Real-Time Battery Monitoring System Using Machine Learning

Project report submitted
In partial fulfillment of the requirement for the degree of

Bachelor of Technology

by

Tathya Bhatt – 191030011041

Gurpreet Singh - 191030012005

Under Supervision Of

Dr. Jagat Jyoti Rath

Department of Mechanical and Aerospace Engineering

Institute of Infrastructure, Technology, Research and Management

(May 2023)

Institute of Infrastructure Technology, Research and Management (IITRAM)

CANDIDATE'S DECLARATION

We hereby certify that the work presented in this project report entitled "Real-time Battery Monitoring System using Machine Learning" in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology is a bonafide project work carried out by us during the period July 2022 to April 2023 under the supervision of Dr. Jagat Jyoti Rath.

This work has not been submitted elsewhere for the award of a degree/diploma/certificate.

Tathya Bhatt
191030011041

Gurpreet Singh
191030012005

This is to certify that the above-mentioned statement in the candidate's declaration is correct to the best of my/our knowledge.

Date:

Dr. Jagat Jyoti Rath
Supervisor

The viva-voce examination of **Tathya Bhatt** and **Gurpreet Singh** B.Tech project group, was held on "**08/05/2023**".

Date:

Dr. Saurabh Kumar Yadav
Project coordinator

Dr. Ajit Kumar
Examiner

Abstract

Every battery-operated device, including electric cars, smartphones, drones, etc., typically has a battery monitoring system (BMS) attached. This system's primary responsibility is to control battery levels and keep track of a battery's State of Charge and State of Health. This project focuses on a thorough analysis of 9 regression algorithms based on the accuracy of three different evaluation metrics and additional analysis of execution time of the algorithms to get faster predictions. For this analysis, we have chosen a complex and comprehensive dataset that includes a wide range of real-world VTOL battery conditions. From this analysis, Extreme Gradient Boosting was found to be the robust algorithm providing accurate results in less than few seconds, which is what machine learning is used for. Following this, the model was implemented on an in-house hardware system made with the aid of sensors and microcontrollers. Utilizing three evaluation metrics, including RMSE, MAE, and R-Squared Error, the analysis was conducted on two distinct datasets to better capture the nonlinearity in the data set. Additionally, the circuit was tested on a drone with a single motor, two motors, and then three motors, and data was collected to simulate the conditions in real life. The results of using these data to train and implement the online prediction were quite promising. The level of error for various charging states and the lack of training datasets was then assessed using the results. The errors were quite high at certain charging levels. However, there is always room for improvement.

Keywords: Machine Learning, Deep Learning, Neural Networks, Battery Monitoring System, State of Charge, Evaluation Metrics, Coulomb Counting

Acknowledgement

We want to sincerely thank everyone who helped finish this report by expressing my gratitude. First of all, we want to express my gratitude to Dr. Jagat Rath (Assistant Professor, Department of Mechanical and Aero-Space Engineering (IITRAM)) my supervisor, for their direction, encouragement, and feedback throughout the project. His knowledge and opinions were extremely helpful in determining the project's course.

Additionally, we want to thank Dr. Ajit Kumar for his assistance and insightful criticism. Their suggestions and insights helped the report's quality. We also want to be recognized for our efforts due to the dedication to the project and careful research that contributed to its completion.

Finally, we would like to express our gratitude to IITRAM, Ahmedabad, for giving us this wonderful opportunity and resources as well as to all of our friends who contributed significantly to the success of this project.

Tathya Bhatt
Gurpreet Singh
Department of Mechanical and Aerospace Engineering
IITRAM, Ahmedabad

List of Figures

List of Tables

Table of Contents

Nomenclature

$\bar{Y}$	mean of the dependent variable known as 'label'
$\overline{X_2}$	the mean values of independent variables known as 'features'
β_0	intercept
V	Voltage
n	Cycle Number
Wh	Energy Charge/Discharge
mAh	Discharge/Charge
℃	Temperature
mA	Current
s	Time

Chapter 1 Introduction

Nowadays, lithium-ion batteries are used in a variety of products, from electric vehicles to personal electronic devices like cell phones. Because these batteries are more vulnerable than lead-acid or NiCd batteries, they require more sophisticated monitoring to ensure safe operation.

The EV sector is experiencing a surge in demand for an effective battery management system, but this requires initial battery monitoring. In India, we can see an exponential increase in the number of electric Rickshaws that use lead acid batteries rather than more efficient lithium-ion batteries. Because the cost of a lead acid battery is significantly lower than that of a lithium-ion battery, it is critical to develop a battery monitoring and management system for lead acid batteries.

These lead acid batteries perform well at room temperature but poorly at lower temperatures. The existing technology is not adaptive to changing air conditions, and in a country like India, temperature plays a vital part in determining battery performance because temperatures here can reach 51.0 °C.

1.1 Battery Management System

The main duty of a Battery Management System (BMS) is to optimize the use of the battery in a portable device and minimize the likelihood of battery harm. This is achieved by monitoring and controlling the battery's charging and discharging processes. Overall, the BMS consists of the following functionalities

- Sensing Functionalities
    - Current
    - Voltage
    - Temperature
- Protection Functionalities
    - Short circuits
    - Over voltage, current and temperature
    - Disconnecting a faulty cell

- Estimation
    - State of charge
    - State of health

In addition to data communication between BMS master module and BMS acquisition modules as well as battery pack and surrounding applications

1.2 Temperature Acquisition

Determining the temperature which is achievable is one of the toughest jobs when making a BMS module. Not only must the pros and disadvantages of the different available sensor types (completely digital interface or analogue) be considered, but also where to measure the temperature of the pack. As a result, the needed number of cell temperature sensors is determined. Simulations may be required to determine the best placement for a limited number of sensors. When designing a lightweight battery pack, it is crucial to minimize the use of heavy copper busbars to handle high power operation. This requires the design engineer to consider potential thermal peaks and implement thermal monitoring of the busbars to address any issues.

A temperature requirement must, in general, address three use cases: charging, discharging, and storage. Batteries based on lithium cannot function adequately in extreme temperatures. Even within these constraints, though, accurate temperature readings are critical. At too high current rates, the significant effect of lithium plating can occur in the usual temperature range. Temperature, voltage, and current must be accurately measured to avoid lithium plating. Battery cells have a high thermal capacitance and strong thermal conductivity (in particular geometric routes), which are influenced and limited by thermal isolation boundary layers (housing, geometry of cells, etc.). If temperature sensors are not set properly, misreading's and thermal blind spots can occur.

1.3 Voltage Acquisition

A traditional lithium-ion battery management system must include at one voltage acquisition channel per battery connected in serial. Specific charging profiles are required for various battery chemistry to improve performance and prevent safety hazards during charge. Almost all Li-ion battery chargers use a charging method based on constant current and constant voltage. When the charger transitions to constant voltage mode, it is crucial to avoid surpassing the maximum permitted charge level to prevent subjecting the batteries to overcharging conditions. Such conditions can lead to excessive internal temperature rise and premature failure of the batteries. Rechargeable lithium ion battery cells can safely be used at 2.75V per cell. However, if an unprotected lithium cell is discharged over the minimum voltage level, the cell may be damaged, resulting in decreased cycle life, unpredictable voltage characteristics, and enlargement of cells due to internal chemical reaction.

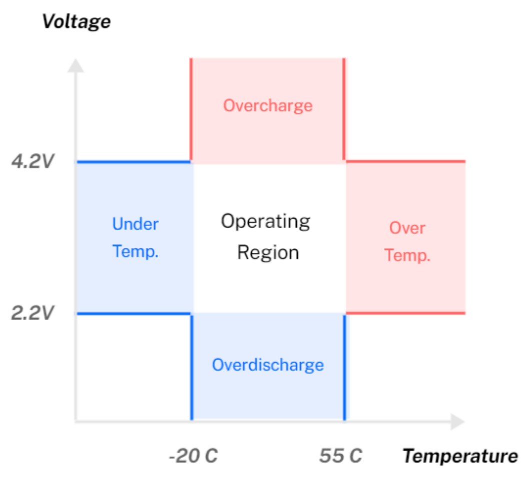


Figure 1.1 - Voltage Region

1.4 Current Acquisition

Current sensors, in addition to being imperfect systems, suffer from drift, offset, and temperature inaccuracy. Current sensors in automotive applications face conflicting demands as they are required to measure a wide range of currents, from milliamperes to 1000 amperes. Depending on the specific application, the sensor may also need to have a high bandwidth to capture rapid current changes. Additionally, the precision and immunity to electromagnetic interference (EMI) of the current sensor currently in use should be assessed.

1.5 Battery Monitoring Indexes

In addition to being non-ideal systems, current sensors are also subject to a certain amount of drift, offset, and temperature inaccuracy. In addition to this, the present sensors that are employed often have to fulfil contradictory requirements at the same time:

State-of-Charge (SoC) - The State-of-Charge (SoC) of a rechargeable battery indicates how much of its maximum possible charge capacity is currently available, expressed as a percentage. To determine SoC, battery measurements are taken and various modeling techniques are employed. For instance, a simple method involves measuring the battery voltage (V) and storing the corresponding V-SoC relationship in a lookup table function within a microcontroller.

State-of-Health (SoH) – SoH is a gauge that shows the overall state and performance capability of a battery in relation to a new battery. It reflects the point in the battery's life cycle and its condition compared to a new battery. SoH calculation can involve various techniques, including cycle-counting. For instance, by counting the number of full charge/discharge cycles (Cn) and utilizing a stored maximum capacity-Cn function, the SoH can be determined.

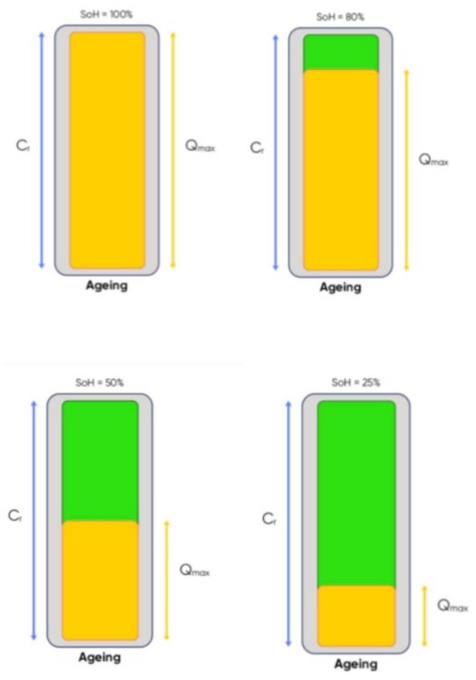


Figure 1.2 - SOH Visualization

The Remaining Run-time - The remaining run-time of a battery refers to the amount of time it can supply power to a portable device before it runs out of charge, under appropriate discharge conditions. There are two ways to estimate the remaining run-time, depending on the type of load. If the device has a current-type load, the remaining capacity (measured in mAh) is divided by the current being drawn (measured in mA). On the other hand, if the device has a power-type load, the remaining capacity (measured in mWh) is divided by the power being drawn (measured in mW).

1.6 Working of a BMS System

The battery management system

1. Keeps track of individual cells in the battery pack.

2. Calculates how much current can safely go in and come out without damaging the battery. The current limits prevent the source and the load from overdrawing or overcharging the battery.

3. Protects the battery pack from cell voltages getting too high or low, which helps increase the battery's longevity.

4. The BMS also measures the remaining charge in the battery. It continually observes the energy entering and exiting the battery module and monitors voltage of the cell. This data is then used to know when the battery is drained and when to shut the battery down.

Following data capture, the data is routed via the necessary components in order to complete the appropriate activities via the Control Area Network Network.

1.7 Applications of a BMS

Battery monitoring systems are used in a variety of applications where the performance and health of a battery are critical. Some common examples include in electric vehicles, renewable energy systems (such as solar and wind power systems), and backup power systems (such as in hospitals

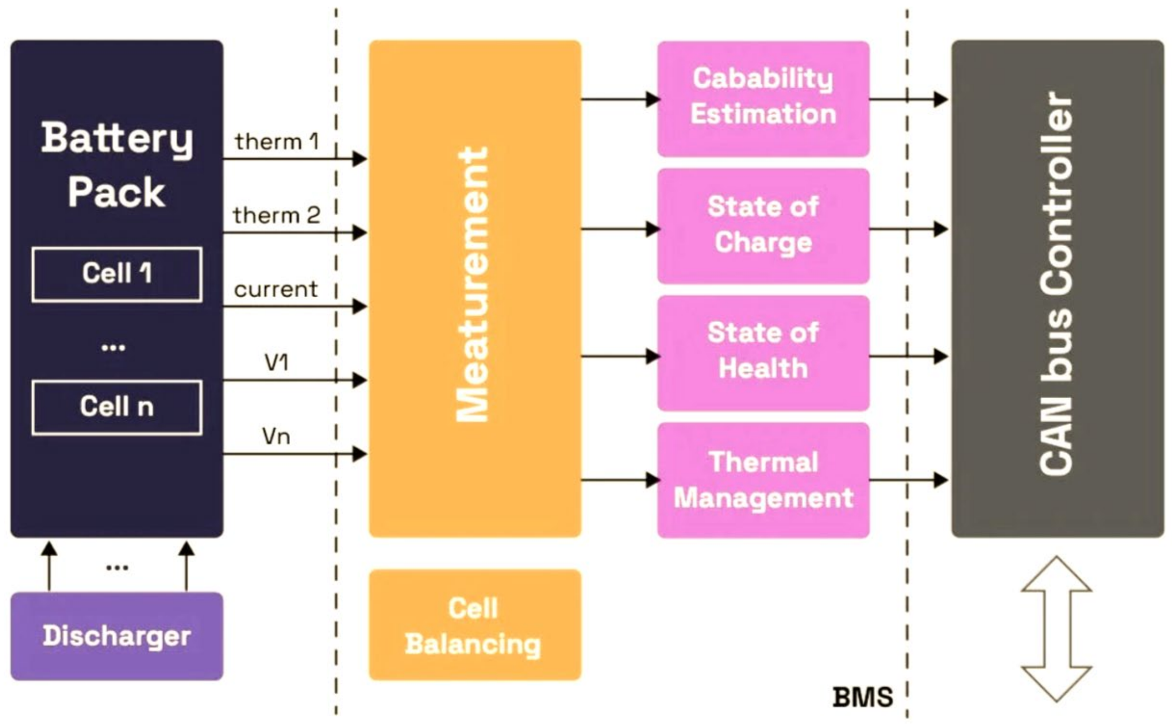


Figure 1.3 - Architecture of a typical BMS

and data centers). Battery monitoring systems can also be used in consumer electronics, such as smartphones and laptops, to provide users with information about the battery life and health of their devices.

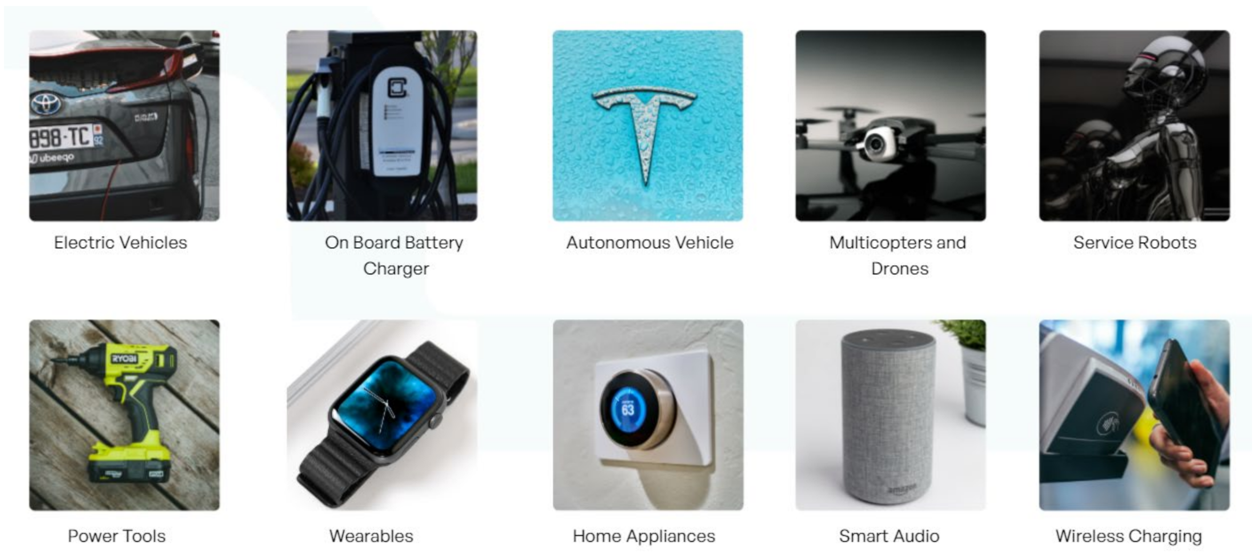


Figure 1.4- Application Areas

Some more application areas -

1. Determining the state of charge in electric vehicles and autonomous vehicles
2. Maintaining the temperature of a charger
3. Battery monitoring of a multicopters and drones
4. BMS is used in service robots
5. Small sized power tools using DC motors require BMS for the users safety
6. Wearables also use BMS
7. Home appliances operated with battery need a good BMS

Chapter 2 Methodology

Our primary objective is to identify a relevant problem statement in this field and examine the existing research on Battery Management System (BMS) modules based on data-driven approaches. This involves conducting a thorough literature review, defining the problem statement, and outlining the project objectives. Finally, we will establish the necessary steps to complete the project.

2.1 Literature Review

Sr. No	Paper Title	
1.	Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *J Big Data* 53 (2021)	Alzubaidi, L. Zhang, J. Humaidi, A.J.
	Summary It focuses on convolutional neural network (CNN) architectures, difficulties, uses, and future directions to give a thorough overview of deep learning. The study starts off with an overview of artificial neural networks, including feedforward networks, recurrent neural networks, and deep feedforward networks, as well as deep learning. The discussion of CNN architectures, which are frequently employed in computer vision tasks like object recognition, segmentation, and detection, continues after that.	
2.	Stock price prediction using support vector regression on daily and up to the minute prices. *The Journal of Finance and Data Science*, *4*(3), 183–201.(2018).	Henrique, B. M. Sobreiro, V. A. Kimura, H.
	Summary It investigates the application of support vector regression (SVR) for stock price prediction using both daily and real-time price data. By combining various input	

	feature and time horizon combinations, the study compares the performance of SVR models. According to the findings, SVR models that use both daily and up-to-the-minute price data outperform those that use only daily or up-to-the-minute data. The authors contend that combining the two types of data yields a more complete picture of the state of the market and aids in identifying brief fluctuations in stock prices. Additionally, the study contrasts the effectiveness of SVR models over a range of time horizons, from one day to one week. The authors discover that models with shorter time horizons typically outperform other models, suggesting that SVR is more appropriate for short-term forecasts than long-term ones.	
3.	A tutorial on support vector regression. Statistics and Computing, 14(3), 199–222.(2004).	Smola, A. J. Schölkopf, B.
	Summary Support vector regression (SVR), a machine learning algorithm for regression analysis, is described in the paper "A tutorial on support vector regression". The tutorial goes over the fundamental ideas of SVR, such as how to formulate an optimization problem, pick kernel functions, and fine-tune hyperparameters. Support vector machines (SVM), which are frequently employed for classification issues, have a variant called SVR. By identifying a function that roughly approximates the relationship between a set of input features and a continuous output variable, SVR is made to handle regression issues. The tutorial explains how to formulate the SVR optimization problem in its simplest form, which involves identifying a function that maximises the difference between expected and actual results. The tutorial provides an overview of frequently used kernel functions, including linear, polynomial, and radial basis function kernels. The choice of kernel function is crucial in SVR.	
4.	A Guide to Lithium Polymer Batteries for Drones Article – Tyro Robotics	Lauren Nagel

	The discharge rate or the C rating is a measure of how quickly the battery can safely discharge. If a battery has a C rating of 25 and a capacity of 5800 mAh/5.8 Ah, you could safely discharge it at 25 times the capacity of the battery, 25 x 5.8 = 145 Ah Lithium Ion batteries are composed of anode and cathode terminals and the electrolyte is filled between them which performs as a charge career. A 4S battery would have four LiPo cells in series (S). giving a 14.8 V battery (4 x 3.7 V = 14.8 V). A battery might also have a code like 4S2P, which tells us that there are four cells connected in series and two cell sets connected in parallel (P), for a total of eight LiPo cells.	
5.	Battery Management System : Hardware Concepts – An Overview Applied Sciences MDPI (Page 2-14)	Markus Lelie Thomas Braun Marcus Knips Hannes Nordmann Florian Ringbeck Hendrick Zappen Dirk Uwe Sauer
	After a short analysis of general requirements, several possible topologies for battery packs and their consequences for the BMS' complexity are examined. This paper focuses on the hardware aspects of battery management systems (BMS) for electric vehicle and stationary applications. Four battery packs that were taken from commercially available electric vehicles are shown as examples. Later, implementation aspects regarding measurement of needed physical variables (voltage, current, temperature, etc.) are discussed, as well as balancing issues and strategies. Finally safety considerations and reliability aspects are investigated	
6.	Design a Battery Monitoring System for Lead-Acid Battery International Journal of Creative Research Thoughts (IJCRT) (Page 302-310)	Niraj Agarwal Phulchand Saraswati Ashish Malik

		Yogesh Bateshwar
	An efficient energy-management system for Lead Acid Battery, using Matlab and Arduino, was developed and tested. The system uses an ACS712 sensor to detect current and voltage in the circuit while LM35 Thermistor is used to detect the temperature. The data output from these sensors is stored and manipulated through Arduino (microcontroller). The State of charge (SOC) of the battery is the index which shows the amount of charge present in the battery. The SOC depends upon various parameters, such as current, voltage, temperature and pressure. The amount of charge in the battery varies significantly with the temperature. To develop a better BMS we need to have a good understanding of the relation of the temperature with the SOC. Also, the precession has to improve with time and hence we should try to incorporate more hybrid methods with which problems like early discharge could be avoided	
7.	Battery Management Systems: Accurate State of Charge Indication for Battery Powered Applications ISBN 978-1-4020-6944-4	P. P. L. Regtien H. J. Bergweld Dmitry Danilov Valer Pop

	A performance analysis of the SoC system was carried out to test the accuracy of the SoC algorithm. The next step was to identify the sources of error in the SoC evaluation system to arrive at more accurate SoC and remaining run-time calculation. Adaptive models for Qmax and overpotential measurement are presented in this book. The results show that the SoC evaluation system is indeed capable of adapting to different battery chemistries and providing accurate, universal SoC indication. For a proper evaluation, the system developed during this study was compared with a competitive system, Texas Instruments' book-keeping system bq26500 SoC IC. The main focus in this book has been on designing a universal SoC system that accurately calculates the SoC in percentage and a tr in minutes for an Li-based battery-powered device. The EMF and overpotential curves were obtained in battery measurements and were implemented in an SoC evaluation system using approximation by means of mathematical functions. Accurate modelling was likewise indispensable in arriving at a new voltage-prediction model that speeds up SoC indication on the basis of the EMF during the relaxation process.	
8.	Machine Learning Approaches in Battery Management Systems: State of the Art: Remaining useful life and fault detection IEEE Explore (Page 63-64)	Ardeshiri, R. R., Balagopal, B., Al-Sabah, A., Ma, C., Chow, M.-Y
	In this paper, machine learning approaches for estimation, monitoring, and control of the battery management systems including the state of charge, state of health, and remaining useful life are reviewed with the focus on their weaknesses and strengths. Secondly, since achieving an accurate model of the battery is almost impossible with physical phenomena such as hysteresis, identification of the battery internal	

	parameters, state of charge and state of health estimation, and prediction of battery status by means of model-based methods a not sufficient. By comparing the existing methods, the advantages and disadvantages of these methods in RUL prediction and fault diagnosis and prognosis are discussed. Further studies are needed to reduce the computational complexity of machine learning approaches by means of different optimization techniques	
9.	XG Boost: A Scalable Tree Boosting System (2016) Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794	Chen, T. Guestrin, C.
	It introduces XG Boost, a new scalable tree boosting system that aims to perform more effectively than existing boosting systems. By combining weak learners, boosting is an ensemble learning technique that develops strong learners. With features like parallel processing, out-of-core computation, and sparsity awareness, the XG Boost system is built to handle large datasets. Additionally, it introduces a brand-new regularization method called "tree pruning" that, by removing pointless nodes from the trees, aids in preventing overfitting. The XG Boost algorithm, along with its objective function, gradient boosting, and tree learning, is thoroughly explained in the paper. Additionally, it covers a number of implementation specifics, including how to deal with missing values and how to incorporate categorical features.	
10.	Prediction of soil water infiltration using multiple linear regression and random forest in a dry flood plain, eastern Iran. *CATENA*, *194*, 104715.(2020)	Pahlavan-Rad M. R. Dahmardeh K. Hadizadeh M., Keykha G. Mohammadnia N. Gangali M.Keikha M. Davatgar N.Brungard

	Accurate predictions of infiltration rates can aid in the planning and design of irrigation systems. Soil water infiltration is a crucial parameter for water resource management and agriculture. The findings demonstrate that both MLR and RF models can reliably forecast soil water infiltration rates, with the RF model performing marginally better. The authors contend that similar dry flood plain environments can benefit from the RF model's accuracy in predicting soil water infiltration.	
11.	Predicting the current and future state of the battery using data driven machine learning Natural Machine Intelligence	Man-Fai Ng Jin Zhao Qingyu Yan Gareth Conduit Zhi Wei Seh
	Currently, most of the studies use experimental data only for machine learning, which is usually defined as 'black box' prediction. We see the opportunity of juxtaposing physics based modelling data and high-throughput experimental data with machine learning to develop a novel type of battery model. Physics-based modelling data can be obtained from multi-scale simulations from atomistic all the way to continuum level. Machine learning can combine models with different time and length scales. Moreover, high-throughput experimentation, perhaps guided by preliminary machine learning results, is the key to provide real-life and high-quality datasets on battery performance for machine learning. Combining all these pieces together with domain knowledge, the resulting physics-based machine learning technique paves the way for explainable 'white box' prediction	
12.	Overview of machine learning approach for Lithium Ion Battery Remaining Useful Lifetime Prediction. Electronics	Si Siyu Jin Xin Siyu Xinroug Huang Shunli Wang Remus Tedorescu Daniel Ioan Stroe

	This paper reviews the ML-based RUL prediction methods for Lithium-ion batteries, which are proposed in the literature.. From the above comparison, from the perspective of computational complexity, SVM, GPR, and ELM have the characteristics of simple structure and small calculation amount, but they are more suitable for calculation problems with small sample sizes. From the perspective of prediction accuracy, DNN, RNN, and LSTM all have good performance and are suitable for nonlinear complex systems, such as Lithium-ion batteries. Among them, RNN has a relatively poor ability of uncertainty management, and LSTM has a long and complicated training process and requires expensive equipment to accelerate training. In summary, DNN has a strong independent learning ability and generalization ability, making DNN more suitable for RUL prediction	
13.	Lithuim Ion Battery Life Cycle Prediction using deep learning regression model	Huawei Yang Yuan Cao Huaiqi Xie Shuai Shao Jie Zhao Tianyi Gao Jiajun Zhang Binghua Zhang
	These sets of data are commonly available in most Battery Management Systems (BMS), which makes the proposed regression model widely applicable to various battery systems. The five-key-feature-bed battery model is evaluated and validated all through the paper and is proved to have high prediction accuracy of battery life cycles. The proposed regression model can be utilized for real-time life cycle prediction without the need of historical data	

Table 2.1 - Literature Review

2.2 Problem Statement

The majority of battery monitoring systems in use today rely on hardware circuits and techniques to measure current, voltage, and ultimately state of charge (SOC). The temperature of the surrounding environment, which is crucial for estimating the battery's runtime accurately, is not taken into account by these methods. Hysteresis, which is a condition in which there is a delay in a system's output and results in a minimal amount of measuring error, can affect current algorithms (such as Coulomb Counting and the Voltage Method). Accuracy suffers because the effects of ageing batteries are not taken seriously. Because of these losses, accuracy can be compromised, and to ensure a reliable output with high accuracy, a data-driven machine learning approach is necessary.

2.3 Objectives

In this modern age of artificial intelligence, there are very few BMS systems based on machine learning.

1. Additionally, it is important to take in mind the impact of the temperature of the battery. A huge quantity of data sets needs to be gathered at various temperature extremes, and based on those data sets, with the help of machine learning, a successful model needs to be developed.

2. Making a BMS system using sensors and microcontrollers to get the required amount of data to train the model.

3. Another objective is the Real time implementation of the desired algorithm on this BMS System on a drone.

2.4 Procedure

1. *To develop a battery monitoring hardware system*

   A microcontroller such as an Arduino or a Raspberry Pi to collect and process data from the battery; a voltage measurement device, such as a voltage divider or a voltage sensor, to measure the battery's voltage; a current measurement device, such as a current shunt or a current sensor, to measure the battery's current; a display, such as an LCD screen or an OLED screen, to display the battery's voltage, current, and any other pertinent informatio n; a power supply.This will enable the microcontroller to measure the voltage of the batte ry. Connect the device for measuring current to the battery and microcontroller. This will enable the microcontroller to measure the current of the battery. Establish a connection be tween the display and the microcontroller. This will allow the microcontroller to display the battery's voltage, current, and other pertinent information. Programming the microcon troller to collect data from the voltage and current measurement devices and display it on the displayThe code for the microcontroller can be written using a programming language , such as C or Python.Verify the functionality of the battery monitoring hardware by performing a series of tests. This can be accomplished by measuring the voltage and curr ent of the battery and comparing the displayed values to the actual values.

2. *Testing the hardware system*

First of all, ensuring that the BMS hardware system is properly installed and connections are valid. Next, power on the system and verify that every component is operating properly. This can be accomplished by manually inspecting each component or by employing diagnostic software to perform a system test. Once confirmed that all of the system's components are functioning properly, you can begin performance testing. This can be accomplished by simulating different operating conditions, such as temperature and load scenarios, and observing the system's response. As you test the system, be aware of any errors or malfunctions that may occur and make a note of any problems you encounter. You can then use this data to troubleshoot and resolve any potential issues. After a system has been thoroughly tested, it is essential to perform routine maintenance and inspections to ensure that it continues to operate properly over time. This may involve routinely inspecting and replacing components, as well as performing periodic diagnostic tests to ensure the optimal operation of the system.

3. *Development of a Machine Learning Algorithm for BMS.*

When selecting a machine learning algorithm for a specific task, numerous considerations must be taken into account. Consider the type of data you have access to, the nature of the problem you are attempting to solve, and the objectives you wish to accomplish. The following steps will assist you in selecting an appropriate machine learning algorithm: Define the problem you are attempting to solve and the type of machine learning problem you must address (e.g. classification, regression, clustering, etc.). Comprehend the various types of algorithms and their respective strengths and weaknesses. Certain types of problems or data are better suited to particular algorithms than others. Think about the quantity and quality of your data. Some algorithms require a large amount of training data, while others can function with a smaller amount. Using metrics such as accuracy, precision, and recall, evaluate the performance of different algorithms on your data. Select the algorithm that performs the best on your dataset and aligns with your objectives. Noting that there is no one-size-fits-all solution for selecting a machine learning algorithm is crucial. The optimal algorithm for your problem will be determined by a number of variables and may require experimentation.

4. *Evaluation of machine learning algorithms*

Collect and prepare the training dataset: To begin, collect and prepare the training dataset. This requires cleansing the data, ensuring it is in a usable format, and separating it into training and validation sets. After selecting a model and preparing the training data, the model needs to be trained using the training data. This requires inputting the data into the model and modifying the model's parameters to minimize the error on the training data. Evaluate the model: After training the model, its performance on the validation data must be evaluated. This will provide insight into how well the model adapts to new data. After model evaluation, it may be necessary to fine-tune its parameters to improve its performance on the validation data. This may involve modifying the model's learning rate, regularization intensity, or other hyperparameters. Finally, test the model on a held-out test set to determine how good is the models prediction accuracy on unseen dataset. This will provide a final estimate of the model performance.

5. *Prediction of battery parameters*

After training and evaluating the model performance and finding the accurate algorithm for a BMS system, we will move ahead with the experimental BMS circuit. This circuit will be helpful in data collection and state of charge prediction. Currently, we are trying to run this algorithm on a multirotor drone module and the results were analysed using Mean absolute error. Execution time analysis was also conducted which made it easier to choose the quickest algorithm time. Next steps focus on the prediction accuracy and comprehensively analyzing the error and find the common patterns.

Chapter 3 Overview of machine learning

Machine learning is a branch of artificial intelligence that uses statistical models and algorithms to enhance a system's ability to perform a specific task by improving its performance with experience. Machine learning algorithms can make predictions, classifications, and decisions based on what they have learned from examples and data. Several tasks which include image and speech recognition, natural language processing, and predictive modelling, are amenable to machine learning. It is a rapidly growing field that is utilized in numerous industries, including healthcare, finance, transportation, and retail.

Models for machine learning include supervised learning and unsupervised learning. Supervised learning refers to the process of training a model on a dataset that has labeled examples, indicating the correct outputs for each example in the training data. This model uses the information so that the relation between input and output can be known, and can then make predictions based on unseen, unobserved data. Regression and classification assignments are examples of supervised learning.

Unsupervised learning entails model training with an unlabeled dataset in which the correct outputs are not provided. The model must self-learn to recognize patterns and structure in the data. Unsupervised learning includes tasks like clustering and dimensionality reduction.

3.1 Choice of model

We opt for supervised learning as the aim is to train a model to provide predictions or decisions based on labeled training data. In supervised learning, the training data includes both the input features and the corresponding correct outputs. The model then uses this information to learn the relationship between the input and the output, and can then make predictions on new, unseen data..

Regression models were implemented because the voltage, current and temperature are continuous variable, such as a price or a probability. They are frequently employed in applications where the aim is to comprehend the relation between various variables and to make predictions based on this relationship. A regression model could be used, for instance, to predict the sale price of a house based on its size, location, and other characteristics.

3.2 Basic Regression Working

The Ordinary Least Squares (OLS) regression is a well-known approach used to determine the coefficients of linear regression equations, which illustrate the association between a dependent variable (simple or multiple linear regression) and one or more independent quantitative variables. The method of least squares estimates the least squares error (SSE). In our case the model is multivariable regression which uses more than one variables as input to predict the output.

3.2.1 Line of Best Fit

The primary objective of the best fit line is to ensure that the predicted values are as close as possible to the actual or observed values. It is not useful to predict values that are significantly different from the actual values. Therefore, the aim is to reduce the difference between the predicted values and the observed values, which is also referred to as error.

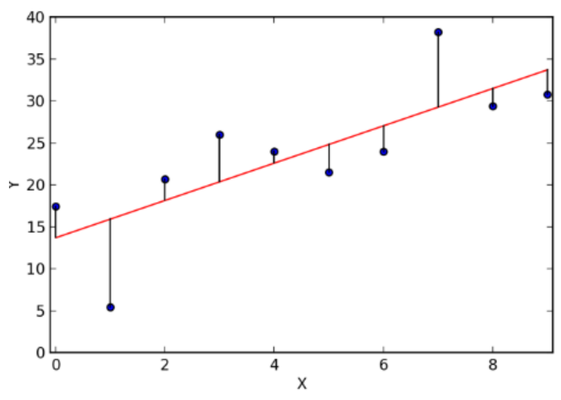


Figure 3.1 - Linear regression of a 2d dataset

3.2.2 Bias

The term "bias" refers to the disparity between the predicted and actual outcomes of our model. Bias results from the basic assumptions made by the model about the data, which enable it to make predictions for new data. If the model's assumptions are too basic, resulting in a high degree of

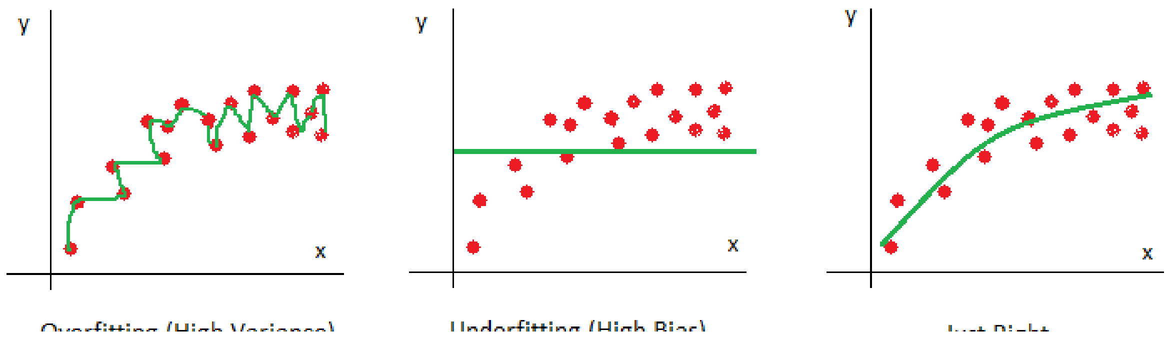


Figure 3.1- Overfitting and Underfitting

bias, it may be unable to capture critical data features. When the model is unable to identify patterns in the training data and consequently performs poorly on both seen and unseen data, it is referred to as underfitting. This occurs when the model does not capture the complexity of the data and is not able to generalize well to new data.

3.2.3 Variance

The model can see the data a set number of times during training in order to identify patterns. Insufficient time spent working with the data will result in bias because patterns won't be discovered. other hand, if the model is allowed to view the data too many times, it will learn very well for only that data. The majority of patterns in the data will be captured, but it will also learn from extraneous data or noise that is present.

Variance can be thought of as the model's sensitivity to changes in the data. From noise, our model might learn. This will lead our model to value unimportant features highly. The Bias-Variance Tradeoff refers to the need for finding a suitable balance between the model's bias and variance. By achieving this balance, we can effectively identify and incorporate the significant patterns in our model while disregarding any extraneous noise. This balance helps to minimize the error in our model, resulting in optimal performance.

3.2.4 Cost Function

After the training is completed, we should check how good the model is giving results. Hence, the function which corrects the model is necessary to get accuracy and make it the best trained model. A cost function is a tool used to evaluate the accuracy of a model in establishing a connection between input and output. It provides a measure of the extent to which the model's predictions are incorrect.

$$Cost\ Function = \frac{1}{n}\sum_{0}^{n}\left(Y_{pred} - Y_{actual}\right)^2 \tag{1}$$

3.2.5 Gradient Descent

Gradient Descent is a technique used to minimize the error or cost function of a model. Its purpose is to locate the lowest possible error value that can be achieved. Gradient Descent can be seen as the path that needs to be followed to reach the smallest error value. As the error in the model may vary at different points, an efficient approach is needed to reduce computational time.

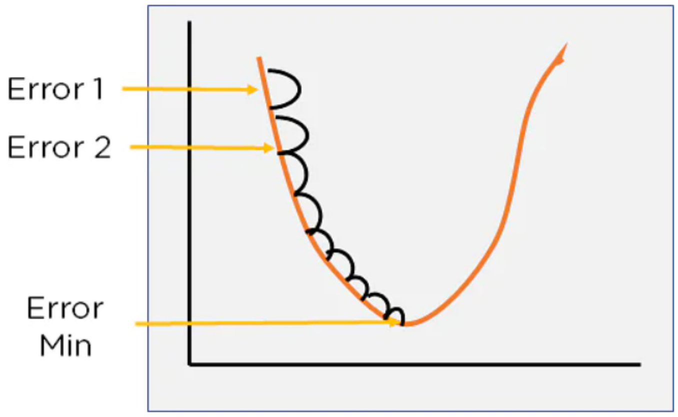


Figure 3.5 - Gradient Descent

The concept of Gradient Descent can be likened to a rolling ball on a hill, where the ball naturally moves towards the lowest point of the hill. In this case, the lowest point of the hill represents the point where the error or cost function of the model is minimized, as it is the point where the model's predictions are most accurate. In a model, the error can be minimum at a specific point, and it may increase beyond that point. In Gradient Descent, the error for each input variable is calculated, and the process is repeated to obtain progressively smaller error values. This approach enables the determination of variable values that result in the minimum error, leading to an optimized cost function.

3.3 Models used in this study

3.3.1 Ordinal Least Square Regression

An approach to statistical regression **[32]** known as ordinary least squares is used to forecast unknown values from a set of data. Ordinary Least Squares, or OLS, can be used, for instance, to predict shoe size from a data set that includes height and shoe size. Given the information, one can calculate a rate of change and predict shoe size based on a subject's height using the ordinary least squares formula. In

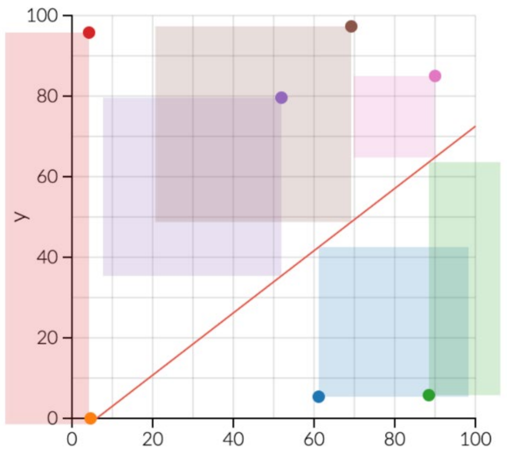


Figure 3.2 - Visualization of OLS

essence, OLS creates a dependent variable from an input, the independent variable.

Ordinary least squares is a method that involves adding and multiplying the input or independent variable with other variables, called β. The value of the first β, denoted as "β1", determines the slope of the function.. Particularly, it finds the output if the input were 0. The coefficient, or how much of a difference there is between increments in the independent variable, is represented by the second beta, designated as "β2" in the equation.

OLS technique determines the optimal slope for a dataset by using the errors, which are the vertical distances between the data points and a regression line, to estimate the β values. OLS finds the regression line by calculating the squared errors, as illustrated in the image. It aims to identify the line that passes through the sample data and minimizes the total sum of squared errors.

3.3.2 Decision Tree

A machine learning algorithm called decision tree regression employs **[31]** a decision tree to predict continuous numeric values. According to the magnitude of the independent variables, the data is partitioned into smaller subsets, and these subsets are recursively split until the final prediction is reached. Each leaf node in the tree indicates a forecasted numerical value, and each internal node of tree represents a decision based on the value of an independent variable.

The algorithm first chooses the variable that best splits the data based on a predetermined criterion, such as minimizing the sum of squared errors, before building a decision tree regression model. The algorithm divides the data into two subgroups based on the value of a selected variable, and then it iteratively repeats this process until the tree is complete. To make a prediction for a new data point, the model navigates from the root node to a leaf node by evaluating the independent variable values.

In disciplines like finance, engineering, and environmental science, decision tree regression is a powerful tool for forecasting continuous numeric values. But it might be prone to overfitting and struggle with complicated data distributions. To increase the precision and generalizability of the model, it is crucial to carefully tune the hyperparameters of the algorithm and use the right regularization techniques.

3.3.3 Random Forest

The machine learning technique called random forest **[30]** utilizes a collection of decision trees as an ensemble to improve the accuracy of predictions. It functions by constructing numerous decision trees, each trained on a randomized subset of the training data and features. The algorithm then combines the individual trees' predictions to arrive at a final prediction.

Comparison of random forest and single decision tree reveals several benefits. It can first handle many input variables and model intricate relationships between them. Second, because the randomness introduced by the random subsets reduces the correlation between the trees, it is less prone to overfitting than a single decision tree. The third benefit is that it can offer an estimation of the feature importance, which is helpful for comprehending the underlying data.

A versatile algorithm, random forest can be applied to classification and regression tasks. It is widely used to forecast outcomes and highlight key elements in a variety of disciplines, including finance, healthcare, and environmental science. However, because there are so many trees involved, it might need more computational power than a single decision tree.

3.3.4 Gradient Boosting

A predictive model **[29]** created using the machine learning technique of gradient boosting is an ensemble of weak prediction models, typically decision trees. With gradient boosting, weak models are iteratively added to the ensemble, each one aiming to correct the flaws in the previous models. A simple initial model, like a decision tree, is typically used in the process. The algorithm then examines the mistakes made by the first model, focusing on those mistakes while training a new model. The algorithm then repeats the process, training a new model to concentrate on the mistakes made by the ensemble of the first two models before the second model is added. This keeps going until either the stopping criterion is satisfied, or the desired level of accuracy is attained.

Both regression and classification tasks can be accomplished using the potent technique of gradient boosting. It is frequently used to forecast outcomes and highlight crucial elements in industries like finance, healthcare, and marketing. If not carefully tuned, it can be prone to overfitting and may use more computing power than other algorithms.

3.3.5 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) **[28]** is an advanced implementation of gradient boosting that uses a combination of regularization techniques and parallel processing to enhance the performance and scalability of the algorithm. Similar to conventional gradient boosting, XGBoost builds an ensemble of weak prediction models, typically decision trees, iteratively in order to boost the model's predictive performance. However, XGBoost makes use of a number of improvements to increase its strength and power. For example, XGBoost uses a technique called "gradient-based sampling" to select the most informative samples for each new model in the ensemble, which reduces the computational cost and improves the accuracy. To avoid overfitting and enhance the model's generalization capabilities, XGBoost also employs regularization methods like L1 and L2 regularization as well as dropout.

A wide variety of objective functions and evaluation metrics are supported by XGBoost, which can be easily customized. Additionally, it has a number of sophisticated features that can boost the algorithm's precision and effectiveness, including early stopping, cross-validation, and parallel processing. XGBoost has been increasingly used in various industries such as finance, healthcare, and e-commerce for predictive modeling, feature selection, and anomaly detection.

3.3.6 Neural Network

A neural network is a type of machine learning model **[27]** inspired by the structure and function of the human brain. It comprises layers of interconnected nodes or neurons that work together to process input data and produce predictions or classifications. A single neuron serves as the fundamental building block of a neural network. It accepts one or more input values, applies a mathematical function to them, and outputs a value. Weighted connections, which regulate the strength of the signal passed between neurons, are used to link neurons to one another.

The neurons in a neural network with multiple layers are arranged in layers, with each layer processing the results of the layer before it. The raw input data is sent to the first layer, which is referred to as the input layer. One or more hidden layers take the output of the input layer and apply a number of nonlinear transformations to the data. The output layer—the top layer—is where the final predictions or classifications based on the transformed data are generated.

During the training of a neural network, the weights of the connections between neurons are adjusted to minimize the error between the predicted output and the actual output. This is often accomplished using an optimization algorithm such as gradient descent. Predictive modelling, speech and image recognition, natural language processing—there are many applications for neural networks. They can learn patterns and relationships in the challenging data, making them particularly useful for tasks involving large amounts of complex data.

3.3.7 Deep Neural Networks

A deep neural network **[26]** is a type of neural network that consists of multiple layers of neurons positioned between the input and output layers. With each layer building on the features learned by the previous layer, these layers allow the network to learn progressively more complex representations of the data. Depending on how complex the problem being solved is, there can be a few, hundreds, or even thousands of layers in a DNN. A DNN has layers of interconnected neurons that process the input data, similar to a regular neural network in terms of its basic structure. DNNs can learn more complex patterns and relationships in the data because they have more layers and neurons per layer than regular neural networks.

The activation function employed controls the computations carried out by each neuron in a DNN. For the network to be able to learn complex relationships in the data, the activation function introduces nonlinearity into the model. The ReLU (Rectified Linear Unit) function, the sigmoid function, and the tanh (hyperbolic tangent) function are a few examples of common activation functions used in DNNs.

3.3.8 Support Vector Machine (SVM Linear Kernel)

This algorithm is a form of supervised machine learning that can be used for both classification and regression purposes. The objective **[24]** of SVM in regression tasks is to learn a function that forecasts a continuous target variable's value based on a set of input features. The SVM algorithm looks for the hyperplane that maximizes the hyperplane and the nearest data points distance while best separating the data points in the feature space. It also searches for the hyperplane that maximizes the margin between the hyperplane and the nearest data points when performing SVM regression with a linear kernel.

3.3.9 Support Vector Machine (Radial Basis Function)

The kernel based on RBF **[25]** is a common pick for SVMs because it is capable of recording nonlinear relationships between variables that are input and output. SVMs with RBF kernels are designed to find a function that can approximate the underlying relationship between input and output variables. In other words, given a set of labelled data, SVMs with RBF kernel seek the best function that can map the input data to the output data. The RBF kernel computes the similarity between input data points in a high-dimensional feature space, and SVMs use this similarity measure to learn a nonlinear function that can predict the output variable.

SVM performance with RBF kernel regression is affected by several factors, including the kernel parameters chosen, the amount and quality of the training data, and the complexity of the input and output variables' relationship. To achieve optimal performance, SVMs with RBF kernels can be computationally intensive to train and also require tuning of the kernel parameters.

3.4 Rationale behind these model selection

For this project, as we need to predict the continuous output variable i.e State of Charge, supervised machine learning regression models which implement a mathematical function or model that can predict the output variable accurately for new, unknown input data. This is accomplished by training the model on a labelled dataset of input-output pairs with continuous output values. As a part of our comprehensive analysis, we wanted to test every regression models' results in the analysis so that a proper more firm understanding can be built with this analysis. So we've analyzed these nine models which best suits our problem statement of supervised learning based on the labeled dataset.

Chapter 4 Dataset

4.1 LG 18650HG2 Li-ion Battery Data

This project requires the dataset consisting of temperature, current, battery capacity and voltage data because these variables are the sole support for machine learning models without which the models will fail to predict the output. So at first **LG 18650HG2 Li-ion Battery Data [23]** is used to train the model. Dr. Phillip Kollmeyer from McMaster University in Hamilton, Ontario, Canada conducted HPPC tests (Hybrid Pulse Power Characterization) to gather the dataset. The tests were conducted on a new 3Ah LG HG2 battery cell, which underwent a series of rigorous tests at different temperatures (10℃, 25℃, 40℃). The battery was charged to 4.2V at a rate of 1C after each test.

4.2 Preliminary Results of the LG Dataset

Our objective was to make a robust system which should reduce the overall cost of the BMS system. Currently BMS systems have complex circuits which has a huge impact on its costs. By incorporating data driven methods and using a microprocessor we are aiming to make it robust system which can predict the charging state accurately with different operating conditions.

Previously we had tried estimating the State of Charge with an LG Battery dataset at different temperatures but there were certain problems

- The results with these dataset were similar irrespective of the algorithm choices. With further rumination it was found that the dataset of each cycle was quite linear which won't be helpful for the models as it will be easier for them to train and predict with higher accuracy.

- Secondly, This dataset had a lot less data which cant be used in real life accurate predictions and furthermore data driven approaches provides the best results when a large amount of dataset is available and trained.

- Moreover, in ML, Features play an important role in improving models performance and thus including more features can impact the models accuracy which was not there in the previous dataset.
- The results of the experiments were on the same training ratio which was 75:25 and with only 3 regression-based algorithms. This was not enough and won't justify the overall algorithm selection and performance.

Thus we found another dataset which could provide a more robust and accurate predictions.

4.3 eVTOL Dataset

The eVTOL (Electric Vertical Take-Off and Landing) Battery Dataset is a freely accessible dataset containing sensor measurements of battery voltage and current during flights of a small electric vertical take-off and landing aircraft (eVTOL). Researchers at Carnegie Mellon University's Robotics Institute gathered the data, which was released in 2021. Battery used was – Sony Murata 18650 VTC – 6Cell. The data collection was authored by Alexander Bills, Venkatasubramaniam Vishwanathan, Shashank Sripad, Evan Frank, Devin Charles, William Leif Fredericks is used to train the model. Each file consisted these variables

- Temperature (°C)
- Time (s)
- Cell Voltage (V)
- Current (mA)
- QCharge (mAh)
- Energy Discharge (Wh)
- QDischarge (mAh)
- Energy Charge (Wh)
- Cycle Number (n)

The folder consists of around 22 csv files at different conditions of eVtol usage. Each file contains data from experimental testing performed on a single cell over the course of that cell's life.

4.4 Data Selection

Five different conditions' data was taken into the analysis namely

VAH01	Baseline
VAH02	Extended cruise (1000 sec)
VAH05	10% power reduction during discharge (flight)
VAH07	CV charge voltage reduced to 4.0V
VAH09	Thermal chamber temperature of 20 degrees C

Table 4.1 - Chosen Conditions

Every condition had numerous cycle iterations and from these we considered five cycle' analyses for our model training. Then each model was studied for five cycles to make the analysis robust in different conditions and three evaluation metrics were compared.

4.5 Feature Engineering

Features refer to the input variables that are fed into machine learning models, and each column in the dataset corresponds to a feature. In order to get the optimal model, we must utilize only the most important features. If there are too many features, the model may learn from noise and capture unimportant patterns. Feature Selection refers to the process of selecting the most relevant data parameters. Training a model requires massive amounts of data to help the machine learn more effectively. Typically, noise contributes to a significant portion of data, and certain columns of our dataset may not significantly contribute to the model performance. In addition, having a large amount of data can make the training process slow and result in a slower model. The model may also be inaccurate if it learns from irrelevant data. Feature selection is what distinguishes high-quality data from the rest. In addition to selecting the appropriate model for our data, we must also select the appropriate data for our model.

Feature selection is a technique used to reduce the number of input variables in a model by retaining only the relevant data and eliminating noise from the dataset.

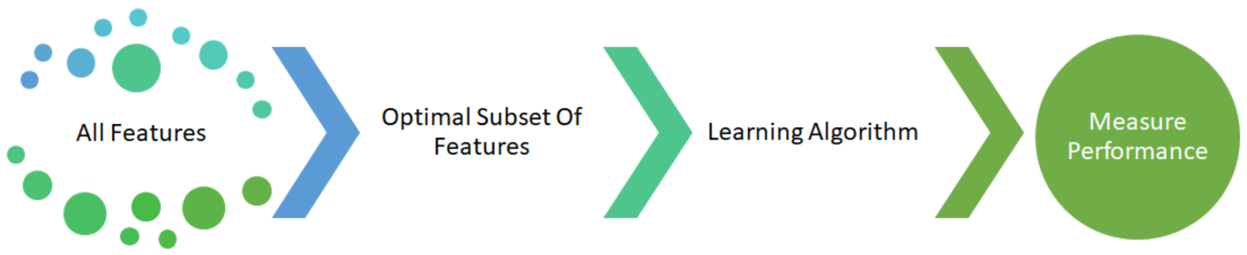


Figure 4.1 - Feature Selection Example (www.towardsdatascience.com)

Thus, in order to create a robust algorithm for our BMS System, we added several features which can help to train the model better. Machine learning (ML) relies heavily on statistical features to quantify and summarize the characteristics of the data. The development of ML models can be influenced by statistical features, which offer a way to analyze and comprehend data patterns.

We can identify significant patterns and characteristics of the electrical system under analysis by adding statistical features of mean, median, variance, and standard deviation to current, voltage, and temperature data using a sliding window technique with a window size of 10. Using the sliding window technique, data is divided into overlapping windows, and statistical features are computed for each window. This method can be helpful for tasks like fault detection, anomaly detection, or system optimization because it can help to capture local patterns in the data.

In comparison to larger window sizes, which might miss significant changes or trends in the data, we are able to gather information about the electrical system at a more granular level when using a window size of 10. Additionally, the use of statistical features like mean, median, variance, and standard deviation can give important information about the distribution, variability, and central tendency of the data for current, voltage, and temperature. We can increase the precision and performance of our predictions or classifications by incorporating these statistical features into our machine learning models.

Power, resistance, conductance, and temperature change have all been calculated from the input variables in addition to the statistical features of mean, median, variance, and standard deviation of current, voltage, and temperature data using a sliding window technique. These characteristics can offer more information about how the electrical system behaves and can be applied to the creation of machine learning models that are more precise and efficient.

Features:

- Mean Temperature
- Mean Current
- Mean Voltage
- Median Temperature
- Median Voltage
- Median Current
- Variance of Current
- Variance of Temperature
- Variance of Voltage
- Standard Deviation of Current
- Standard Deviation of Voltage
- Standard Deviation of Temperature
- Power
- Resistance
- Conductance
- Temperature Change

4.6 Normalization

To change the data into a standard format and increase the model's accuracy and performance, normalization is a crucial data preprocessing technique in machine learning. The input data is scaled to a standard range, such as between 0 and 1 or -1 and 1, as part of the normalization process. The Min-Max Scaler is a popular normalization technique that transforms the data to a particular range by taking away the minimum value and dividing it by the data range. The performance of the machine learning model can be impacted by problems like data skewness, outliers, and different units of measurement. This is why normalization is crucial. Different scales or ranges in the input data may cause the model to prioritize some features over others, which may result in skewed or incorrect predictions. Normalization makes sure that each feature is given the same weight and can help the model learn from the data. A popular normalization method is the Min-Max Scaler because it is straightforward, simple to use, and has minimal impact on the data's shape or distribution. The procedure involves dividing by the data range and subtracting the minimum value of each feature (i.e., the maximum value minus the minimum value).

$$x^{`} = \frac{x - x_{min}}{x_{\max} - x_{max}} \tag{2}$$

Chapter 5 Comprehensive Analysis

5.1 Data Splitting & Training Ratios

The greatest challenge for Machine Learning practitioners is determining how to split the dataset into training and testing subsets. Although it initially appears to be a simple problem, its complexity can only be determined by delving deeper into it. Inadequate training and testing sets can have ramifications on the model's output. It may result in overfitting or underfitting, causing our model to produce biased outcomes.

Training Set - The data which will be supplied into the model will be contained within the train set. In other words, our model would acquire information from this data. For example, a regression model will use these examples to identify gradients to decrease the cost function. Then, these gradients will be utilized to cut costs and accurately predict data.

Testing Set - The test set consists of the data used to evaluate the trained and validated model. It informs us about the effectiveness of our overall model and the probability that it may make unreasonable predictions. Numerous evaluation metrics (such as precision, recall, and accuracy, etc.) can be used to access the model performance.

5.1.1 Training Ratio

The term "training ratio" describes the ratio between the amount of data used to train a machine learning model and the amount used to test or validate the model. Usually expressed as a percentage, the training ratio has typical values between 60 and 80% for training and 20 to 40% for testing or validation.

The performance and accuracy of the model are directly impacted by the training ratio, which is crucial in machine learning. If the training ratio is too low, the model would not be capable of identifying the fundamental patterns and connections within the data. It may also be vulnerable to overfitting, in which case the model memorizes the training data but performs badly on new, untried data. Contradicting to it, if the training ratio is too high, the model might underfit, where it is too simple and fails to adequately represent the complexity of the data, and it might also have trouble generalizing to new data.

Thus, for this comprehensive analysis, three training ratio's 60:40, 70:30, and 80:20 were selected accordingly because only then it will help to understand how the model behaves with different amount of training dataset.

5.1.2 Feature Correlation

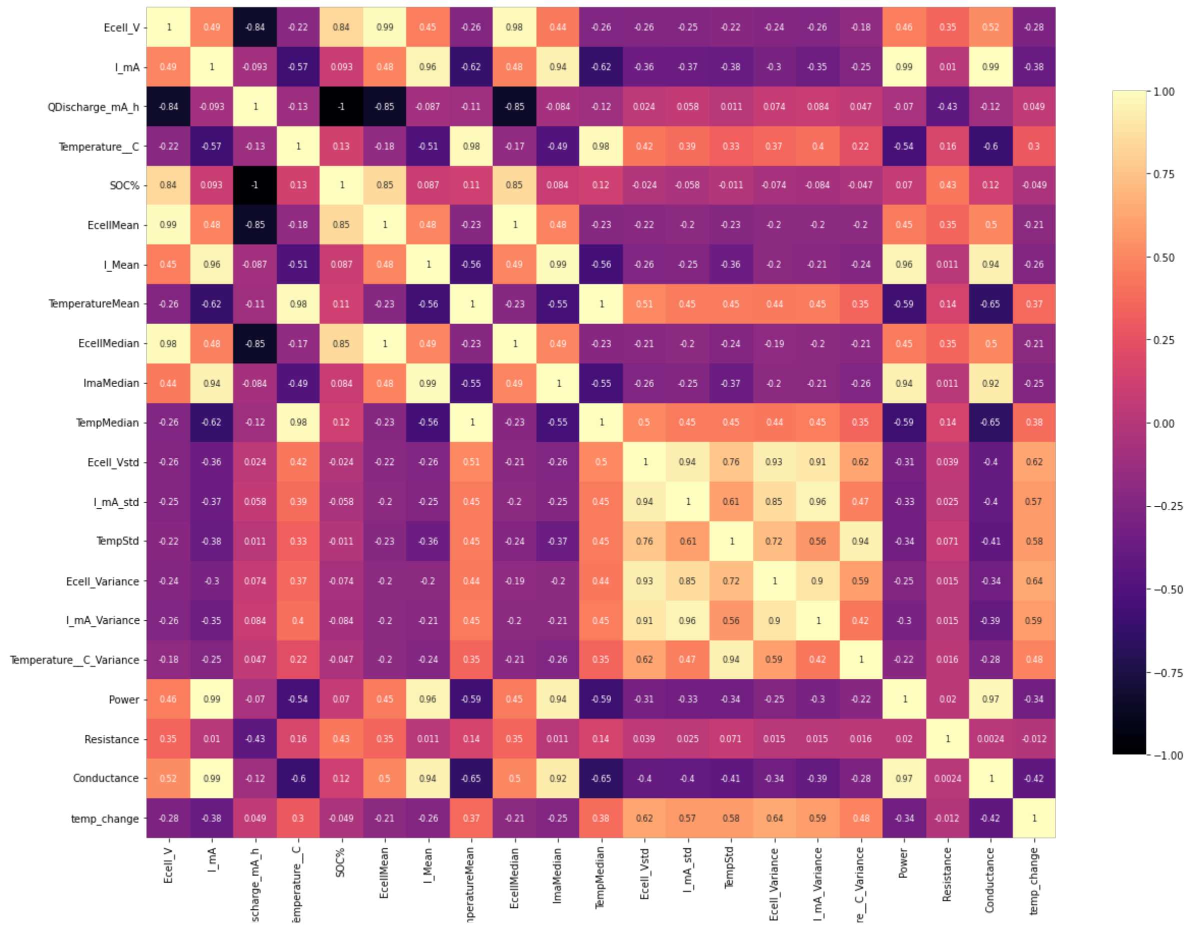


Figure 5.1 - Feature Correlation Heatmap

The correlation between two or more variables is found using heatmaps. High correlation between two or more features in a dataset can hinder a model's capacity to learn and make reliable predictions.

A model may be less able to generalize to other datasets with different correlations between the features if it was trained on a dataset with correlated features.

5.2 Evaluation Metrics

A machine learning model's effectiveness and accuracy are assessed using evaluation metrics. These metrics are crucial for evaluating the model's effectiveness and pinpointing potential areas for improvement. Depending on the task and model objectives, a variety of evaluation metrics can be used in machine learning.

5.2.1 Root Mean Squared Error (RMSE)

The most widely used metrics when dealing with regression issues is root mean squared error. RMSE is defined as the standard deviation of the prediction errors, which are also known as residuals. In essence, residuals are a measurement of how far data points are from the regression line.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} \tag{3}$$

Where:

Σ – Summation

$\hat{y_i}$ – Predicted Values

y_i – Original Values

N – Number of Observations

The concentration of data points around the regression line is described by RMSE. It is assumed that residuals in RMSE follow a normal distribution and are unbiased. The model performs better at prediction the lower the RMSE value. A model with an RMSE of 0 is the ideal predictor, and a model with a value of 1 is the worst. A low RMSE value indicates that the model's predictions are accurate and close to the actual values, while a high RMSE value indicates that the model's predictions are inaccurate and far from the actual values. RMSE is sensitive to outliers, meaning that RMSE gives more weight to larger errors compared to smaller ones. This can be useful in situations where large errors are particularly problematic. It can be used for most models.

5.2.2 R-Squared Error

R squared is a statistical indicator of the closeness of the data point fits the regression line. As the coefficient of determination, it has additional names. R-Square is calculated by dividing the variance that is explained by the linear model by the total variance .R-squared value always ranges from 0% to 100%, so 0% denotes that there is no variability in the response data around its mean and 100% denotes that the model fully accounts for all such variability. This demonstrates how much more accurate your model is if the R square value is higher.

R-squared = Explained variation / Total variation

It can be misleading: A high R-squared value is not always indicative of a good predictive model for the dependent variable.

The model may be overfitting the data or missing important predictors that could improve the model's accuracy. It is sensitive to outliers: R-squared is sensitive to outliers in the data, which can cause it to overestimate or underestimate the true relation between the independent and dependent variables. It only applies to linear regression models: R-squared is only applicable to linear regression models, and cannot be used to evaluate the goodness of fit of other types of models, such as non-linear regression or time series models.

5.2.3 Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) is a measure of the average magnitude of the difference between the original values and the predicted values. Additionally, it gauges how the predictions deviate from the actual output, or the average magnitude of error. Additionally, MAE does not tell us whether we are underfitting the data or overfitting the data, i.e., the direction of error.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \widehat{y_i}| \tag{4}$$

MAE is a good option for datasets that contain outliers because it is less sensitive to outliers than other metrics like Mean Squared Error (MSE). MAE can be applied to non-linear models without a closed-form gradient solution. This makes it useful for assessing the effectiveness of models like random forests and decision trees

5.3 Results

5.3.1 R-Squared Comparison of models with & without features (60:40)

First of all, a study to determine the impact of features during model training was carried out using the data which we combined using the five conditions and five different cycles and trained models using data from five cycles in all five conditions and compared the R-squared metric using a training ratio of 6040 to choose the most accurate result between datasets that included features and those that did not. A bar graph was used to represent the mean R-squared values for the different models, and it was observed that the dataset including features produced an improved

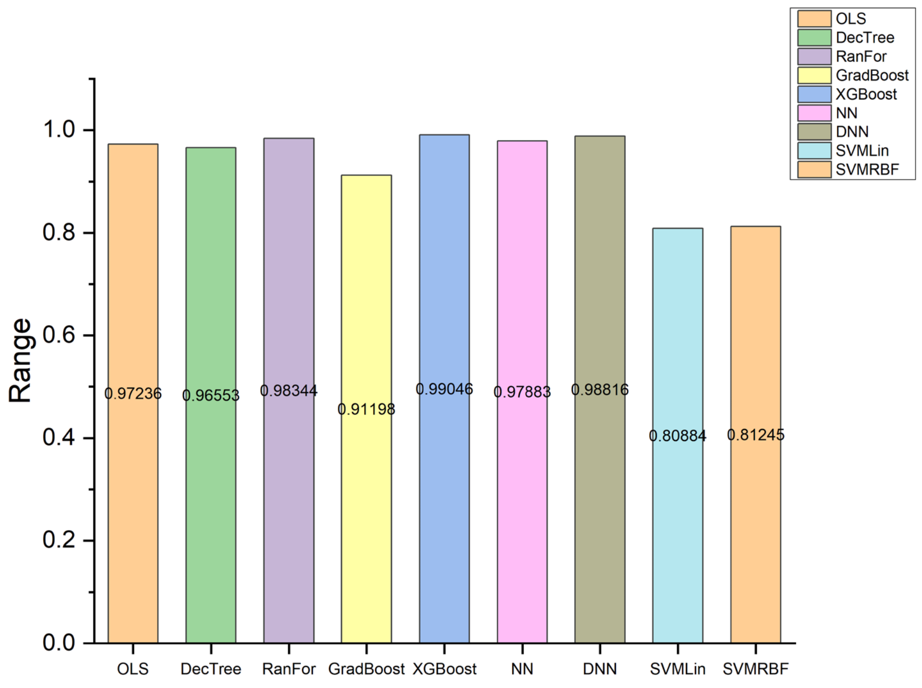


Figure 5.2 - R-Sq Error Chart with Features

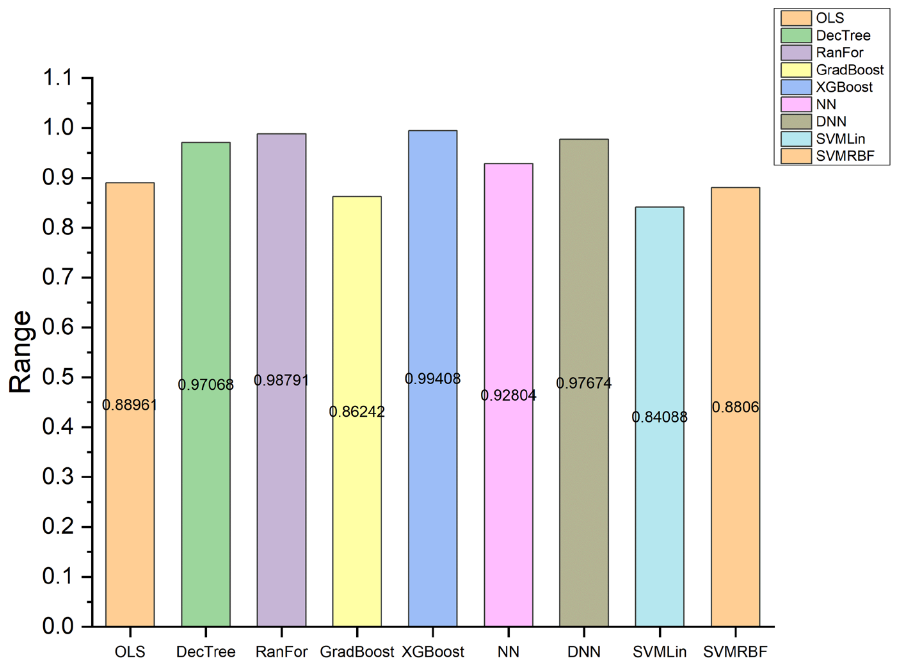


Figure 5.3 – R-Sq Error Chart without Features

results compared to the dataset with only input variables. The inclusion of features resulted in a 10% increase in model accuracy compared to the previous model.

Conclusion – Thus, it is now clear that the models with features provide a better result compared to former.

5.3.2 RMSE Comparison of 80:20, 70:30, and 60:40 training ratios with features

After we found the importance of features, we now moved on with the same dataset of five cycles and five different conditions to choose between a good training ratio.

Three histograms were made, to show the normal distribution of RMSE values for each algorithm & to show the count of RMSE values close to 0 for various algorithms. The histograms demonstrated that compared to the 60:40 training ratio and 70:30 training ratio, the 80:20 training ratio had a higher proportion of RMSE values close to 0 and higher peaks in the normal distribution curves. This suggests that when compared to the 60:40 and 70:30 training ratio, the 80:20 ratio produced better results in the output prediction.

Conclusion - The more RMSE values that are close to 0, the more accurate the model's predictions were with the 80:20 training ratio.

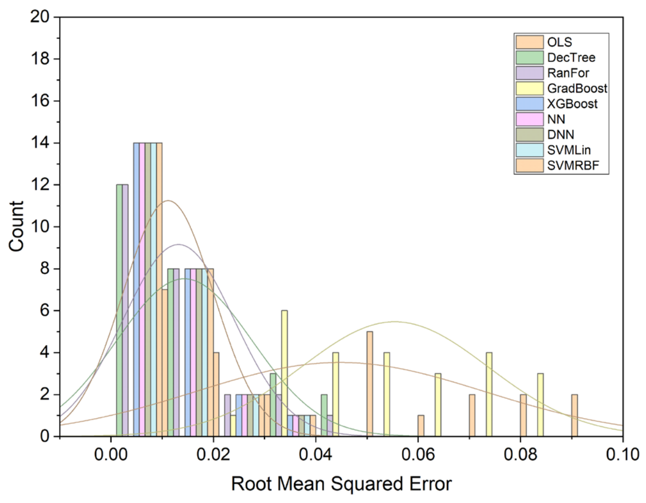


Figure 5.3 - RMSE Histogram of 80:20 Ratio

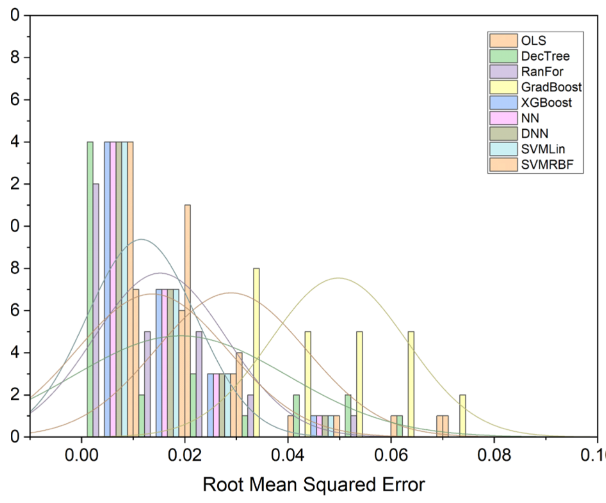


Figure 5.2 - RMSE Histogram of 60:40 Ratio

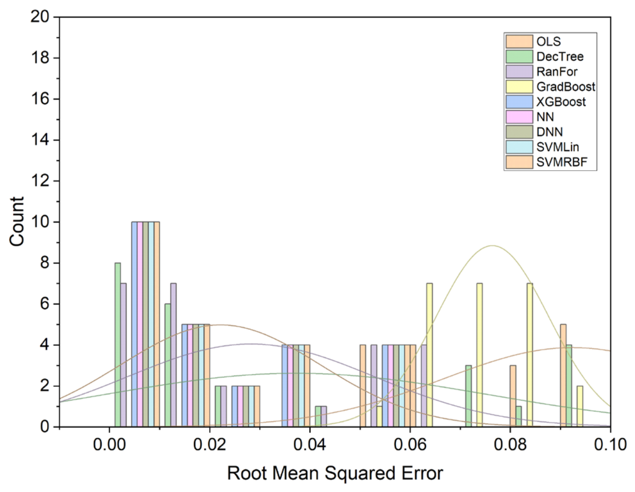


Figure 5.4 - RMSE of 70:30 Ratio

5.3.3 Selection of five algorithms by comparing their MAE and RMSE

Following an analysis of the fundamental ideas that form the strong performance foundations, we are now down to choosing five algorithms with an 80:20 training ratio and features. These graphs show two metrics: on the left, the mean MAE values across all algorithms under study are shown, and on the right, the mean RMSE values are shown. We compared these 2 metrics because they cannot be used alone to determine accuracy due to their limitations. For example, as we have already seen, RMSE takes into account outliers while MAE is not sensitive to them. Because there

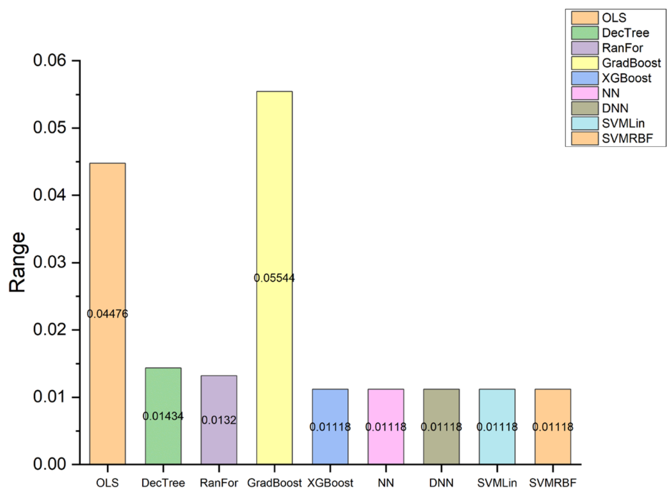


Figure 5.5 - Mean MAE Bar Chart

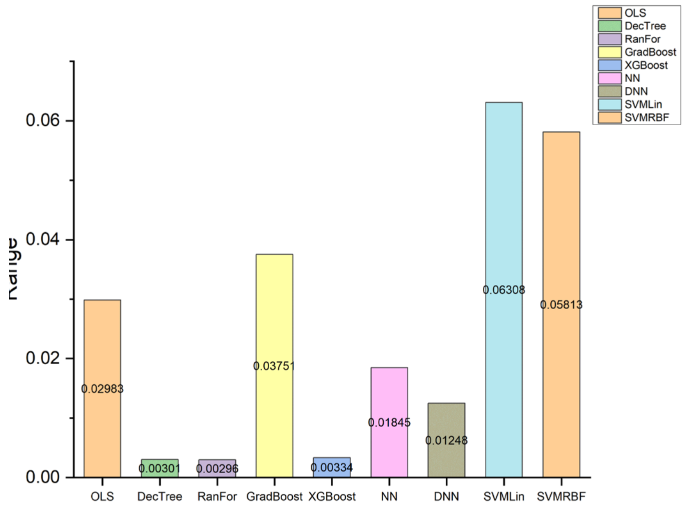


Figure 5.6- Mean RMSE Bar Chart

could be measurement error and the model could become inaccurate if it is overfitted, we need to be less sensitive to outliers in this case.

From these charts, it can be clearly seen that, Decision Tree, Random Forest, Extreme Gradient Boost, Neural Network, and Deep Neural Network provide the accurate results when compared to other algorithms because in both the evaluation metrics, the value of respective errors from all the algorithms is the least (close to zero).

5.3.4 Selection of three algorithms based on individual cycle analysis

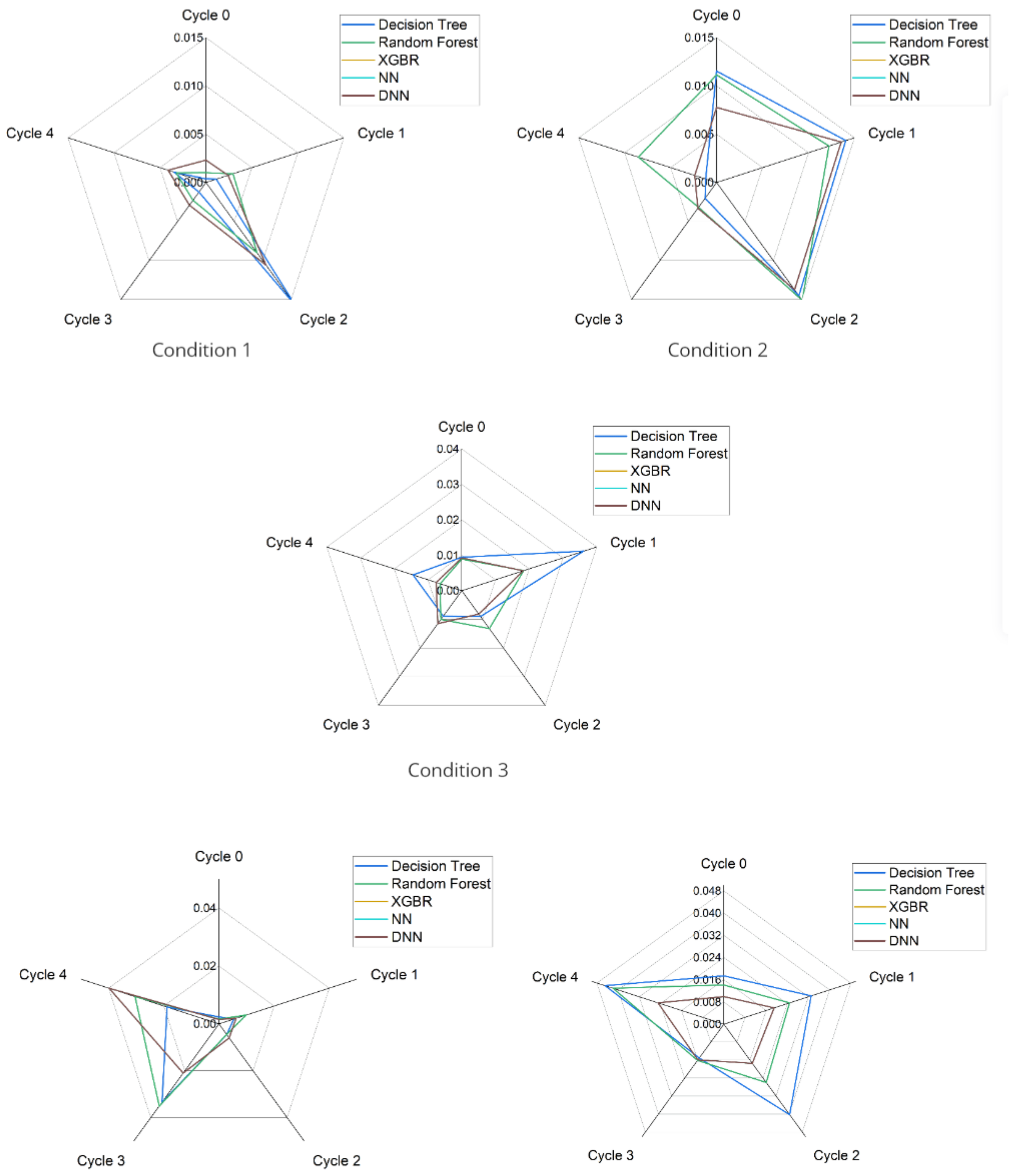


Figure 5.7 - Cycle Radar Charts

In order to conduct a more thorough analysis, we used those five algorithms in five cycles and produced five radar charts that showed how the models' RMSE varied over those five cycles. The radar charts show how closely the RMSE values are spaced in each cycle, creating a web-like structure. It is clear from this that the Extreme Gradient Boosting, Neural Networks and Deep Neural Network offer the best prediction among the five options. because these three algorithms' RMSE values are nearly zero, which is excellent.

5.3.5 Execution Time Analysis of top three algorithms

Execution time is crucial in a real-time battery monitoring system for accurately predicting battery health and calculating the battery's remaining life. Estimating the battery's remaining life is crucial for deciding when to charge or replace the battery to maintain continuous operation of the system or device. Additionally, where power consumption is a major concern, battery monitoring systems may be used in mobile devices or remote monitoring applications. The system will use more energy the longer it runs, potentially shortening the battery life even more.

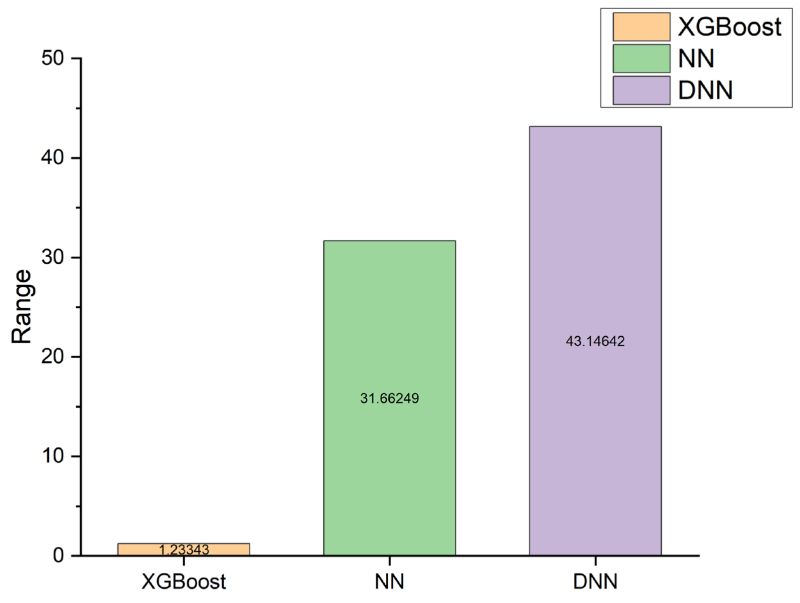


Figure 5.8 - Time vs Algorithms

Finally, the chart depicts that the fastest execution was done by XGBoost followed by Neural Network and then Deep Neural Network. As DNN is quite complex and has more hidden layers, it indeed is time consuming and is not good for this system, taking around 43 Seconds to predict an output compared to XGBoost which takes as low as 1.2 Seconds to predict the output.

Chapter 6 BMS Module

6.1 Hardware Requirements

6.1.1 Voltage Sensor – (Voltage Detection Sensor Module 25V).

A **Voltage Detection Sensor Module 25V [22]** is a device designed to measure the voltage of a circuit with a maximum voltage of 25V. It typically consists of a sensor, amplifier, and output interface, which enables a microcontroller or other device to read the measured voltage. Typically, the sensor consists of a voltage divider circuit that transforms the input voltage into a smaller, more easily measurable voltage. The amplifier amplifies the sensor's output signal to make it more accurate and easier to read, and the output interface permits a microcontroller or other device to read the measured voltage. This type of voltage sensor module is typically employed in applications where the input voltage is restricted to 25V or less, such as in battery-powered devices or low-voltage electrical systems. This module follows the design principle of resistive voltage divider and allows the input voltage at the red terminal connector to be reduced by a factor of 5. It is suitable for use with Arduino analog input voltages up to 5V, and the input voltage for the voltage detection module should not exceed 5V x 5 = 25V (or 3.3V x 5 = 16.5V for 3.3V systems). The Arduino AVR chips have 10-bit analog-to-digital (AD) conversion, which results in a simulation resolution of 0.00489V (5V/1023) for this module. Therefore, the minimum detectable input voltage for the voltage detection module is 0.00489V x 5 = 0.02445V.

6.1.2 Current Sensor (ACS712)

The **ACS712 [21]** is a module for measuring the current flowing through an electrical circuit. It consists of a Hall effect sensor, which works on the detection of magnetic field generated by a conductor's current, and an amplifier circuit, which amplifies the sensor's output signal. Depending on the model, the ACS712 can measure both AC and DC currents with a range of 5A, 20A, or 30A. The measured current can be output as an analogue voltage or a digital pulse-width modulation (PWM) signal, which can be read by a microcontroller or other device. Common applications for the ACS712 include motor control, overcurrent protection, and power supply design.

6.1.3 Temperature Sensor (LM35)

The LM35 is a temperature sensor used to measure the temperature of a physical object or the surrounding environment. It is a precision integrated-circuit temperature sensor, so it is extremely accurate and stable over a broad temperature range. With a scale factor of 10 mV/°C, the LM35 outputs an analogue voltage proportional to the temperature in Celsius. This means that the output voltage of the **LM35 [20]** will vary by 10 millivolts for every degree Celsius of temperature change. The LM35 is utilized in numerous temperature sensing applications, including thermostats, temperature-controlled fans, and temperature data loggers.

6.1.4 Orange 2000mAh 3S 25C Lithium Polymer Battery

The **Orange 2000mAh 3S [19]** 25C/50C Lithium Ion battery is a common rechargeable battery found in radio-controlled (RC) vehicles and other electronic devices. It has a capacity of 2000mAh, indicating that it can store a substantial quantity of energy. The "3S" designation denotes that the battery is made up of three lithium-ion cells connected in series, resulting in a nominal voltage of 11.1V. The designation "25C/50C" indicates that the battery can deliver a continuous discharge current of 25C or 50C, depending on the application. The "Orange" brand produces premium lithium ion rechargeable batteries, and the 2000mAh 3S 25C/50C battery is a popular option for RC vehicles and other high-performance electronic devices.

6.1.5 Microprocessor (Raspberry Pi 4)

The **Raspberry Pi [17]** Foundation designs and manufactures the Raspberry Pi 4, a small, low-cost, single-board computer. It was released in June 2019 and is the most recent instalment in the Raspberry Pi computer line. Common applications include a home media center, a do-it-yourself gaming console, and an inexpensive computer for education and programming. The Raspberry Pi 4 is a highly capable single-board computer that can perform a variety of tasks, such as media playback, gaming, and basic computing.

6.1.6 Power Distribution Board (PDB-XT60)

Within an electronic system, a power distribution board (PDB) is a device used to distribute electrical power to multiple circuits or devices. A **PDB-XT60 [16]** is a specific type of PDB that provides power to devices via XT60 connectors. XT60 connectors are a common type of high-current connector found on radio-controlled aircraft and vehicles. Typically, a PDB-XT60 is used to power the electronic speed controllers (ESCs) and other devices in a radio-controlled aircraft or vehicle. It assists in power distribution and protects against overcurrent and short-circuit conditions.

6.1.7 ADC Converter (ADS1115)

ADS1115 [18] is a 16-bit, extremely accurate, low-power analog-to-digital converter (ADC) that is frequently used in a variety of applications that call for precise measurement of analogue signals. The ADS1115 has four input channels with programmable gain amplifiers that support a wide range of input voltages and can convert analogue signals into digital data at up to 860 samples per second (SPS). For increased accuracy, it also includes an integrated voltage reference and temperature sensor. The I2C interface allows the ADS1115 to communicate with a microcontroller, making it simple to integrate into a variety of projects. ADS1115 is frequently used in sectors like medical equipment, industrial automation, and environmental monitoring due to its high precision and low power consumption.

6.2 Module Circuit

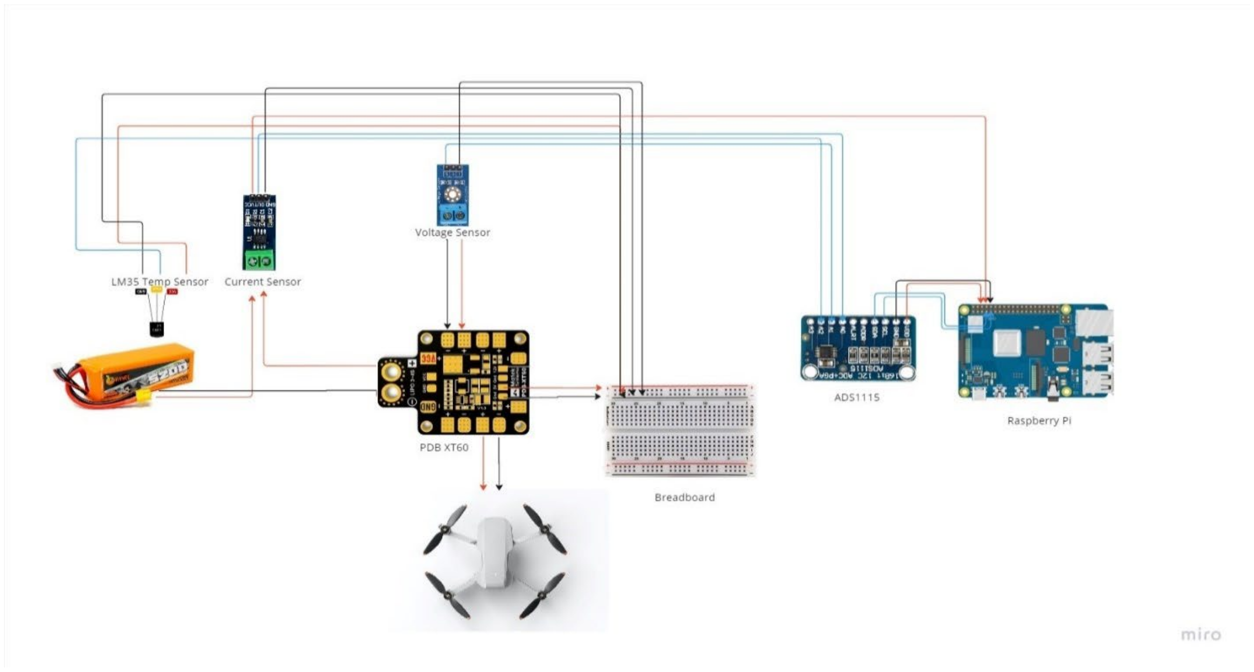


Figure 6.2 - Schematic Circuit

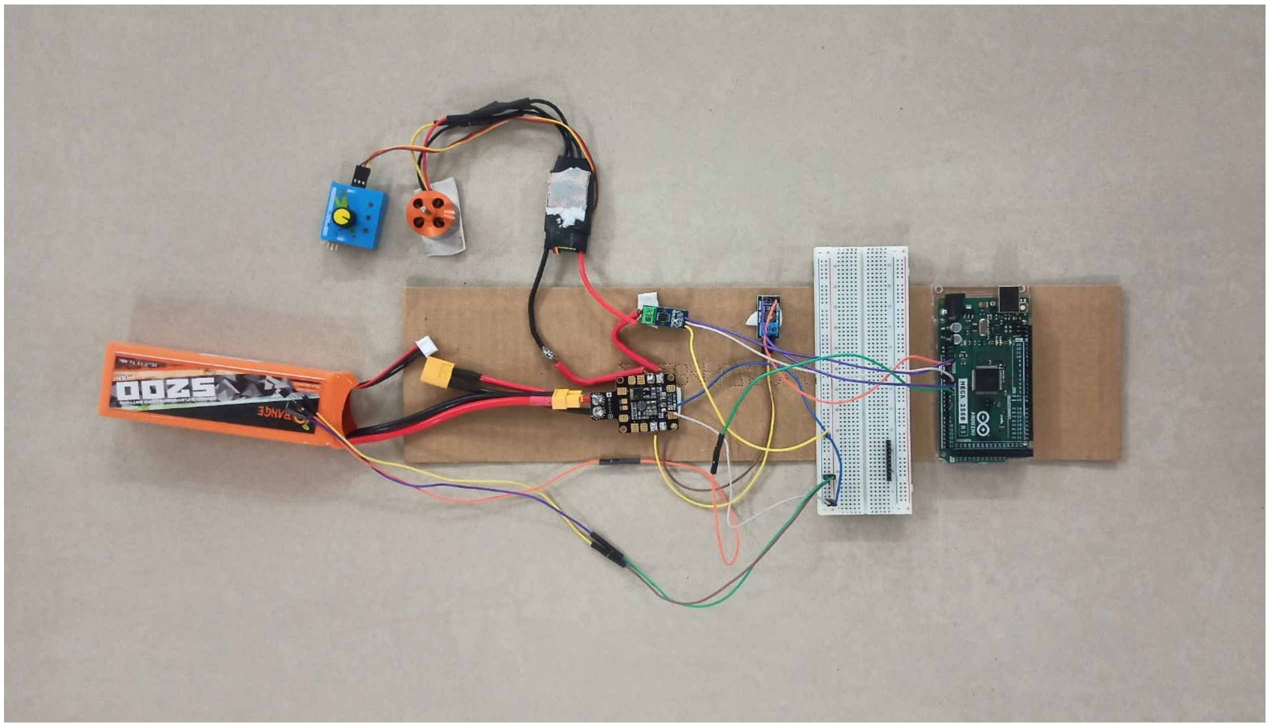


Figure 6.1 - Real Life Module

Chapter 7 Data Collection & Results

7.1 Data Collection using single motor

Now that we have a thorough understanding of how the Extreme Gradient Boosting Algorithms deliver accurate results while operating extremely quickly, we want to gather data for a single motor with propellers using our custom BMS Module. Therefore, to get started, we connected the module to a hexacopter using the circuit depicted above, then we operated a single motored and recorded the data for a full cycle.

A drone cycle includes

1. Takeoff - Normally, a drone takes 50 secs to takeoff and travels at a high burst speed of 1 to 1.5 m/s.

2. Cruise - Next, the drone is allowed to travel for a predetermined amount of time to ensure consistent battery life and efficiency.

3. Landing - The drone speeds up for another 50 seconds during the landing phase before its propellers are eventually turned off.

To train the model using this data set, a total of three cycles of a single motor run were run for the data collection.

7.2 Data Collection for multi motor runs

After confirming the module's functionality, it is now time to collect data for two and three motors in order to study how the system behaves when operating in multi-rotor systems like quadcopters and hexcopters. As more motors are running, the load on the battery and the system's overall power consumption will increase. Although there were some voltage reading fluctuations, the data from the two motors was largely linear. Three iterations were performed for the previous full cycles for this as well.

Regarding the three-motor dataset, it was a fairly accurate representation of real-world data collection since most drones have four motors and the data won't be linear when using four motors. And since the dataset was quite nonlinear, we did in fact discover a similar pattern in it. We performed six iterations of this scenario, each iteration lasting until the battery dropped from 11.1 volts to 9 volts as a safety measure. For the purposes of testing and training machine learning, this data was combined.

Unfortunately, the current sensor only measured currents up to 20 Amps, and the load when a quadcopter is connected to the module exceeds 20 Amps. As a result, we implemented three motors to observe the differences.

7.3 Noise

Now as our aim is to test the system in real life conditions, there will always be some kind of noise during sensor readings or any environmental conditions may impact it. Considering this, we introduced noise to our data set and then implemented the models on it. The term "noise" in a dataset describes the presence of random or irrelevant data points that do not accurately reflect any underlying patterns or relationships. Several things, including measurement errors, data entry errors, and outliers, can contribute to noise. Because noise can result in false conclusions or predictions, it can have a negative effect on the accuracy and reliability of statistical models and machine learning algorithms. Therefore, before beginning any data analysis or modelling, it is crucial to recognise and eliminate noise from a dataset.

7.3.1 Gaussian Noise

A common type of statistical noise used in signal processing and communication systems is referred to as gaussian noise. A Gaussian or normal distribution is followed by the probability density function (PDF) of this particular type of noise.

Gaussian noise is frequently used in machine learning as a regularization technique to avoid overfitting the model to the training data. During training, it is added to the model's input data, which aids in teaching it more robust and generalizable features. The standard deviation in Gaussian noise has a big impact because it controls how much noise is added to the input data.

More noise is introduced as a result of a higher standard deviation, which can make it more challenging to draw conclusions from the input data. A lower standard deviation, on the other hand, leads to less noise being added, which may make it simpler for the model to absorb the input data.

So, for this 3Motors Dataset, we added three deviations—0.0125, 0.025, and 0.05—as noise and compared how different algorithms behaved. The findings of this study are then presented after this section.

7.4 Evaluation Results

First, the Mean Absolute Error (MAE) for each of the three types of noise was examined and contrasted using a bar graph. Understanding which algorithm is operating accurately in each of the three conditions is made easier thanks to this statistical representation. As a result, the following charts show the MAE of 0.0125, 0.025, and 0.05 noise results (from the left). We can identify the top three algorithms under these circumstances by comparing them.

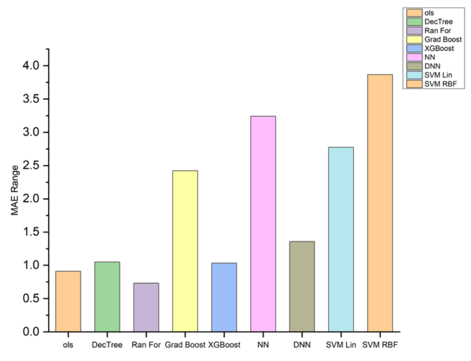


Figure 7.1 - 0.0125 SD Results

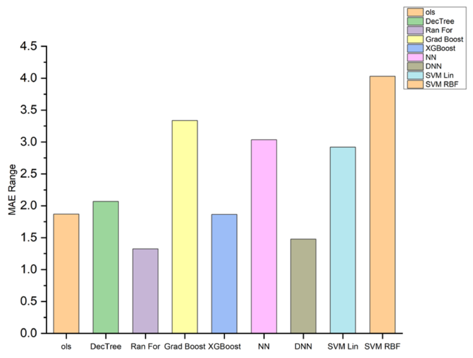


Figure 7.2 - 0.025 SD Results

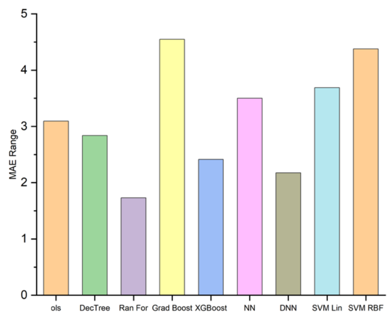


Figure 7.3 - 0.05 SD Results

The aim of this analysis is to determine which algorithms were accurate with increasing noise. Therefore, it is evident from this chart that, out of the nine algorithms, the Random Forest, XGBoost, and Deep Neural Network provided good accuracy.

Additionally, as previously mentioned, their execution time is crucial in determining real-time results; consequently, it was discovered during the time analysis that the XGBoost predicts the results quickly. In all three scenarios, DNN takes around 3.5 seconds to complete its prediction, whereas XGBoost only needs 0.05 seconds. This is an excellent method.

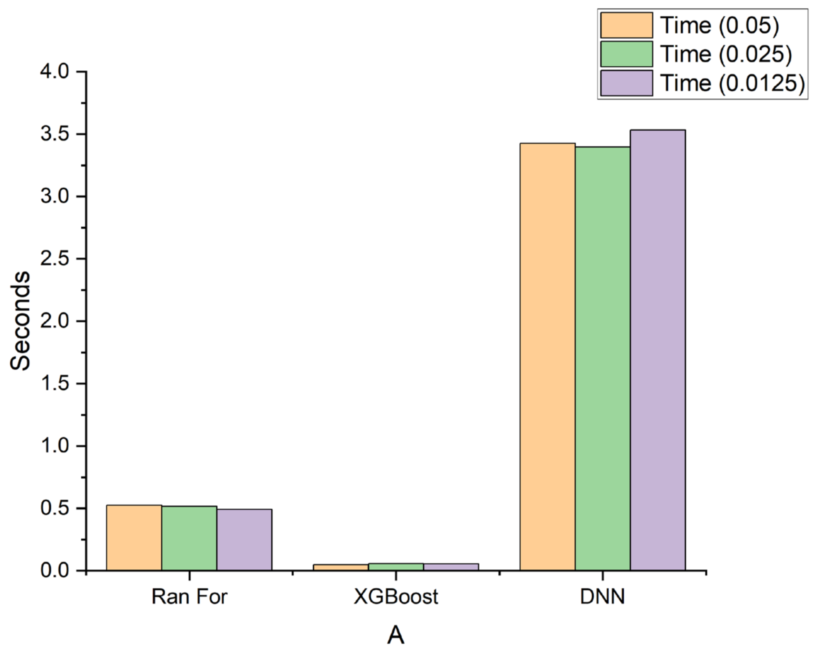


Figure 7.4 - Time Analysis for 3 Algorithms

Chapter 8 Real-time Interfacing

Based on the circuit diagram from the previous section, an Analogue to Digital Converter is crucial for connecting the Raspberry Pi to the circuit and reading the data because the sensors produce analogue signals, and the microcontroller does not have an internal ADC. The Raspberry Pi will additionally process the incoming data, run the model, and forecast the State of Charge in a matter of seconds. Pi's primary responsibility in this project is to gather data, preprocess it, use it in a specific way, and forecast the results.

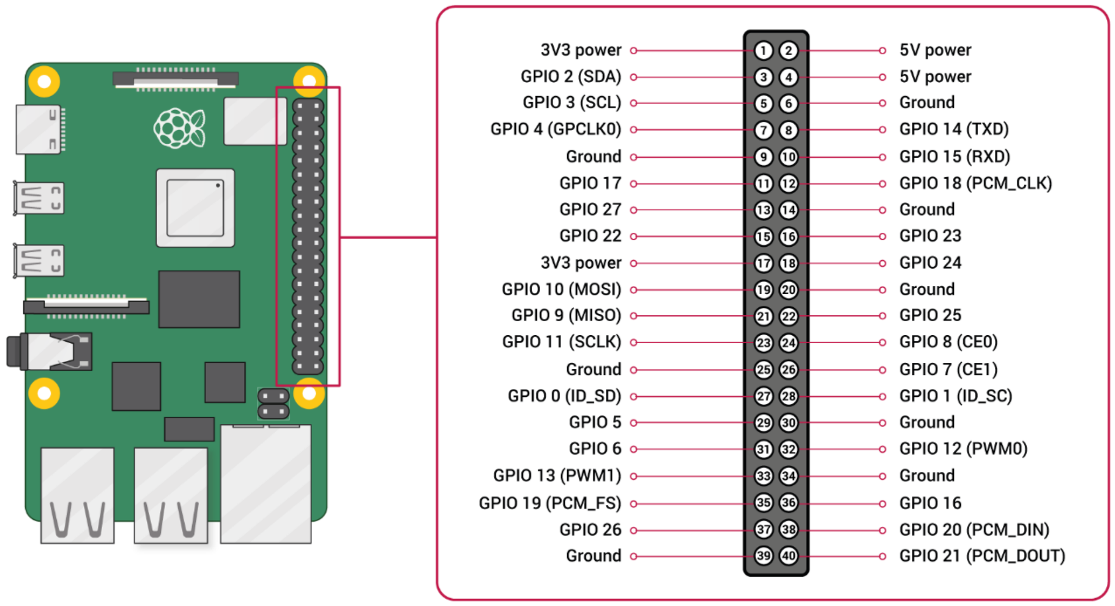


Figure 8.1 - Pi's GPIO Pins

The figure above shows the Graphical Input/Output Pin Headers of a Raspberry Pi

8.1 I2C Communication

When an ADC is connected to raspberry pi, it used the I2C Communication Protocol to transfer and receive the data from the sensors. Two of the most crucial pins used in the I2C communication protocol are SCL and SDA. Serial Data is referred to as SDA, and Serial Clock is referred to as SCL. In an I2C network, these two pins are used to transmit and receive data between devices.

SCL is a signal that gives devices the timing for communication. It transmits a square wave signal that establishes the communication's frequency. The speed of communication is determined by the signal's frequency. The master device in the I2C network produces the SCL signal.

SDA, on the other hand, is the signal that travels between devices and carries data. Each packet or frame that contains the data has a start bit, eight bits of data, an acknowledge bit, and a stop bit. Since the SDA signal is bidirectional, data can be sent and received using it.

8.2 ADS1115 with Raspberry PI and Display

It's easy to connect the ADS1115 to a microcontroller. Since we are reading voltage and current sensors up to 5V, the ADC must first be powered using the 5V pin on the Pi. The other pins of the ADC must be connected with the sensor output pins, which are P0 with the current sensor, P1 with the voltage sensor, and P2 with the temperature sensor, in order to take readings from them. I2C communication will then transfer the readings to the PI in order to perform the necessary calculations.

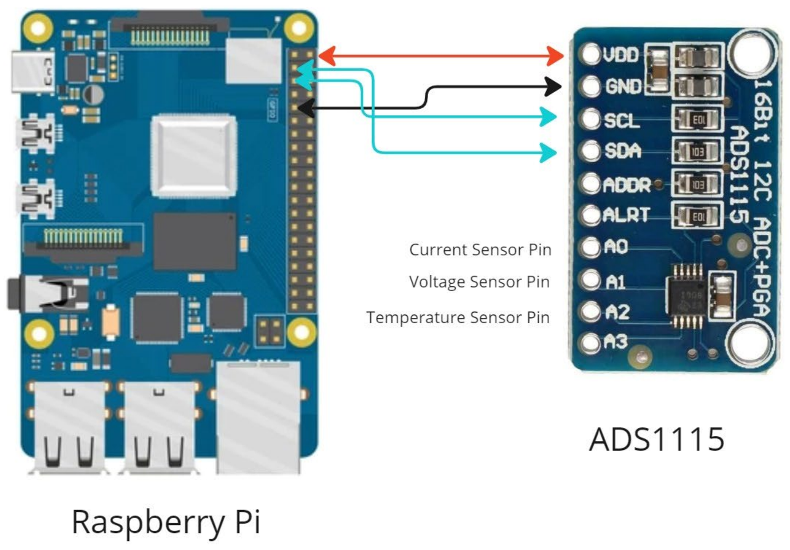


Figure 8.2 - ADS1115 with Pi

And to get the display of what the results are, we connected the desktop with HDMI cable and performed the machine learning using the visual interface.

8.3 Code & Documentation

We've created an entire GitHub Repository for the documentation of this work which includes the code files, datasets and other important resources.

It can be found here – Real Time Battery Monitoring System GitHub

8.4 Results

We applied this algorithm to the same three-motor drone module that was used for data collection, and we saw generally positive results. However, the accuracy could be increased, which will be covered in more detail later.

To start, we mimicked three conditions.

High Speed Condition	As the drone takes off, it experiences a high burst and maximum throttle, resulting in a high current flow and a quick loss of battery capacity.
Cruise Condition	A drone on a cruise basically travels at a constant speed while its battery power steadily depletes. This is another situation occurs in real life and contributes to battery loss.
Variable Speed Conditions	Real-world conditions are a little trickier because they frequently involve unforeseen events like sudden gusts of wind, high altitudes, etc. that necessitate adjusting the drone's speed. Therefore, real-time prediction algorithms need to take this scenario into account.

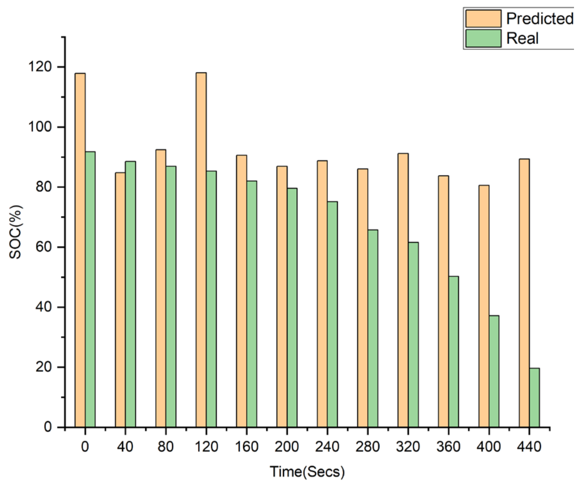


Figure 8.2 - Real vs Predicated SOC at Variable Conditio

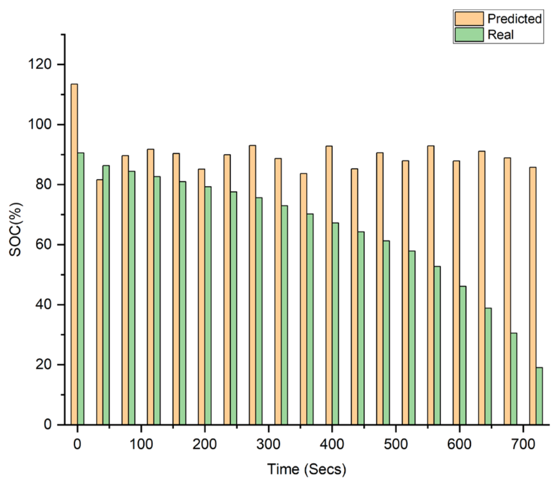


Figure 8.3 - Real vs Predicted SOC at Cruise Condition

The images mentioned above show the actual and predicted values for the variable condition and cruise condition states of charge. It is clear that the algorithm behaves inappropriately and generates a significant amount of error at low states of charge. Similar results can be seen when travelling; the prediction is less accurate at lower battery percentages, and as a result, error increases at lower state of charge.

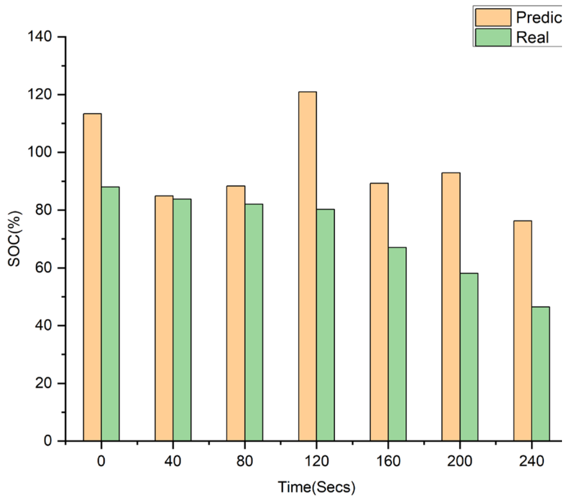


Figure 8.4-Real vs Predicated SOC at High Speed

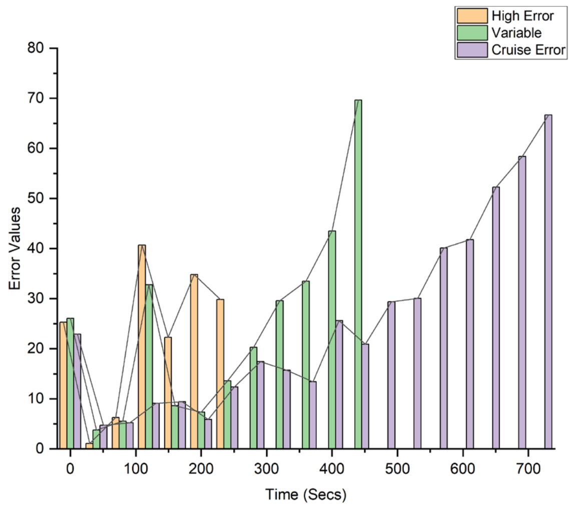


Figure 8.5 - Error values of three conditions

A specific pattern in the predicted values cannot be seen in the scenario for high-speed conditions, but eventually it can be seen that the error increases at lower percentages of battery state.

Figure 8.5 represents the error values for the three conditions. And following the similar patterns, it shows that when the cycle ends, the error increases for all the three scenarios and thus now it is evident that the algorithm is lacking accuracy at low state of charge levels. The main reason for these low accuracy at particular charging level is the lack of datapoints at these low levels. Due to this the model is not able to train at these lower levels and thus prediction is also inaccurate.

Chapter 9 Conclusion and Future Work

9.1 Conclusion

This project involved more than just using machine learning in real-time; it also involved selecting the best algorithm to deliver accurate results by feature engineering data preprocessing, data collection at various stages, and simulating real-world conditions using various techniques like noise, drone testing, etc. Knowing how to integrate sensors and software systems, taking into account training ratios while analyzing the effects on the results, monitoring displays in real-time, combining data collection and prediction at the same time, and ultimately obtaining the final State of Charge are all skills that are required.

We acquired a variety of skills over the course of the entire period, which made the project challenging for both of us. The project's goal was to reduce the price of a typical BMS circuit using arcane circuitry and to make it simpler using simple sensors and artificial intelligence. By drawing the conclusions listed below, we were able to accomplish the project's goal.

1. To find the best algorithms for this system by obtaining a suitable dataset that can be used for an offline test of various algorithms.
2. Machine learning techniques were helpful for this project, and an offline analysis of the dataset led to the discovery of a more reliable model.
3. Using various hardware parts, a straightforward BMS Module was created that was nearly flawless given the connection with various components.
4. 4. The module did quite well during the data collection stage, which was important because if there were errors in the dataset, the model would be trained to produce inaccurate results, and the prediction would be uninspiring.
5. Finally, the BMS module's interface with the microcontroller was successful because it quickly and accurately predicted the desired real-time result.

9.2 Future Work

This project focused on predicting the State of Charge of a battery while a load is connected to it. More features can be added to this system and prospects are listed below

1. As a battery works continuously, there are chances of high heat generation and thus a heat sink can be made. A similar AI driven heat sink which activated upon a certain temperature range and helps the battery maintain a constant temperature would suffice the need.
2. Use of high ended sensors which can measure to a quite high range of current and voltage will increase the versatility of this system and can be useful in any electronics devices. Currently it is limited to smaller dimensional aero vehicles like drones.
3. Also, if we collect the data for 2S and 4S batteries, this system can also be run on the devices using these batteries, which will also add up to its versatility.

Bibliography

1. Vilsen, S. B., & Stroe, D.-I. (2021). Battery state-of-health modelling by multiple linear regression. *Journal of Cleaner Production*, *290*, 125700. https://doi.org/10.1016/j.jclepro.2020.125700

2. Lelie, M., Braun, T., Knips, M., Nordmann, H., Ringbeck, F., Zappen, H., & Sauer, D. (2018). Battery Management System Hardware Concepts: An Overview. *Applied Sciences*, *8*(4), 534. https://doi.org/10.3390/app8040534

3. *Battery Management Systems* (Vol. 9). (2008). Springer Netherlands. https://doi.org/10.1007/978-1-4020-6945-1

4. Ardeshiri, R. R., Balagopal, B., Alsabbagh, A., Ma, C., & Chow, M.-Y. (2020). Machine Learning Approaches in Battery Management Systems: State of the Art: Remaining useful life and fault detection. *2020 2nd IEEE International Conference on Industrial Electronics for Sustainable Energy Systems (IESES)*, 61–66. https://doi.org/10.1109/IESES45645.2020.9210642

5. Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*(1), 53. https://doi.org/10.1186/s40537-021-00444-8

6. Ying LU, Song Y., (2015). Decision tree methods: applications for classification and prediction. Shanghai Archives of Psychiatry, https://doi.org/10.11919%2Fj.issn.1002-0829.215044

7. Henrique, B. M., Sobreiro, V. A., & Kimura, H. (2018). Stock price prediction using support vector regression on daily and up to the minute prices. *The Journal of Finance and Data Science*, *4*(3), 183–201. https://doi.org/10.1016/j.jfds.2018.04.003

8. Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222. https://doi.org/10.1023/B:STCO.0000035301.49549.88

9. Pahlavan-Rad, M. R., Dahmardeh, K., Hadizadeh, M., Keykha, G., Mohammadnia, N., Gangali, M., Keikha, M., Davatgar, N., & Brungard, C. (2020). Prediction of soil water infiltration using multiple linear regression and random forest in a dry flood plain, eastern Iran. *CATENA*, *194*, 104715. https://doi.org/10.1016/j.catena.2020.104715

10. Chen, T., & Guestrin, C. (2016). XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

11. Vilsen, S. B., & Stroe, D.-I. (2021). Battery state-of-health modelling by multiple linear regression. *Journal of Cleaner Production*, *290*, 125700. https://doi.org/10.1016/j.jclepro.2020.125700

12. Ardeshiri, R. R., Balagopal, B., Alsabbagh, A., Ma, C., & Chow, M.-Y. (2020). Machine Learning Approaches in Battery Management Systems: State of the Art: Remaining useful life and fault detection. *2020 2nd IEEE International Conference on Industrial Electronics for Sustainable Energy Systems (IESES)*, 61–66. https://doi.org/10.1109/IESES45645.2020.9210642

13. Yang, H., Cao, Y., Xie, H., Shao, S., Zhao, J., Gao, T., Zhang, J., & Zhang, B. (2020). Lithium-ion Battery Life Cycle Prediction with Deep Learning Regression Model. *2020 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 3346–3351. https://doi.org/10.1109/APEC39645.2020.9124049

14. Ng, M.-F., Zhao, J., Yan, Q., Conduit, G. J., & Seh, Z. W. (2020). Predicting the state of charge and health of batteries using data-driven machine learning. *Nature Machine Intelligence*, *2*(3), 161–170. https://doi.org/10.1038/s42256-020-0156-7

15. https://www.tytorobotics.com/blogs/articles/a-guide-to-lithium-polymer-batteries-for-drones

16. PDB-XT60 with BEC 5V and 12V buy online at Low Price in India ElectronicsComp.com

17. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

18. https://www.adafruit.com/product/1085

19. https://robu.in/product/orange-2200mah-3s-30c60c-lithium-polymer-battery-pack-lipo/?gclid=CjwKCAjwge2iBhBBEiwAfXDBR4pOSEadZgp_gMOWm4ylOhucrmzIDLP05IJ6__lt7OSG84o77PjJERoCn3oQAvD_BwE

20. https://electronicguidebook.com/what-is-a-lm35/

21. https://www.allegromicro.com/en/Products/Sense/Current-Sensor-ICs/Zero-To-Fifty-Amp-Integrated-Conductor-Sensor-ICs/ACS712

22. https://robocraze.com/products/voltage-sensor-module-25v-pack-of-25

23. LG 18650HG2 Li-ion Battery Data and Example Deep Neural Network xEV SOC Estimator Script - Mendeley Data

24. Support Vector Machines — scikit-learn 1.2.2 documentation

25. Deep Neural Network - an overview | ScienceDirect Topics

26. Chien, J.-T. (2019). Deep Neural Network. In *Source Separation and Machine Learning* (pp. 259–320). Elsevier. https://doi.org/10.1016/B978-0-12-804566-4.00019-X

27. Neural Networks | A beginners guide – GeeksforGeeks

28. Introduction to Boosted Trees — xgboost 2.0.0-dev documentation

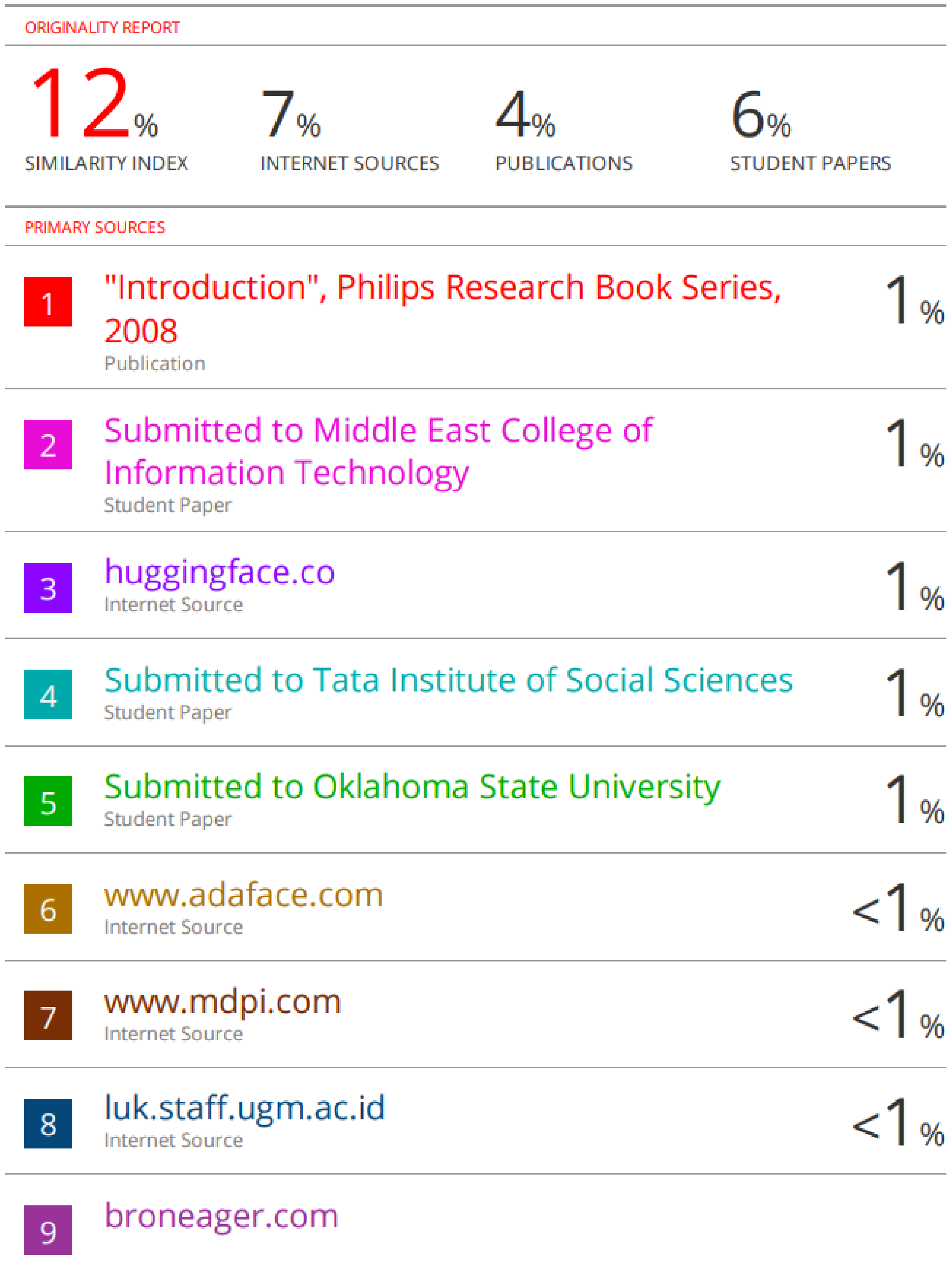
29. A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning - MachineLearningMastery.com

30. What is Random Forest? | IBM

31. Decision Tree – GeeksforGeeks

32. Ordinary Least Squares Definition | DeepAI

Certificate



ORIGINALITY REPORT

12%	7%	4%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Introduction", Philips Research Book Series, 2008 Publication	1%
2	Submitted to Middle East College of Information Technology Student Paper	1%
3	huggingface.co Internet Source	1%
4	Submitted to Tata Institute of Social Sciences Student Paper	1%
5	Submitted to Oklahoma State University Student Paper	1%
6	www.adaface.com Internet Source	<1%
7	www.mdpi.com Internet Source	<1%
8	luk.staff.ugm.ac.id Internet Source	<1%
9	broneager.com	