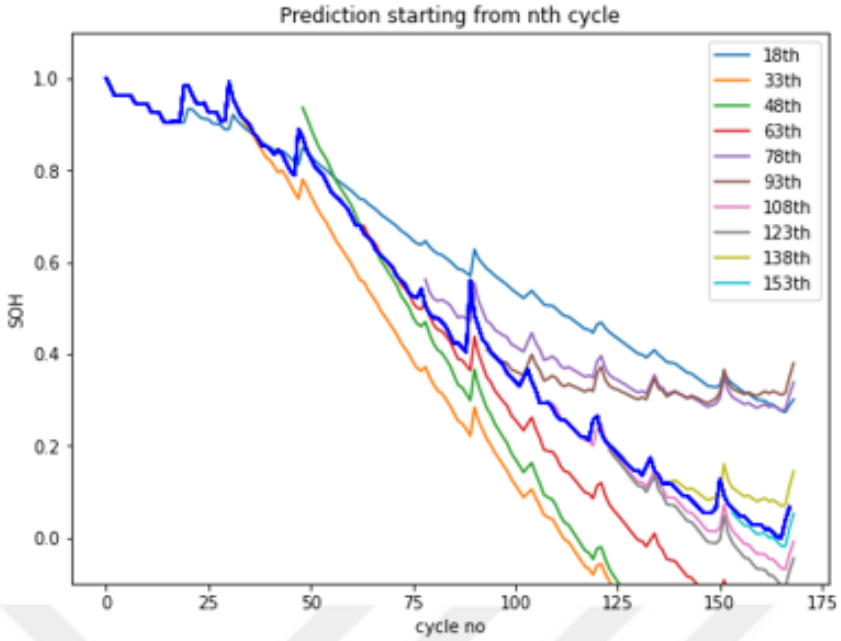



**Figure 4.8** FNN Model on NASA Data

### 4.1.5 Recurrent Neural Network

As it can be seen on figures 4.9 and 4.10 using a simple 64 noded RNN instead of FNN improves the performance of the models.

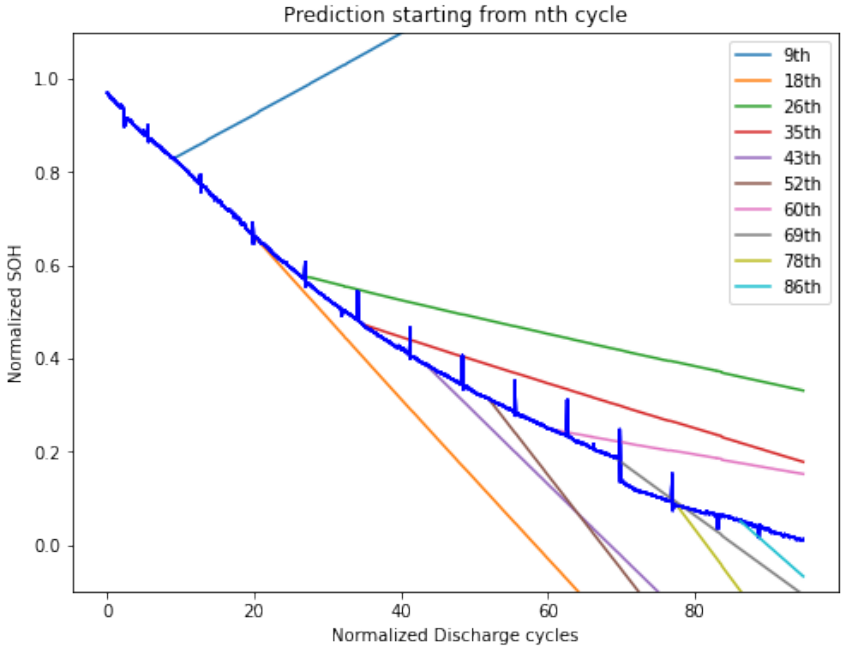



**Figure 4.9** RNN Model on Lab Data

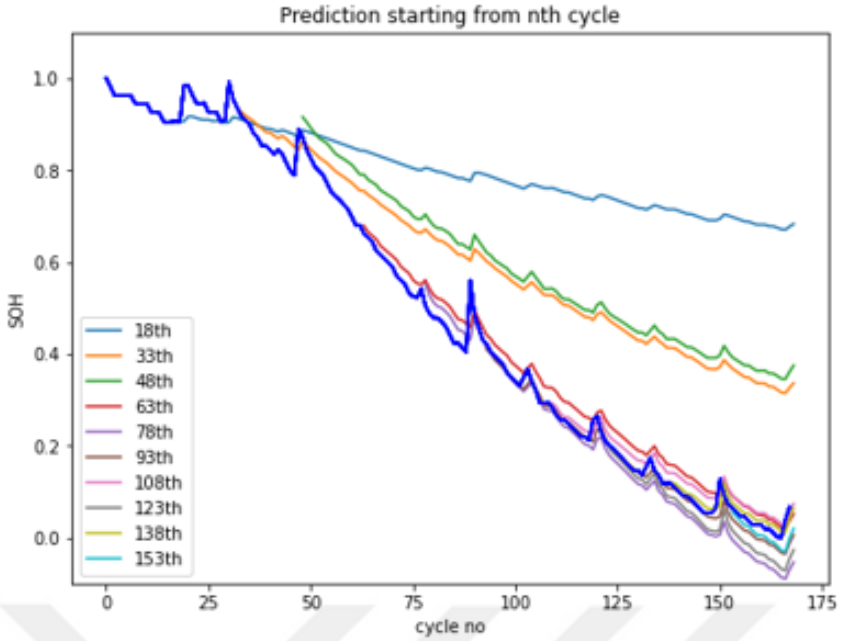



**Figure 4.10** RNN Model on NASA Data

### 4.1.6 Long Short-Term Memory (LSTM)

In this section a stacked LSTM have been used. The first LSTM layer is built with 50 nodes and outputs to a 25 noded second layer.

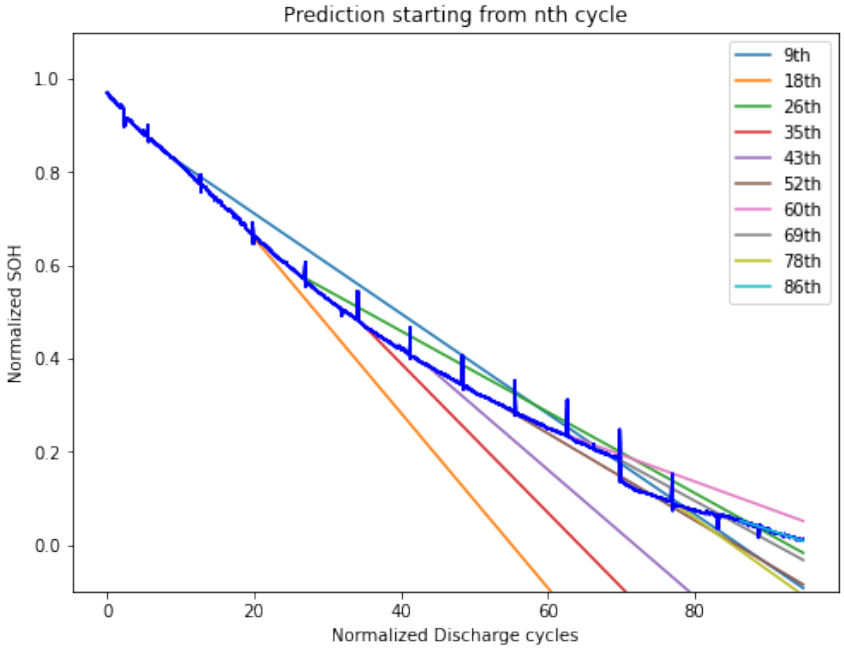



**Figure 4.11** LSTM Model on Lab Data

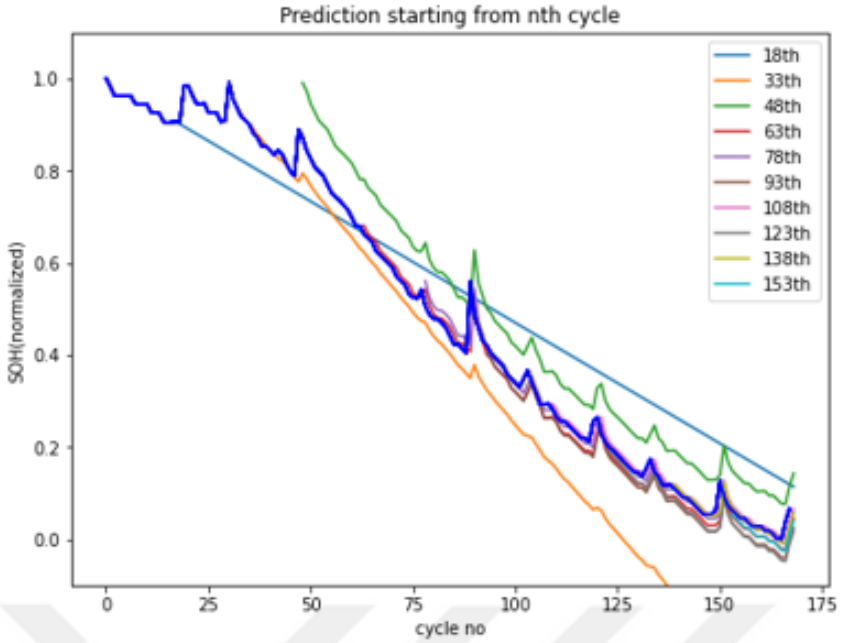



**Figure 4.12** LSTM Model on NASA Data

## 4.2 Comparison Of The Models

Model performances have been summarized on tables 4.1, 4.2 and 4.3. These tables show the performance of each model on the specified performance. The prediction to measure performance captures last 10% of the data (i.e. 90% of the data used to train with last 10% used to test). Since this is a timeseries data, no validation set has been used.

| Model | NASA Data Result | Lab Data Result |
|---|---|---|
| Linear Regression | 0,0436 | 0,1153 |
| Decision Tree Regressor | 0,1166 | 0,0510 |
| Random Forest Regressor | 0,1100 | 0,0469 |
| Feedforward Neural Network (FNN) | 0,0283 | 0,0173 |
| Recurrent Neural Network (RNN) | 0,0245 | 0,0027 |
| Long-Short Term Memory (LSTM) | 0,0656 | 0,0012 |

**Table 4.1** Comparison of Model Performance by Root Mean Squared Error

| Model | NASA Data Result | Lab Data Result |
|---|---|---|
| Linear Regression | 0.1014 | 0,0423 |
| Decision Tree Regressor | 0.0922 | 0,0417 |
| Random Forest Regressor | 0.0913 | 0,0416 |
| Feedforward Neural Network (FNN) | 0.0212 | 0.0177 |
| Recurrent Neural Network (RNN) | 0.019 | 0.0024 |
| Long-Short Term Memory (LSTM) | 0.0644 | 0.0008 |

**Table 4.2** Comparison of Model Performance by Mean Absolute Error

| Model | NASA Data Result | Lab Data Result |
|---|---|---|
| Linear Regression | -221.877 | -406,793 |
| Decision Tree Regressor | -224.468 | -582,914 |
| Random Forest Regressor | -197.663 | -488,178 |
| Feedforward Neural Network (FNN) | -2.957 | -208.746 |
| Recurrent Neural Network (RNN) | -2.179 | -3.935 |
| Long-Short Term Memory (LSTM) | -21.387 | $7.98x10^{-5}$ |

**Table 4.3** Comparison of Model Performance by $R^2$ score

Based on these performance metrics it can be said that models perform at acceptable levels (less than 1) when it comes to root mean square or mean absolute error. However it is intriguing as to why $R^2$ value is negative. The negativity of this value indicates that the trend in the prediction data fails to match the trend in test dataset. Only LSTM achieves positive r-squared value. With other metrics considered as well, on lab data, LSTM was the most successful model. However on NASA dataset RNN performed better.

# 5
# Conclusion and Future Work

## 5.1 Conclusion

This thesis investigated the use of various machine learning techniques in prediction of state of health (SOH) of Li-On batteries. SOH of a battery is a critical parameter indicating remaining battery capacity and performance. An accurate prediction is a vital part of optimizing battery usage to extend battey lifespan.

The thesis began by reviewing the background information and current literature status on li-on batteries, battery aging, machine learning and SOH prediction. This provided an understanding to guide the research in the thesis.

For this thesis two different datasets was used. One of which was a dataset collected in our labs and the other NASA battery aging dataset [1]. Both of these data contained cycle, capacity, voltage, current and some other information to use. This data was extracted and normalized for usage in predictive models.

Next, six different machine learning models have been trained and tested on the aforementioned data. These models included linear regression, decision tree, random forest, feedforward neural-network, recurrent neural network and long-short term memory. These models were used to find discharge duration and SOH. Afterwards models were evaluated based on root mean square error (RMSE), mean abosulute error (MAE) and r-squared metrics

Our finding show that the models can accurately predict SOH. The MAE and RMSE scores of each model show that error is at acceptable levels for the models performance. However r-squared metric tells that models can struggle when it comes to predicting the trend of the data.

Overall this thesis contributes to growing research interest in battery health monitoring and predictive maintenance. The models used show great potential to be used on the field for proactive decision making and accurate battery management.

With further research this approach can lead the way to much more optimized battery usage and lifespan.

## 5.2 Future Work

It is important to note that both of the datasets used in this study have been gathered in laboratory environment and may not represent real world data. For example, both data have been collected at near constant temperature and with full charging and discharging at every cycle. For a model to work well in real world application, the model needs further training with real world data. In addition the models were trained with only discharge duration. Further research may be needed with more complex models that takes other parameters such as temperature, charging, voltage difference into account.

# REFERENCES


[1] *Li-on battery aging datasets*. [Online]. Available: `https://data.nasa.gov/dataset/Li-ion-Battery-Aging-Datasets/uj5r-zjdb`.

[2] H. Chen, T. N. Cong, Y. Wei, T. Chunging, L. Yongliang, D. Yulong, "Progress in electrical energy storage system: A critical review," *Progress in Natural Science*, vol. 19, pp. 291–312, 2009.

[3] M. M. Kabir, D. E. Demirocak, "Degradation mechanisms in li-ion batteries a state-of-the- art review," *International Journal of Energy Research*, vol. 41, pp. 1963–1986, 2017.

[4] I. Babaeyazdi, "State estimation of li-ion batteries using machine learning algorithms," Ph.D. dissertation, York University, 2923.

[5] M. R. Palacín, "Understanding ageing in li-ion batteries: A chemical issue," *Chemical Society Reviews*, vol. 47, pp. 4924–4933, 2018.

[6] K. Xu, "Electrolytes and interphases in li-ion batteries and beyond," *Chemical Reviews*, vol. 114, pp. 11 503–11 618, 2014.

[7] A. M. Abakumov, S. S. Fedotov, E. V. Antipov, J. M. Tarascon, "Electrolytes and interphases in li-ion batteries and beyond," *Nature Communications*, vol. 11, pp. 1–14, 2020.

[8] Y. K. Sun, "Promising all-solid-state batteries for future electric vehicles," *ACS Energy Letters*, vol. 5, pp. 3221–3223, 2020.

[9] A. A. Heidari, H. Mahdavi, "Recent development of polyolefin-based microporous separators for liion batteries: A review," *Chemical Record*, vol. 20, pp. 570–595, 2020.

[10] D. E. Demirocak, B. Bhushan, "Probing the aging effects on nanomechanical properties of a lifepo4 cathode in a large format prismatic cell," *Journal of Power Sources*, vol. 280, pp. 256–262, 2015.

[11] S. Ramdon, B. Bhushan, "Nanomechanical characterization and mechanical integrity of unaged and aged li-ion battery cathodes," *Journal of Power Sources*, vol. 246, pp. 219–224, 2014.

[12] D. Aurbach, E. Zinigrad, Y. Cohen, H. Teller, "A short review of failure mechanisms of lithium metal and lithiated graphite anodes in liquid electrolyte solutions," *Solid State Ionics*, vol. 148, pp. 405–416, 2002.

[13] S. C. Nagpure, B. Bhushan, S. S. Babu, "Multi-scale characterization studies of aged li-ion large format cells for improved performance: An overview," *Journal of Electrochemical Society*, vol. 160, A2111–A2154, 2013.

[14] *Guide to lithium plating in lithium-ion batteries*. [Online]. Available: https://www.accure.net/battery-knowledge/blog-guide-to-lithium-plating-in-lithium-ion-batteries#:~:text=emit%20%5C%20an%5C%20electron.-,Causes%5C%20of%5C%20lithium%5C%20plating,the%5C%20potential%5C%20of%5C%20metallic%5C%20lithium..

[15] J. W. Braithwaite et al, "Corrosion of lithium-ion battery current collectors," *Journal of Electrochemical Society*, vol. 146, pp. 448–456, 1999.

[16] M. Broussely et al, "Main aging mechanisms in li ion batteries," *Journal of Power Sources*, vol. 146, pp. 90–96, 2005.

[17] P. Arora, Z. Zhang, "Battery separators," *Chemical Reviews*, vol. 104, pp. 4419–4462, 2004.

[18] L. Norin, R. Kostecki, F. McLarnon, "Study of membrane degradation in high-power lithium-ion cells," *Electrochemical Solid State Letters*, vol. 5, pp. 67–70, 2002.

[19] J. Warner, *Handbook of Lithium-Ion Battery Pack Design*. Elsevier Inc., 2015.

[20] *Artificial intelligence — wikipedia, the free encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_intelligence.

[21] I. Goodfellow, Y. Bengio, A. Courville, *Deep learning*. MIT press, 2016.

[22] *Machine learning algorithms*. [Online]. Available: https://authoritysport.blogspot.com/2019/11/machine-learning-algorithms.html.

[23] A. K. Jain, J. Mao, M. K. M, "Artificial neural networks: A tutorial," *Computer*, vol. 29, pp. 31–44, 1996.

[24] W. S. McCulloch, W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of mathematical biology*, vol. 52, pp. 99–115, 1990.

[25] F. Rosenblatt, "Perceptron simulation experiments," *Bulletin of mathematical biology*, vol. 48, pp. 301–309, 1960.

[26] G. F. A, J. Schmidhuber, F. Cummins, "Learning to forget: Continual prediction with lstm," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99.*, IET, 2000, pp. 2451–2471.

[27] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[28] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, pp. 1550–1560, 1990.

[29] R. J. Hyndman, G. Athanasopoulos, *Forecasting: Principles and Practice*. OTexts, 2018.

[30] *How to interpret r-squared in regression analysis*. [Online]. Available: https://statisticsbyjim.com/regression/interpret-r-squared-regression/.

[31] G. Zhao, G. Zhang, Y. Liu, B. Zhang, C. Hu, "Lithium-ion battery remaining useful life prediction with deep belief network and relevance vector machine," *IEEE International Conference on Prognostics and Health Management (ICPHM)*, 2017.

[32] H. X. Dong, J. Y, Lou, W. C, "Lithium-ion battery state of health monitoring and remaining useful life prediction based on support vector regression-particle filter," *Journal of Power Sources*, vol. 271, pp. 114–123, 2014.

[33] D. Liu, J. Pang, J. Zhou, Y. Peng, P. M, "Prognostics for state of health estimation of lithium-ion batteries based on combination gaussian process functional regression," *Journal of Energy Storage*, vol. 53, pp. 832–839, 2013.

[34] D. Yang, Y. Wang, R. Pan, R. Chen, Z. Chen, "A neural network based state-of-health estimation of lithium-ion battery in electric vehicles," *Energy Procedia*, vol. 105, pp. 2059–2064, 2017.

[35] Y. Tan, G. Zhao, "Transfer learning with long short-term memory network for state-of-health prediction of lithium-ion batteries," *IEEE International Conference on Prognostics and Health Management (ICPHM)*, vol. 67, pp. 8723–8731, 2020.

[36] *Minmaxscaler*. [Online]. Available: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`.

# A
# Python Code Used with NASA Dataset

```
get_ipython().run_line_magic('matplotlib', 'inline')
get_ipython().system('pip show tensorflow')


# # Library import
# In this part, all the necessary libraries for the treatment of the dataset

# In[2]:


import numpy as np
import pandas as pd
import scipy.io as sio
from scipy.io import loadmat
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
from os.path import dirname, join as pjoin ##for loadmat
from sklearn.model_selection import TimeSeriesSplit


# # Loading the dataset into memory

# In[3]:


df_link = (r'C:/Users/kguner2/JN/naza/' + 'B0005.mat')
mat = sio.loadmat(df_link)


# In[4]:


df = pd.DataFrame(mat['B0005'][0][0][0][0])
```

```
df.head()


# In[5]:


indices = list(range(6))
plt.figure(figsize=(12,15))
cycle = 1

for index in indices:
    plt.subplot(6, 1, index+1)
    plt.plot(df['data'].iloc[cycle][0][0][index][0])
    print(df['data'].iloc[cycle][0][0].dtype)


# In[6]:


df_discharge = df[df.type == 'discharge']
df_disc_copy = df_discharge.copy()


# In[7]:


#seperating each data in the data column
d_types = ['Voltage_measured', 'Current_measured', 'Temperature_measured',
#creating empty columns to fill with data
for d_type in d_types:
    df_disc_copy[d_type] = 0

def seperate_data(df_row):
    indices = list(range(7))
    for index in indices:
        df_row[d_types[index]] = df_row.data[0][0][index][0]
    return df_row

df_disc_updated = df_disc_copy.apply(seperate_data, axis=1)


# In[8]:


df_disc_updated.head()


# In[ ]:
```

```
# In[9]:


#creating array of only constant current discharge

voltages_times =[]
def drop_non_cc(df_row):
    index_min_voltage = df_row.Voltage_measured.argmin()
    new_voltage = df_row.Voltage_measured[:index_min_voltage]
    new_time = df_row.Time[:index_min_voltage]
    return new_voltage, new_time

voltages_times = df_disc_updated.apply(drop_non_cc, axis=1)


# In[10]:


voltage_changes = []
durations = []
for voltage_time in voltages_times:
    voltage_change = voltage_time[0][0]-voltage_time[0][-1]
    duration = voltage_time[1][-1]-voltage_time[1][0]
    voltage_changes.append(voltage_change)
    durations.append(duration)


# In[11]:


plt.figure(figsize=(12,16))
indices = list(range(10))*10
for index in indices:
    plt.plot(df_disc_updated.iloc[index].Time, df_disc_updated.iloc[index].V


# In[12]:


soh = df_disc_updated.Capacity/df_disc_updated.Capacity.iloc[0]
plt.plot(soh)
```

```
# ## Predicting SOH

# In[13]:


from keras.preprocessing import sequence
from sklearn.linear_model import LinearRegression
from keras.layers import Input,Dense, Dropout, Embedding, LSTM, Bidirectiona
from sklearn.model_selection import train_test_split

import tensorflow as tf
from tensorflow.keras.models import Model
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_validate
from keras.models import Sequential
from keras.layers import Input,Dense


# In[14]:


#normalize the features
durations = np.array(durations)
voltage_changes = np.array(voltage_changes)

scaler = MinMaxScaler()
scaled_duration = scaler.fit_transform(durations.reshape(durations.shape[0]
scaled_voltages = scaler.fit_transform(voltage_changes.reshape(voltage_chang


# In[17]:


soh = np.array(soh)
scaled_soh = scaler.fit_transform(soh.reshape(soh.shape[0], 1))


# In[18]:


previous_value = 1
soh_differences = []
for value in scaled_soh:
    difference = value-previous_value
    previous_value = value
    soh_differences.append(difference)
```

```
plt.plot(soh_differences)
soh_differences = np.array(soh_differences)


# In[19]:


previous_value = 1
duration_differences = []
for value in scaled_duration:
    difference = value-previous_value
    previous_value = value
    duration_differences.append(difference)

plt.plot(duration_differences)
duration_differences = np.array(duration_differences)
#scaled_features = duration_differences
scaled_features = scaler.fit_transform(duration_differences)


# In[20]:


# create the time series cross-validation strategy
tscv = TimeSeriesSplit(n_splits=10)

models = [LinearRegression(), DecisionTreeRegressor(), RandomForestRegresso
scoring = ['r2', 'neg_mean_absolute_error', 'neg_mean_squared_error']
cv_results = []

for model in models:
    scores = cross_validate(model, scaled_features, soh, scoring=scoring, cv
    cv_results.append(scores)

# create a table of the results
table = pd.DataFrame()
for i, model in enumerate(models):
    model_scores = pd.DataFrame(cv_results[i])
    model_scores['model'] = str(model)
    table = pd.concat([table, model_scores])
table = table.reset_index()
table = table.drop(['index', 'fit_time', 'score_time'], axis=1)
table = table.groupby('model').mean()
print(table)


# In[21]:
```

```
models = [LinearRegression(), DecisionTreeRegressor(), RandomForestRegressor
tscv = TimeSeriesSplit(n_splits=10)

for model in models:
    plt.figure(figsize=(8, 6))
    for i, (train_index, test_index) in enumerate(tscv.split(scaled_features
        X_train = scaled_features[train_index]
        X_test = scaled_features[test_index]
        y_train = soh_differences[train_index]
        y_test = soh_differences[test_index]

        # training
        model.fit(X_train, y_train)

        #(prediction starting from nth cycle)

        forecast = model.predict(scaled_features[train_index[-1]:])
        previous_value = scaled_soh[train_index[-1]]
        new_forecasts = []
        for difference in forecast:
            new_forecast = previous_value + difference
            previous_value = new_forecast
            new_forecasts.append(new_forecast)
        scores = cross_validate(model, scaled_features, soh_differences, sc
        cv_results.append(scores)

        plt.plot(scaled_soh, color='b')
        plt.plot(list(range(test_index[0], len(soh)+1)), new_forecasts,
                 label=('{}th'.format(test_index[0])))
        plt.title('Prediction starting from nth cycle')
        plt.xlabel('cycle no')
        plt.ylabel('SOH')
        plt.legend()
        plt.ylim(top=1.1, bottom=-0.1)

# create a table of the results
table = pd.DataFrame()
for i, model in enumerate(models):
    model_scores = pd.DataFrame(cv_results[i])
    model_scores['model'] = str(model)
    table = pd.concat([table, model_scores])
table = table.reset_index()
table = table.drop(['index', 'fit_time', 'score_time'], axis=1)
table = table.groupby('model').mean()
print(table)
```

```
# In[22]:


#Import support vector regressor
from sklearn.svm import SVR

# Define the FNN architecture
model = SVR()

# Compile the model with optimizer, loss function, and metrics
#model.compile(loss='mean_squared_error', optimizer='adam')

plt.figure(figsize=(8, 6))
for i, (train_index, test_index) in enumerate(tscv.split(scaled_features)):
    X_train = scaled_features[train_index]
    X_test = scaled_features[test_index]
    y_train = soh_differences[train_index]
    y_test = soh_differences[test_index]

    # training
    model.fit(X_train, y_train)

    #model.evaluate(X_test, y_test)

    #(prediction starting from nth cycle)

    forecast = model.predict(scaled_features[train_index[-1]:])
    previous_value = scaled_soh[train_index[-1]]
    new_forecasts = []
    for difference in forecast:
        new_forecast = previous_value + difference
        previous_value = new_forecast
        new_forecasts.append(new_forecast)
#    scores = cross_validate(model, scaled_features, scaled_soh, scoring=sc
#    cv_results.append(scores)

    plt.plot(scaled_soh, color='b')
    plt.plot(list(range(test_index[0], len(soh)+1)), np.array(new_forecasts)
            label=('{}th'.format(test_index[0])))
    plt.title('Prediction starting from nth cycle')
    plt.xlabel('cycle no')
    plt.ylabel('SOH')
    plt.legend()
    plt.ylim(top=1.1, bottom=-0.1)


# In[23]:
```

```
plt.plot(soh_differences[train_index[-1]:])
plt.plot(new_forecast)


# In[24]:


previous_value = scaled_soh[train_index[-1]]
new_forecasts = []
for difference in forecast:
    new_forecast = previous_value + difference
    previous_value = new_forecast
    new_forecasts.append(new_forecast)

plt.plot(scaled_soh, color='b')
plt.plot(list(range(test_index[0], len(soh)+1)), new_forecasts,
                label=('{}th'.format(test_index[0])))


# In[25]:


# Define the FNN architecture
model = Sequential()
model.add(Dense(units=64, activation='relu', input_shape=(1,1)))
model.add(Dense(1))

# Compile the model with optimizer, loss function, and metrics
model.compile(loss='mean_squared_error', optimizer='adam')

plt.figure(figsize=(8, 6))
for i, (train_index, test_index) in enumerate(tscv.split(scaled_features)):
    X_train = scaled_features[train_index]
    X_test = scaled_features[test_index]
    y_train = soh_differences[train_index]
    y_test = soh_differences[test_index]

    # training
    model.fit(X_train, y_train)

    model.evaluate(X_test, y_test)

    #(prediction starting from nth cycle)

    forecast = model.predict(scaled_features[train_index[-1]:])
    previous_value = scaled_soh[train_index[-1]]
```

```
    new_forecasts = []
    for difference in forecast:
        new_forecast = previous_value + difference
        previous_value = new_forecast
        new_forecasts.append(new_forecast)
#    scores = cross_validate(model, scaled_features, scaled_soh, scoring=sc
#    cv_results.append(scores)

    plt.plot(scaled_soh, color='b')
    plt.plot(list(range(test_index[0], len(soh)+1)), np.array(new_forecasts)
            label=('{}th'.format(test_index[0])))
    plt.title('Prediction starting from nth cycle')
    plt.xlabel('cycle no')
    plt.ylabel('SOH')
    plt.legend()
    plt.ylim(top=1.1, bottom=-0.1)


# In[32]:


y_pred = model.predict(y_test)
r2= metrics.r2_score(y_test.reshape(y_pred.shape[0], 1), y_pred.reshape(y_pr
mae = metrics.mean_absolute_error(y_test.reshape(y_pred.shape[0], 1), y_pred
mse = metrics.mean_squared_error(y_test.reshape(y_pred.shape[0], 1), y_pred
scores = [r2, mae, mse]
print(scores)


# In[33]:


# Define the RNN architecture
model = Sequential()
#model.add(Embedding(vocab_size, embedding_dim, input_length=1))
#model.add(Dense(units=64, activation='relu', input_shape=(1,1)))
model.add(SimpleRNN(units=64, activation='relu', input_shape=(1,1))),
model.add(Dense(1))

# Compile the model with optimizer, loss function, and metrics
model.compile(loss='mean_squared_error', optimizer='adam')

plt.figure(figsize=(8, 6))
for i, (train_index, test_index) in enumerate(tscv.split(scaled_features)):
    X_train = scaled_features[train_index]
    X_test = scaled_features[test_index]
    y_train = soh_differences[train_index]
    y_test = soh_differences[test_index]
```

```
    # training
    model.fit(X_train, y_train)

    model.evaluate(X_test, y_test)

    #(prediction starting from nth cycle)

    forecast = model.predict(scaled_features[train_index[-1]:])
    previous_value = scaled_soh[train_index[-1]]
    new_forecasts = []
    for difference in forecast:
        new_forecast = previous_value + difference
        previous_value = new_forecast
        new_forecasts.append(new_forecast)
#    scores = cross_validate(model, scaled_features, scaled_soh, scoring=sc
#    cv_results.append(scores)

    plt.plot(scaled_soh, color='b')
    plt.plot(list(range(test_index[0], len(soh)+1)), np.array(new_forecasts)
            label=('{}th'.format(test_index[0])))
    plt.title('Prediction starting from nth cycle')
    plt.xlabel('cycle no')
    plt.ylabel('SOH')
    plt.legend()
    plt.ylim(top=1.1, bottom=-0.1)


# In[34]:


y_pred = model.predict(y_test)
r2= metrics.r2_score(y_test, y_pred)
mae = metrics.mean_absolute_error(y_test, y_pred)
mse = metrics.mean_squared_error(y_test, y_pred)
scores = [r2, mae, mse]
print(scores)


# In[40]:


#LSTM
model = Sequential()
model.add(LSTM(50, activation='relu', return_sequences=True, input_shape=(1
model.add(LSTM(25, return_sequences=True))
model.add(LSTM(25))
model.add(Dense(1))
```

```
# Compile the model with optimizer, loss function, and metrics
model.compile(loss='mean_squared_error', optimizer='adam')

plt.figure(figsize=(8, 6))
for i, (train_index, test_index) in enumerate(tscv.split(scaled_features)):
    X_train = scaled_features[train_index]
    X_test = scaled_features[test_index]
    y_train = soh_differences[train_index]
    y_test = soh_differences[test_index]

    # training
    model.fit(X_train, y_train, epochs=10)

    model.evaluate(X_test, y_test)

    #(prediction starting from nth cycle)

    forecast = model.predict(scaled_features[train_index[-1]:])
    previous_value = scaled_soh[train_index[-1]]
    new_forecasts = []
    for difference in forecast:
        new_forecast = previous_value + difference
        previous_value = new_forecast
        new_forecasts.append(new_forecast)
#    scores = cross_validate(model, scaled_features, scaled_soh, scoring=sc
#    cv_results.append(scores)

    plt.plot(scaled_soh, color='b')
    plt.plot(list(range(test_index[0], len(soh)+1)), np.array(new_forecasts)
            label=('{}th'.format(test_index[0])))
    plt.title('Prediction starting from nth cycle')
    plt.xlabel('cycle no')
    plt.ylabel('SOH(normalized)')
    plt.legend()
    plt.ylim(top=1.1, bottom=-0.1)


# In[41]:


y_pred = model.predict(y_test)
r2= metrics.r2_score(y_test, y_pred)
mae = metrics.mean_absolute_error(y_test, y_pred)
mse = metrics.mean_squared_error(y_test, y_pred)
scores = [r2, mae, mse]
print(scores)
```

```
y_test

plt.plot(scaled_soh)
plt.ylabel('Normalized SOH')
plt.xlabel('Discharge cycles')
plt.title('Nasa Data SOH vs. Cycle')



df_x = pd.DataFrame()
df_x['soh'] = pd.DataFrame(np.stack(soh, axis=0))
df_x['soh_differences'] = pd.DataFrame(soh_differences)
df_x['voltages'] = pd.DataFrame(voltage_changes)
df_x['durations'] = pd.DataFrame(durations)
df_x['duration_differences'] = pd.DataFrame(duration_differences)



df_x

def plotCorrMatrix(df, height=8, width=6):
    plt.subplots(figsize=(height, width))
    corr = df.corr()
    sns.heatmap(corr, annot=True, cmap='RdYlGn')
    print(corr)

plotCorrMatrix(df_x)
```

# PUBLICATIONS FROM THE THESIS

## Conference Papers

1. Guner, K. and Bayraktar, M. (2023, May 26–28). Machine Learning Prediction of Battery State of Health (SOH) [Paper presentation]. ICNASEN 2023: Nevsehir, Turkey