

AMBIENT TEMPERATURE PREDICTION OF ELECTRIC MOTOR USING MACHINE LEARNING

*Minor project report submitted
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology
in
Computer Science & Engineering**

By

UTPAL KUMAR SHARMA	(18UECS0886)
RITESH RANJAN	(18UECS0736)
KUMKUM KUMARI	(18UECS0443)

*Under the guidance of
Ms.M.ShyamalaDevi,M.E.,Ph.D.,
PROFESSOR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING
VEL TECH RANGARAJAN Dr.SAGUNTHALA R&D
INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be University Estd u/s 3 of UGC Act, 1956)**

**Accredited by NAAC with A Grade
CHENNAI 600 062, TAMILNADU, INDIA**

June,2021

CERTIFICATE

It is certified that the work contained in the project report titled "Ambient Temperature Prediction of Electric Motor using Machine Learning" by "UTPAL KUMAR SHARMA (18UECS0886) RITESH RANJAN (18UECS0736) KUMKUM KUMARI (18UECS0443)" has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor

Dr.M.ShyamalaDevi

Designation

Computer Science & Engineering

School of Computing

Vel Tech Rangarajan Dr.Sagunthala R&D

Institute of Science and Technology

June,2021

Signature of Head of the Department

Dr. V. Srinivasa Rao

Professor & Head

Computer Science & Engineering

School of Computing

Vel Tech Rangarajan Dr.Sagunthala R&D

Institute of Science and Technology

June,2021

DECLARATION

We declare that this written submission represents my ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

UTPAL KUMAR SHARMA

Date: / /

(Signature)

RITESH RANJAN

Date: / /

(Signature)

KUMKUM KUMARI

Date: / /

APPROVAL SHEET

This project report entitled (Ambient Temperature Prediction of Electric Motor using Machine Learning) by (UTPAL KUMAR SHARMA(18UECS0886), (RITESH RANJAN (18UECS0736), (KUMKUM KUMARI (18UECS0443) is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners

Supervisor

Dr.M.ShyamalaDevi, M.E.,Ph.D.,

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (EEE), B.E. (MECH), M.S (AUTO). DSc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Chairperson Managing Trustee and Vice President.

We are very much grateful to our beloved **Vice Chancellor Prof. S. SALIVAHANAN**, for providing us with an environment to complete our project successfully.

We record indebtedness to our **Dean & Head, Department of Computer Science and Engineering Dr. V. SRINIVASA RAO, M.Tech., Ph.D.**, for immense care and encouragement towards us throughout the course of this project.

We also take this opportunity to express a deep sense of gratitude to Our Internal Supervisor **Dr.M.ShyamaDevi,M.E.,Ph.D.**, for her cordial support, valuable information and guidance, he helped us in completing this project through various stages.

A special thanks to our **Project Coordinators Mr. V. ASHOK KUMAR, M.Tech., Ms. C. SHYAMALA KUMARI, M.E., Ms.S.FLORENCE, M.Tech.**, for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

UTPAL KUMAR SHARMA	(18UECS0886)
RITESH RANJAN	(18UECS0736)
KUMKUM KUMARI	(18UECS0443)

ABSTRACT

As we know that Plan temperature evaluations apply to the most smoking spot inside the motor's windings, not how much of that warm is exchanged to the motor's surface. The warm exchange will change incredibly from engine to engine based on outline measure and mass, whether the outline is smooth or ribbed, whether open or completely encased, and other cooling variables. Indeed the productivity of the engine may have small impact on the surface temperature. An exact torque assess leads to more exact and satisfactory control of the engine, diminishing control misfortunes and in the long run heat build-up. In this venture, the machine learning strategies were utilized for the expectation of surrounding temperature of Electric engine. The forecast of ambient temperature of Electric engine are accomplished in four ways. With this context, we have utilized Electric motor temperature dataset extracted from UCI Machine Learning repository for predicting the ambient temperature prediction. The forecasting of ambient temperature are achieved in four ways. Firstly, the data set is preprocessed with Feature Scaling and missing values. Secondly, empirical feature examination is done and the relation of motor speed and ambient temperature of the motor is visualized. Thirdly, the fresh data set is fitted to all the regressors and the execution is dismembered before and after scaling. Fourth, the raw data set is subjected to Convolutional neural network Convid with various activation layers like Relu, Sigmoid, softmax, Softplus, Softsign, Tanh, Selu, Elu and exponential layers. The performance is analyzed with EVS, MAE, MSE, RScore and Step loss of the Convolutional neural network. The execution is

done using python language under Spyder platform with Anaconda Navigator. Experimental results shows that the Conv1D-Softsign activation layer tends to reach the RScore of 99.842 with the step loss of 0.0001155.

**Keywords:Anaconda Navigator,Spyder Python,Machine Learning
Numpy tools,Packages**

LIST OF FIGURES

4.1	Architecture Diagram	11
4.2	Data Flow Diagram	12
4.3	UML Diagram	13
4.4	Use Case Diagram	14
4.5	Class Diagram	15
4.6	Sequence Diagram	16
5.1	Input	19
5.2	Output	20
5.3	Test Image	23
6.1	Comparision	26
6.2	Sample Code	27
6.3	Output 1	28
6.4	R2 Score before and after Scaling	29
6.5	Regressor performance of the raw dataset before scaling	30
6.6	Regressor performance of the raw dataset after scaling	31
8.1	Poster Presentation	33
9.1	Poster Presentation	50

LIST OF ACRONYMS AND ABBREVIATIONS

IDE	Integrated Development Environment
GUI	Graphica User Interfacel

TABLE OF CONTENTS

	Page.No
ABSTRACT	v
LIST OF FIGURES	vii
LIST OF TABLES	viii
LIST OF ACRONYMS AND ABBREVIATIONS	viii
1 INTRODUCTION	1
1.1 Introduction	1
1.2 Aim of the project	2
1.3 Project Domain	2
1.4 Scope of the Project	2
1.5 Methodology	2
1.5.1 DATA COLLECTION	2
1.5.2 DATA ANALYSIS	3
2 LITERATURE REVIEW	4
3 PROJECT DESCRIPTION	7
3.1 Existing System	7
3.2 Proposed System	7
3.3 Feasibility Study	8

3.3.1	Economic Feasibility	9
3.3.2	Technical Feasibility	9
3.3.3	Social Feasibility	9
3.4	System Specification	10
3.4.1	Hardware Specification	10
3.4.2	Software Specification	10
3.4.3	Standards and Policies	10
4	MODULE DESCRIPTION	11
4.1	General Architecture	11
4.2	Design Phase	12
4.2.1	Data Flow Diagram	12
4.2.2	UML Diagram	13
4.2.3	Use Case Diagram	14
4.2.4	Class Diagram	15
4.2.5	Sequence Diagram	16
4.3	Module Description	16
5	IMPLEMENTATION AND TESTING	18
5.1	Input and Output	18
5.1.1	Input Design	19
5.1.2	Output Design	20
5.2	Testing	20
5.3	Types of Testing	21
5.3.1	Unit testing	21
5.3.2	Integration testing	21
5.3.3	Functional testing	21
5.3.4	White Box Testing	21
5.3.5	Black Box Testing	22

5.3.6	Test Result	23
5.4	Testing Strategy	23
6	RESULTS AND DISCUSSIONS	25
6.1	Efficiency of the Proposed System	25
6.2	Comparison of Existing and Proposed System	26
6.3	Advantages of the Proposed System	26
6.4	Sample Code	27
7	CONCLUSION AND FUTURE ENHANCEMENTS	32
7.1	Conclusion	32
7.2	Future Enhancements	32
8	PLAGIARISM REPORT	33
9	SOURCE CODE & POSTER PRESENTATION	34
9.1	Source code	34
9.2	Poster Presentation	50
	References	50

Chapter 1

INTRODUCTION

1.1 Introduction

The foremost curious target highlights are rotor temperature ("pm"), stator temperatures ("stator*") and torque. Particularly rotor temperature and torque are not dependably and financially quantifiable in a commercial vehicle. Being able to have solid estimators for the rotor temperature makes a difference the car industry to fabricate engines with less fabric and empowers control methodologies to utilize the engine to its greatest capability. A premium-efficiency engine, in spite of the fact that its inner temperature will be cooler as a result of lower misfortunes, may not have lower surface temperatures, since the ventilation fan will likely be littler to diminish windage misfortunes. Motor's surface isn't the way to judge working temperature, a motor's winding temperature is critical. The concern, of course, is for the astuteness of the engine stator's separator framework. Its work is to partitioned electrical components from each other, avoiding brief circuits and, hence, winding burnout and disappointment.

1.2 Aim of the project

To develop a software to know the ambient temperature of the electric motor and processed by machine learning algorithms to know the different aspects of electric motor like prevent motor insulation ageing, insulation burning, permanent magnet demagnetization and other faults caused by high stator winding temperature

1.3 Project Domain

Our domain reputation analysis reveals that the optimization of the motor performance and will give warnings when the motor temperature is abnormally high (above ambient temperature).

1.4 Scope of the Project

By using machine learning techniques and algorithms the values will be analyzed to predict the Ambient Temperature of electric motor, based on the running duration and heat produced by the motor to increase the motor life in any industries and automobiles.

1.5 Methodology

1.5.1 DATA COLLECTION

The data collection is the process of selecting the data for the Ambient Temperature of motor. In this paper, the data set is collected and given to the system to study and predict the new data values using the machine learning algorithms.

1.5.2 DATA ANALYSIS

The electric motor temperature dataset extricated from the UCI machine learning store is utilized for usage. The dataset comprises of 9,98,8071 information with following 12 autonomous highlights

- (i) Ambient- Encompassing temperature as measured by a warm sensor found closely to the stator.
- (ii) Coolant - Coolant temperature. The engine is water cooled. Estimation is taken at outflow.
- (iii) Ud- Voltage d-component
- (iv) U-q - Voltage q-component
- (v) Motor speed
- (vi) Torque - Torque induced by electricity.
- (vii) Id – electricity d component
- (viii) Iq – electricity d component
- (ix) statoryoke - Stator yoke burden temperature measured with a warm sensor.
- (x) statortooth - Stator tooth burden temperature measured with a warm sensor.
- (xi) statorwinding - Stator wind burden temperature measured with a warm sensor.
- (xii)Profile id – identifier of the electric motor machine

Chapter 2

LITERATURE REVIEW

A dataset is considered having genuine time information of surrounding temperature, coolant temperature, coordinate pivot and quadrature hub voltage and current, burden temperature, rotor temperature and stator temperature for forecast of engine speed and torque. This dataset is collected from the test seat of College of Paderbon research facility. Different machine learning models have been connected on the dataset. The result appears that Fine Tree is the finest show for forecast of both speed and torque of the changeless magnet synchronous engine having most reduced RMSE of 0.029224 and 0.052538 for expectation of speed and torque individually [1]

Lifted temperatures constrain the top execution of frameworks since of visit mediations by warm throttling. Non-uniform warm states over framework hubs moreover cause execution variety inside apparently comparable hubs driving to significant debasement of in general execution. In this paper we display a system for making a lightweight warm prediction system reasonable for run-time administration choices. We seek after two roads to investigate optimized lightweight warm indicators. First, we utilize highlight choice calculations to progress the execution of already planned machine learning strategies. Moment, we develop elective strategies utilizing neural arrange and straight regression-

based strategies to perform a comprehensive comparative study of expectation strategies [2].

The precise assessment and forecast of stator winding temperature is of incredible centrality to the security and unwavering quality of PMSMs. In arrange to ponder the impacting components of stator winding temperature and avoid engine cover maturing, separator burning, lasting magnet demagnetization and other deficiencies caused by tall stator winding temperature, we propose a computer show for PMSM temperature forecast. Surrounding temperature, coolant temperature, direct-axis voltage, quadrature-axis voltage, engine speed, torque, direct-axis current, quadrature-axis current, changeless magnet surface temperature, stator burden temperature, and stator tooth temperature are taken as the input, whereas the stator winding temperature is taken as the yield. A profound neural organize (DNN) show for PMSM temperature forecast was developed. The exploratory comes about appeared the forecast blunder (MAE) was 0.1515, the RMSE was 0.2368, the goodness of fit (R^2) was 0.9439 and the goodness of fit [3].

Temperature expectation based on twofold input of three-dimensional shaping machine control system. Temperature criticism slack of three-dimensional shaping machine leads the protest adhere to the hot bed and shaping fabric stream issue in progress, the three-dimensional shaping machine control framework was analyzed, and put forward a kind of input control calculation of double temperature forecast based on slightest squares bolster vector machine (LSSVM) [4].

The essential thought of the present proposed warm circuit is to utilize as it were two warm resistances to show the electric motor: one proportionate to the full conduction and other for convection. It is vital to know a few previous data approximately a known stack condition and the comparing temperature rise within the winding. In this way, the

number of input parameters is radically diminished and overcomes the challenges related to geometric and physical parameter determination. The validation of the warm circuit was done with the help of tests of two engines in numerous stack conditions. [5].

The proposed method may be a third-order warm show which permits the expectation of three temperature spots. It comprises of warm resistances, warm capacitances, and warm sources. The warm resistances are related to the warm exchange between two spots and depend on the warm conductance of the materials and the length and zone of the components. The warm capacitances are related to the warm amassed and depend on the warm capacity of the components and their volumes. The particular meaning and values of the demonstrate parameters depend on the particular chosen spots. In our case, the chosen spots are the rotor bars, the stator winding, and the engine frame. The warm resistances are gotten from the steady-state operation. For this condition, the warm capacitances gotten to be open circuited, and the circuit show is diminished to the straightforward circuit [6].

This paper proposes a Standard IGBTs in a multilayer structure. The temperature changes of diverse materials with jumbled Coefficients of Warm Development (CTE) may cause a detachment within the contact regions and lead to disappointment of the IGBTs such as bond wire lift-off and patch layer weakness. This paper has displayed a strategy to foresee the lifetime of IGBTs amid a long-term driving cycle by considering numerous impact variables, such as the encompassing temperature, the driving cycle, and the debasement of warm resistance. At that point the impact components on lifetime forecast are talked about in detail to progress the precision of lifetime expectation [7]

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

The existing system is an application that motor temperature prediction model mainly uses the finite element method. The specific method is to simulate the transient temperature by using the finite element method, establish the temperature field, and then predict the motor temperature. This method can only be used to calculate and process the current linear data, but it cannot deal with a large number of non-linear historical data. On the other hand, the method based on machine learning can effectively solve this problem, and the overall prediction effect can be greatly improved compared with the traditional methods. Existing System used to support vector machine to predict the hot spot temperature of an oil-immersed transformer, and verified its practicability and effectiveness on a large power transformer.

3.2 Proposed System

The electric motor temperature dataset with 12 independent variables and 1 dependent variable with 9,98,8071 observations has been

used for implementation. The prediction of electric motor temperature is done with the following contributions. Fig. 1. Shows the overall workflow of the system. (i) Firstly, the data set is preprocessed with Feature Scaling and missing values. (ii) Secondly, empirical feature examination is done and the relation of motor speed and ambient temperature of the motor is visualized. (iii) Thirdly, the fresh data set is fitted to all the regressors and the execution is dismembered before and after scaling. (iv) Fourth, the raw data set is subjected to Convolutional neural network Conv1D with various activation layers like Relu, Sigmoid, softmax, Softplus, Softsign, Tanh, Selu, Elu and exponential layers. The performance is analyzed with EVS, MAE, MSE, RScore and Step loss of the Convolutional neural network.

Advantages

Easy to implement using machine learning.

It is optimizing the motor performance

It maximize the life of motor. Avoid damages and malfunction of devices.

3.3 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

3.3.1 Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.3.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.3.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user.

3.4 System Specification

The proposed system must contain an installed Anaconda Navigator .profound learning is aCPU serious programs queue thing to be running.

3.4.1 Hardware Specification

Quad center Intel Core i7 SkyLake or higher. M.2 PCIe or customary PCIe SSD with in any event 256GB of capacity, in spite of the fact that 512GB is best for execution. The quicker you can stack whats more spare your applications.The better the framework will perform.

3.4.2 Software Specification

All you need is and Python incorporating IDE. The one which is user amicable and easy to utilize is Spyder3.

3.4.3 Standards and Policies

Standards of the project ensure that quality is maintained throughout the project and to effectively manage all the documentation created throughout the project. The project is maintained and done according to the standards of the system.Policies of the project are managed effectively within the scope. quality, Resources (time and risk limitations. Appropriate governance and supervision are established During the life of a project, communication, quality, and risk management plans are developed and executed. Appropriate authorization and ac shall be established during a project's lifetime Acceptance.

Chapter 4

MODULE DESCRIPTION

4.1 General Architecture

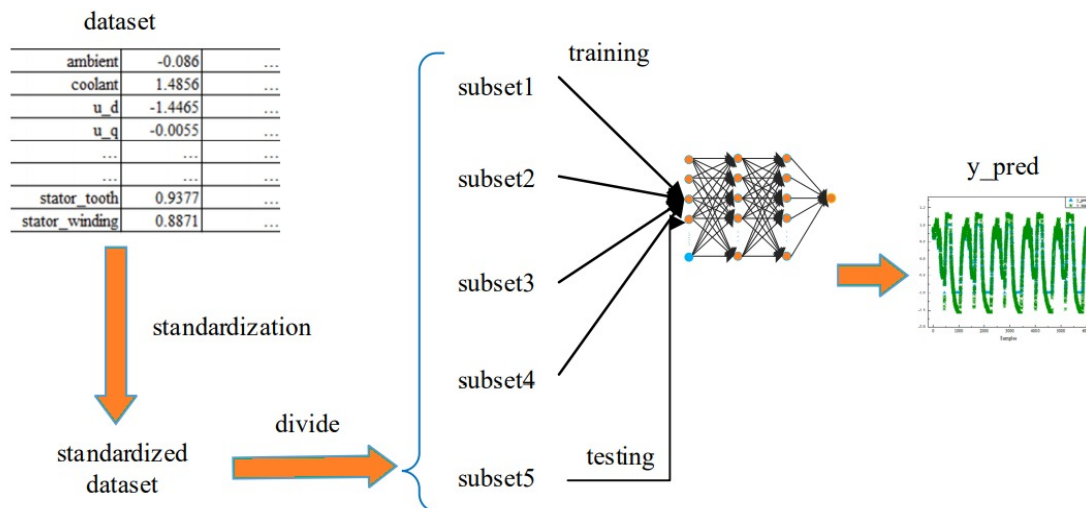


Figure 4.1: **Architecture Diagram**

Description The data set is preprocessed with Feature Scaling and Missing Values. the correlation of the features is done and the relation of each features are visualized , data set is subj raw data set is fitted to all the regressor s raw ected to multilayer perceptron with various activation layers 11

4.2 Design Phase

4.2.1 Data Flow Diagram

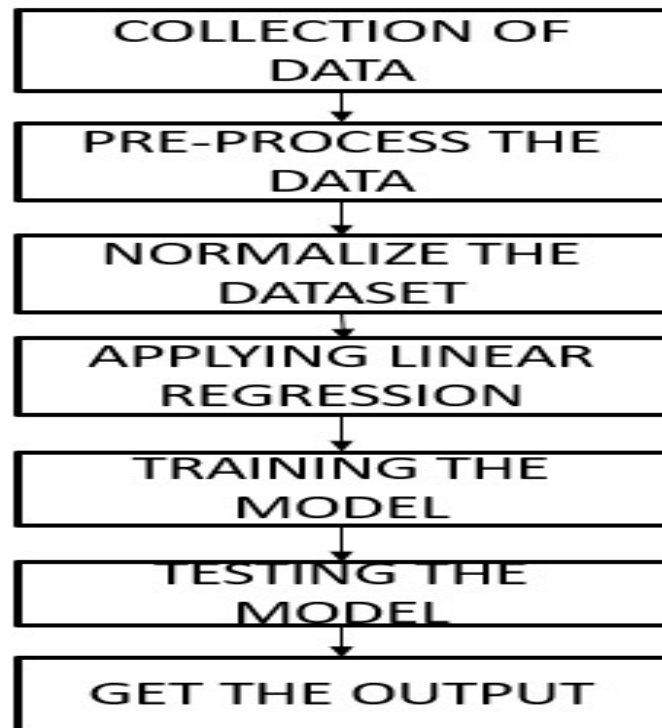


Figure 4.2: Data Flow Diagram

Description it is a conventional visual portrayal of the data streams inside a framework. It shows how information enters and leaves framework, what changes the data, and where information is put the away. The target of a DFD is to overall. show the extension and limits of a framework

4.2.2 UML Diagram

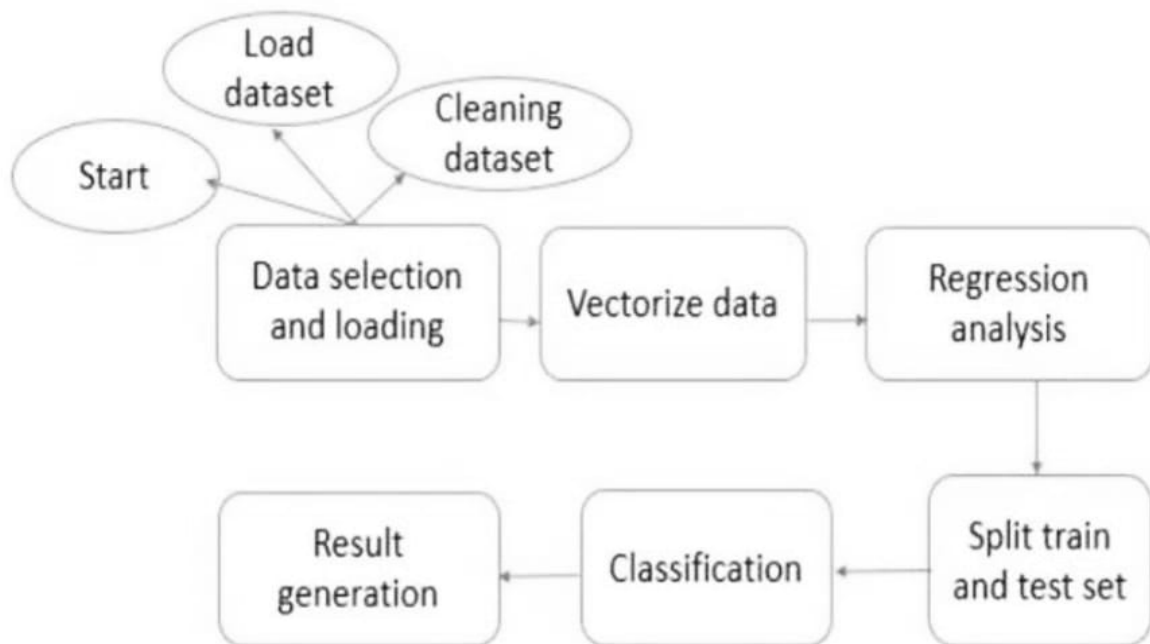


Figure 4.3: UML Diagram

Description Description it is a standard language for determining, picturing, veloping, and archiving the ancient rarities of programming frame works. It was at first begun to catch the conduct of complex pro gramming and non deprogramming framework and now it has become –an OMG standard. 13

4.2.3 Use Case Diagram

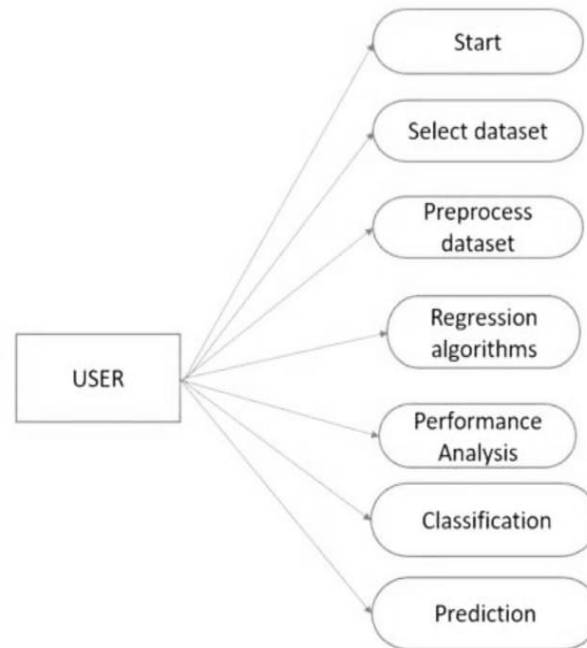


Figure 4.4: Use Case Diagram

Description Description Use case charts are an approach to catch the framework's usefulness and necessities in UML outlines. It catches the unique conduct of a live framework. A utilization case graph comprises of a utilization case a nd an entertainer. A utilization case addresses a particular usefulness of a framework, a segment, a bundle, or a class. 14

4.2.4 Class Diagram

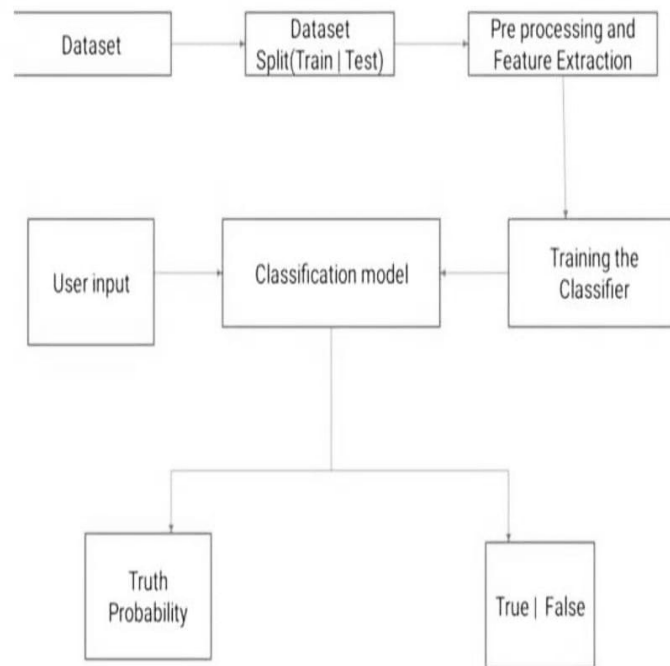


Figure 4.5: **Class Diagram**

Description Description Use case charts are an approach to catch the framework's usefulness and necessities in UML outlines. It catches the unique conduct of a live framework. A utilization case graph comprises of a utilization case a nd an entertainer. A utilization case addresses a particular usefulness of a framework, a segment, a bundle, or a class.

4.2.5 Sequence Diagram

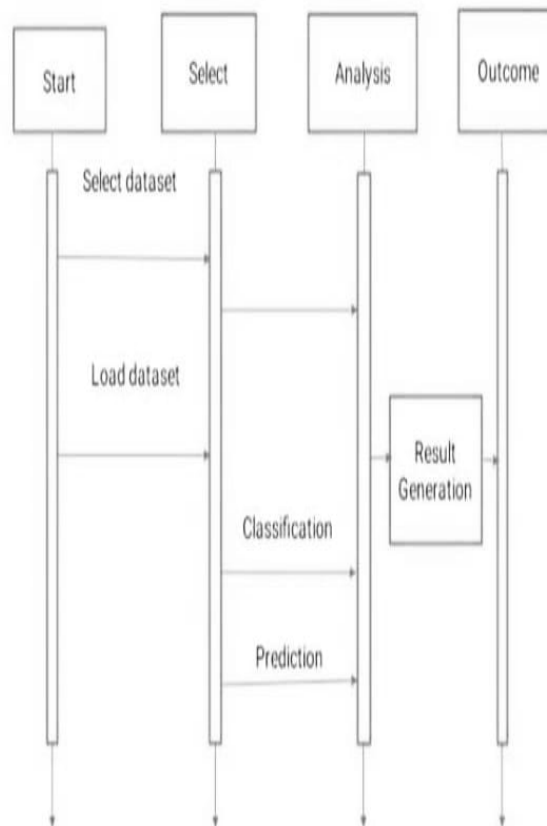


Figure 4.6: **Sequence Diagram**

Description Description Use case charts are an approach to catch the framework's usefulness and necessities in UML outlines. It catches the unique conduct of a live framework. A utilization case graph comprises of a utilization case and an entertainer. A utilization case addresses a particular usefulness of a framework, a segment, a bundle, or a class. 14

4.3 Module Description

ANACONDA NAVIGATOR (IDE) Anaconda Navigator is an distribution IDE for the languages like Python and R .It is used for com-

puting machine learning algorithms etc.

CODING TOOL Spyder Python is an open source platform for running and testing the programs. It has a very good remarkable interface to use.

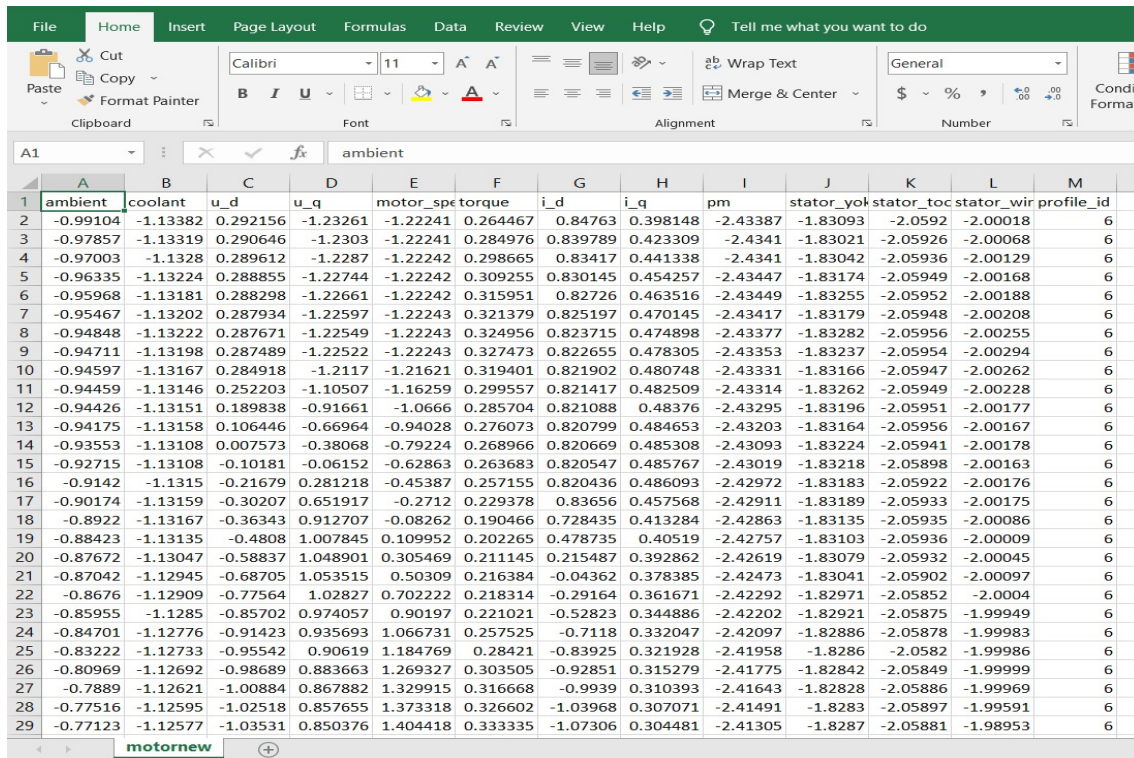
Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

The raw dataset is filled with various regression algorithms to predict the ambient temperature of electric motor .Based on the motor speed, temperature, voltage and current .The performance analysis is done for the raw dataset and system analyses the output based on the previous dataset.

5.1.1 Input Design



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	ambient	coolant	u_d	u_q	motor_spec	torque	i_d	i_q	pm	stator_yok	stator_toc	stator_wir	profile_id
2	-0.99104	-1.13382	0.292156	-1.23261	-1.22241	0.264467	0.84763	0.398148	-2.43387	-1.83093	-2.0592	-2.00018	6
3	-0.97857	-1.13319	0.290646	-1.2303	-1.22241	0.284976	0.839789	0.423309	-2.4341	-1.83021	-2.05926	-2.00068	6
4	-0.97003	-1.1328	0.289612	-1.2287	-1.22242	0.298665	0.83417	0.441338	-2.4341	-1.83042	-2.05936	-2.00129	6
5	-0.96335	-1.13224	0.288855	-1.22744	-1.22242	0.309255	0.830145	0.454257	-2.43447	-1.83174	-2.05949	-2.00168	6
6	-0.95968	-1.13181	0.288298	-1.22661	-1.22242	0.315951	0.82726	0.463516	-2.43449	-1.83255	-2.05952	-2.00188	6
7	-0.95467	-1.13202	0.287934	-1.22597	-1.22243	0.321379	0.825197	0.470145	-2.43417	-1.83179	-2.05948	-2.00208	6
8	-0.94848	-1.13222	0.287671	-1.22549	-1.22243	0.324956	0.823715	0.474898	-2.43377	-1.83282	-2.05956	-2.00255	6
9	-0.94711	-1.13198	0.287489	-1.22522	-1.22243	0.327473	0.822655	0.478305	-2.43353	-1.83237	-2.05954	-2.00294	6
10	-0.94597	-1.13167	0.284918	-1.2117	-1.21621	0.319401	0.821902	0.480748	-2.43331	-1.83166	-2.05947	-2.00262	6
11	-0.94459	-1.13146	0.252203	-1.10507	-1.16259	0.299557	0.821417	0.482509	-2.43314	-1.83262	-2.05949	-2.00228	6
12	-0.94426	-1.13151	0.189838	-0.91661	-1.0666	0.285704	0.821088	0.48376	-2.43295	-1.83196	-2.05951	-2.00177	6
13	-0.94175	-1.13158	0.106446	-0.66964	-0.94028	0.276073	0.820799	0.484653	-2.43203	-1.83164	-2.05956	-2.00167	6
14	-0.93553	-1.13108	0.007573	-0.38068	-0.79224	0.268966	0.820669	0.485308	-2.43093	-1.83224	-2.05941	-2.00178	6
15	-0.92715	-1.13108	-0.10181	-0.06152	-0.62863	0.263683	0.820547	0.485767	-2.43019	-1.83218	-2.05898	-2.00163	6
16	-0.9142	-1.1315	-0.21679	0.281218	-0.45387	0.257155	0.820436	0.486093	-2.42972	-1.83183	-2.05922	-2.00176	6
17	-0.90174	-1.13159	-0.30207	0.651917	-0.2712	0.229378	0.83656	0.457568	-2.42911	-1.83189	-2.05933	-2.00175	6
18	-0.8922	-1.13167	-0.36343	0.912707	-0.08262	0.190466	0.728435	0.413284	-2.42863	-1.83135	-2.05935	-2.00086	6
19	-0.88423	-1.13135	-0.4808	1.007845	0.109952	0.202265	0.478735	0.40519	-2.42757	-1.83103	-2.05936	-2.00009	6
20	-0.87672	-1.13047	-0.58837	1.048901	0.305469	0.211145	0.215487	0.392862	-2.42619	-1.83079	-2.05932	-2.00045	6
21	-0.87042	-1.12945	-0.68705	1.053515	0.50309	0.216384	-0.04362	0.378385	-2.42473	-1.83041	-2.05902	-2.00097	6
22	-0.8676	-1.12909	-0.77564	1.02827	0.702222	0.218314	-0.29164	0.361671	-2.42292	-1.82971	-2.05852	-2.0004	6
23	-0.85955	-1.1285	-0.85702	0.974057	0.90197	0.221021	-0.52823	0.344886	-2.42202	-1.82921	-2.05875	-1.99949	6
24	-0.84701	-1.12776	-0.91423	0.935693	1.066731	0.257525	-0.7118	0.332047	-2.42097	-1.82886	-2.05878	-1.99983	6
25	-0.83222	-1.12733	-0.95542	0.90619	1.184769	0.28421	-0.83925	0.321928	-2.41958	-1.8286	-2.0582	-1.99986	6
26	-0.80969	-1.12692	-0.98689	0.883663	1.269327	0.303505	-0.92851	0.315279	-2.41775	-1.82842	-2.05849	-1.99999	6
27	-0.7889	-1.12621	-1.00884	0.867882	1.329915	0.316668	-0.9939	0.310393	-2.41643	-1.82828	-2.05886	-1.99969	6
28	-0.77516	-1.12595	-1.02518	0.857655	1.373318	0.326602	-1.03968	0.307071	-2.41491	-1.8283	-2.05897	-1.99591	6
29	-0.77123	-1.12577	-1.03531	0.850376	1.404418	0.333335	-1.07306	0.304481	-2.41305	-1.8287	-2.05881	-1.98953	6

Figure 5.1: Input

5.1.2 Output Design

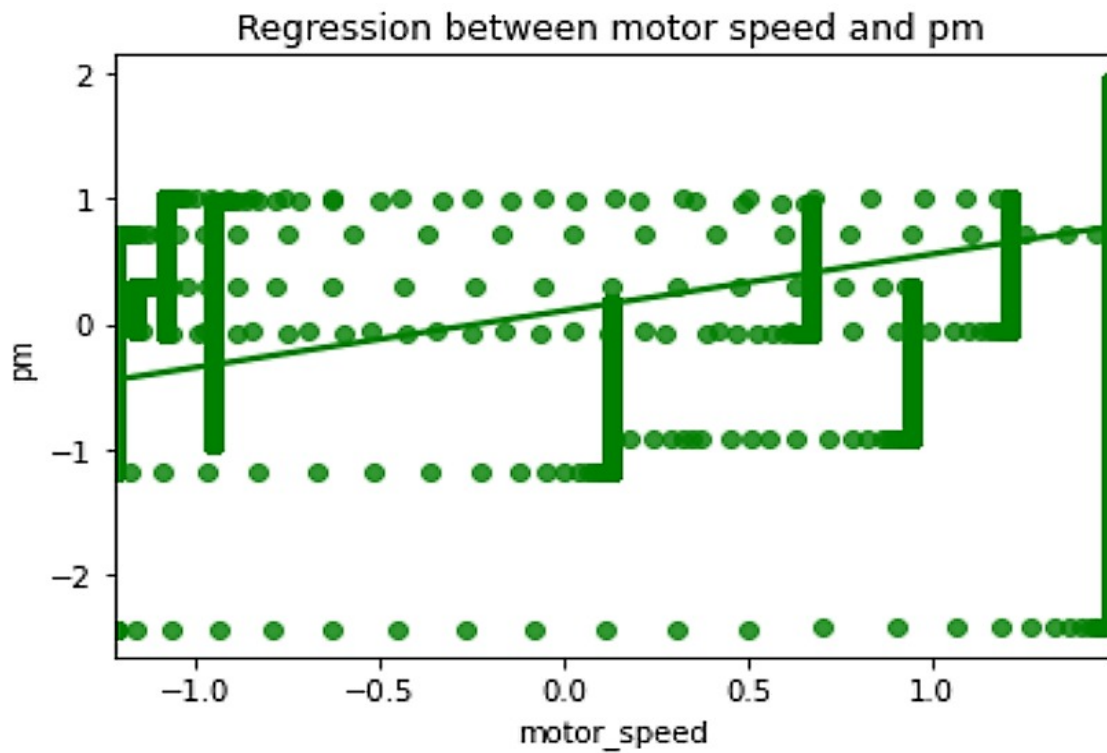


Figure 5.2: **Output**

5.2 Testing

It is the way toward checking a framework fully intent on distinguishing any mistakes, holes or missing necessity versus the genuine prerequisite.

5.3 Types of Testing

5.3.1 Unit testing

This project features a very vast scope in future. this will be implemented on intranet in future. Project are often updated in near future and when requirement for an equivalent arises, because it is extremely flexible in terms of the expansion.

5.3.2 Integration testing

Integration testing may be a level of software testing where the individual units are combined and tested as group. The purpose of this level of testing is to show faults within the interaction between the integrated units. Test drivers and test stubs are wont to assist in integration testing.

5.3.3 Functional testing

Functional testing is a type of software testing where the system is being tested against the functional specifications. Functions are tested by feeding them input and examining the output. Functional testing ensures that the wants are properly satisfied by the appliance

5.3.4 White Box Testing

White Box Testing could also be a way of software testing that tests internal structures or workings of an application, as against its functionality. In white box testing an indoor perspective of the system, also as programming skills, are wont to design test cases.

5.3.5 Black Box Testing

Black Box Testing could also be a way of software testing that examines the functionality of an application without peering into its internal structures or workings.

5.3.6 Test Result

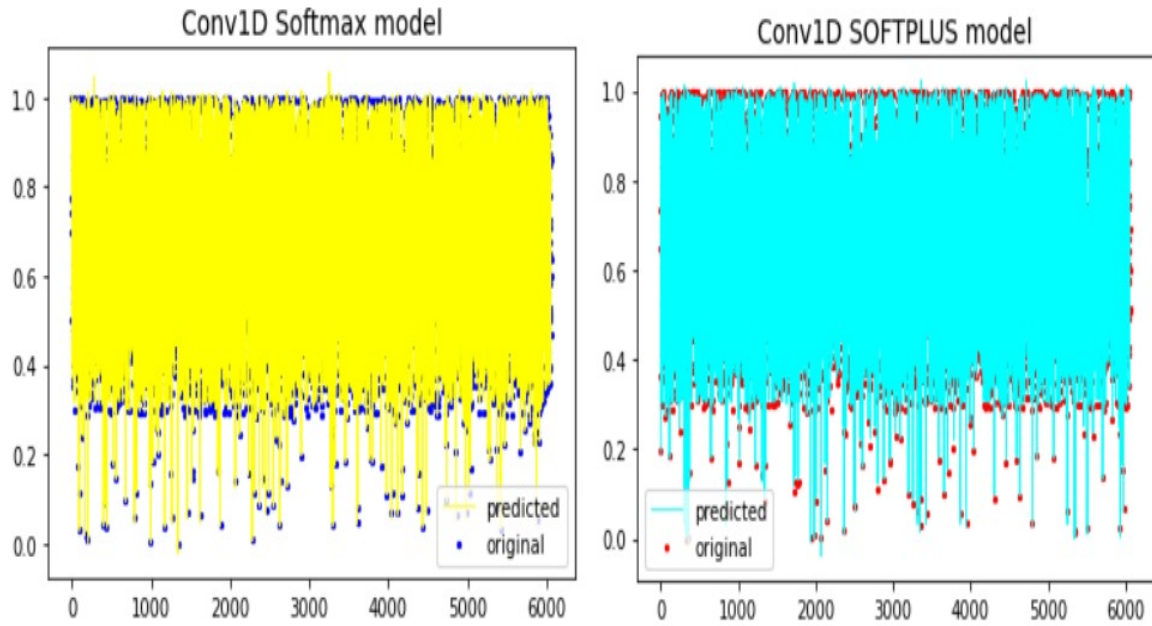


Fig.7. Performance of testing VS predicted target for Softmax and Softplus CNN Activation

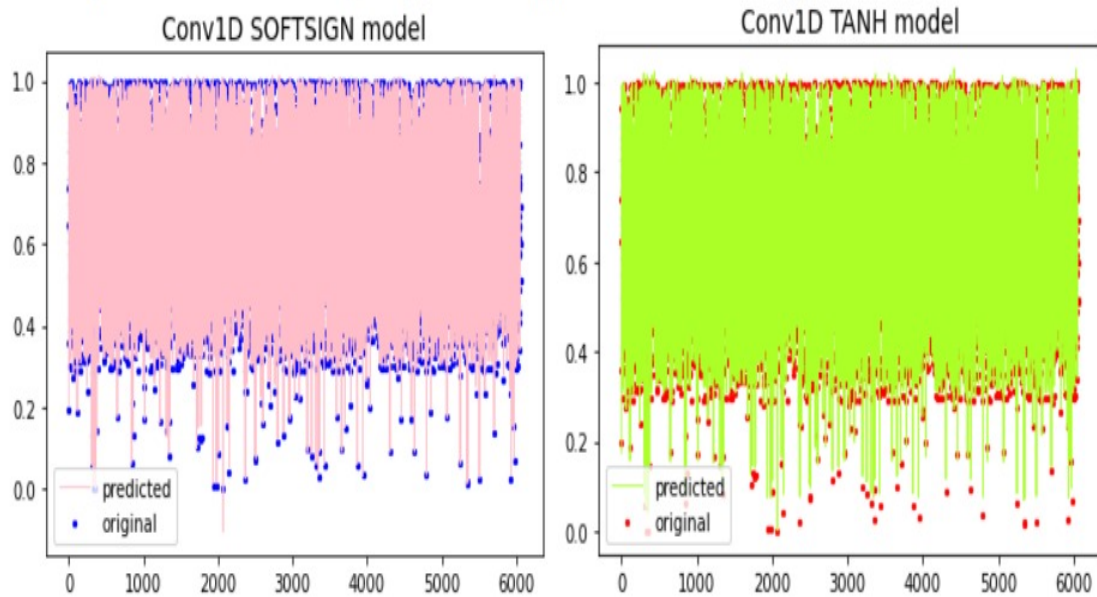


Figure 5.3: **Test Image**

5.4 Testing Strategy

The designer should lead the fruitful specialized audits to play out the testing successful. Testing begins with the segment blending of the whole PC based system.

level and work from outside toward the Different testing procedures are reasonable at various point on schedule. Testing is coordinated by the designer of the product and by a free experimental group. Investigating and testing are various exercises, at that point additionally the troubleshooting ought to be obliged in any procedure of testing.

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The proposed system's efficiency is 80 percent compared to the existing system. The prediction of these through the existing system is not more accurate and precise. Regression technique in the proposed system makes it easier to analyze the report of Ambient Temperature based on the details based on the motor speed, temperature, voltage and current , the system analysis the previous dataset and produces the output.

6.2 Comparison of Existing and Proposed System

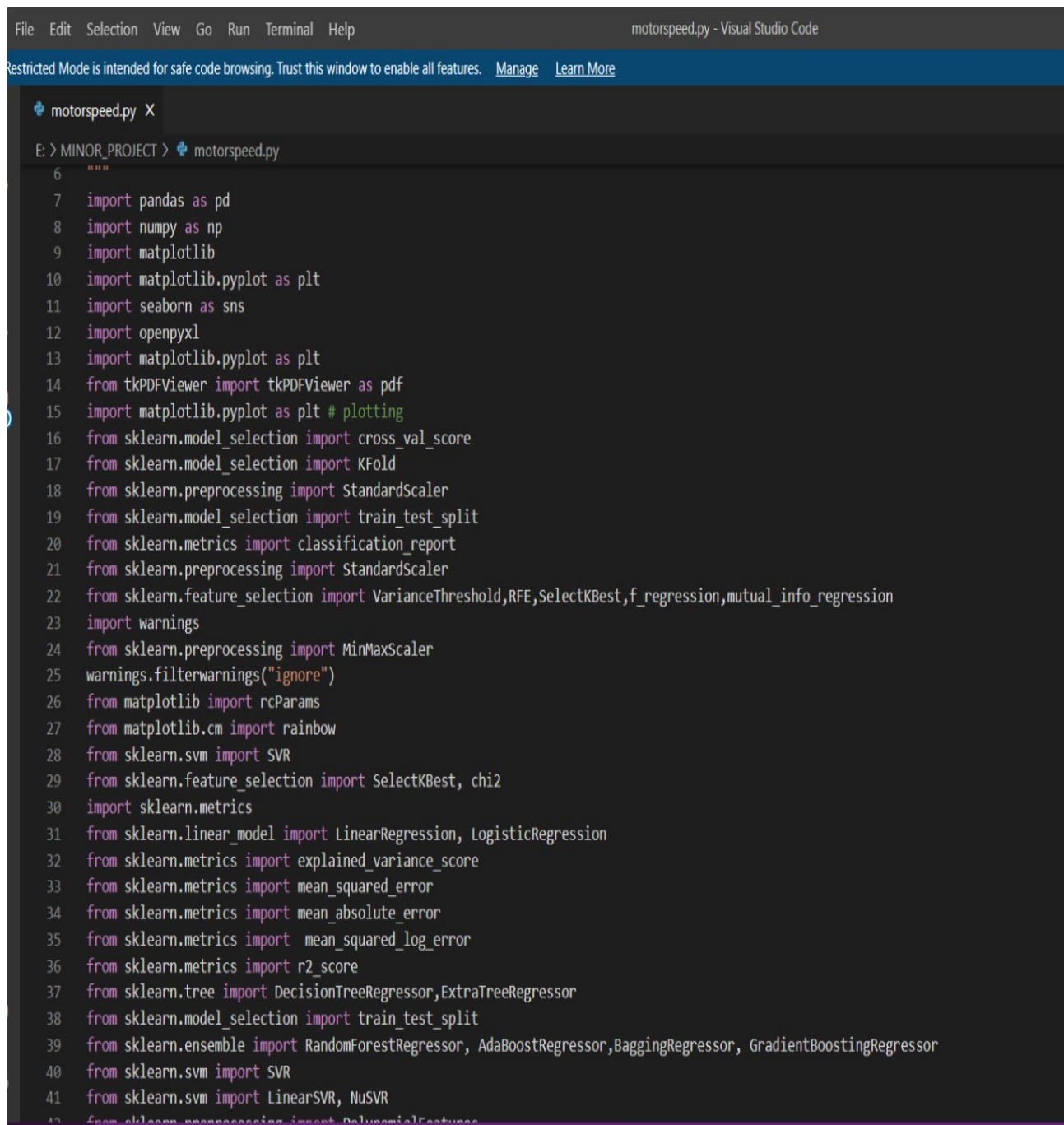
Existing System	Proposed System
<ul style="list-style-type: none">• Can fit just restricted information. Partiality of data.• Information alteringIncreasingly unpredictable• Less predictors• Poor execution• Chance of prediction is less	<ul style="list-style-type: none">• Boundless data.• Informational very simple.• Validation can be incorporated• More predictors• Takes a couple of second to create a these.• Prediction is more

Figure 6.1: Comparision

6.3 Advantages of the Proposed System

Easy to implement and use. Low cost and safe method.

6.4 Sample Code



```
6  """
7  import pandas as pd
8  import numpy as np
9  import matplotlib
10 import matplotlib.pyplot as plt
11 import seaborn as sns
12 import openpyxl
13 import matplotlib.pyplot as plt
14 from tkPDFViewer import tkPDFViewer as pdf
15 import matplotlib.pyplot as plt # plotting
16 from sklearn.model_selection import cross_val_score
17 from sklearn.model_selection import KFold
18 from sklearn.preprocessing import StandardScaler
19 from sklearn.model_selection import train_test_split
20 from sklearn.metrics import classification_report
21 from sklearn.preprocessing import StandardScaler
22 from sklearn.feature_selection import VarianceThreshold, RFE, SelectKBest, f_regression, mutual_info_regression
23 import warnings
24 from sklearn.preprocessing import MinMaxScaler
25 warnings.filterwarnings("ignore")
26 from matplotlib import rcParams
27 from matplotlib.cm import rainbow
28 from sklearn.svm import SVR
29 from sklearn.feature_selection import SelectKBest, chi2
30 import sklearn.metrics
31 from sklearn.linear_model import LinearRegression, LogisticRegression
32 from sklearn.metrics import explained_variance_score
33 from sklearn.metrics import mean_squared_error
34 from sklearn.metrics import mean_absolute_error
35 from sklearn.metrics import mean_squared_log_error
36 from sklearn.metrics import r2_score
37 from sklearn.tree import DecisionTreeRegressor, ExtraTreeRegressor
38 from sklearn.model_selection import train_test_split
39 from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, BaggingRegressor, GradientBoostingRegressor
40 from sklearn.svm import SVR
41 from sklearn.svm import LinearSVR, NuSVR
42 from sklearn.preprocessing import PolynomialFeatures
```

Figure 6.2: Sample Code

Output

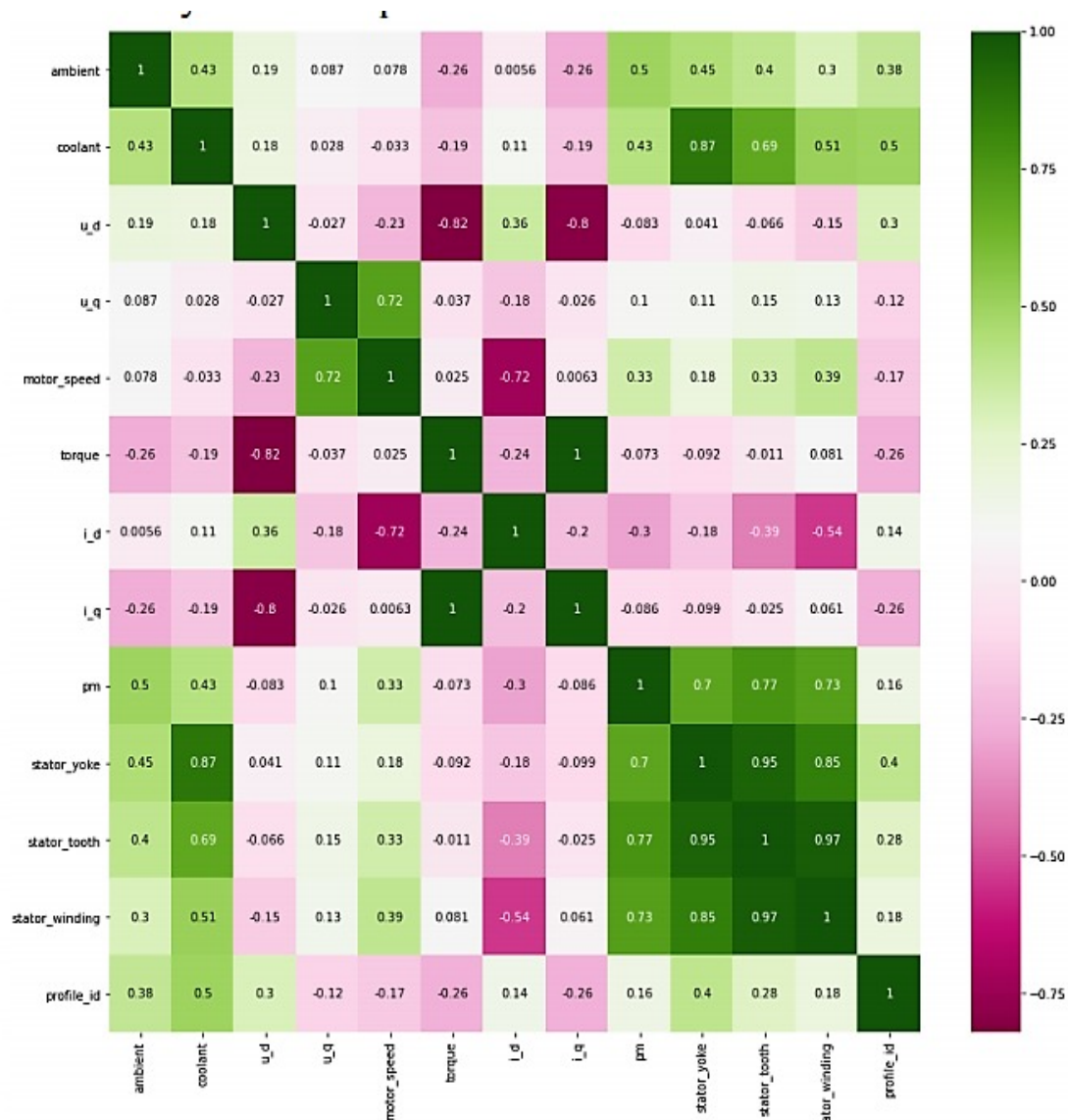


Figure 6.3: Output 1



Figure 6.4: **R2 Score before and after Scaling**

Regressor	EVS	MSE	MAE	RScore	Running Time (ms)
Linear	0.927315	0.002991	0.040319	0.927278	0.014054
Ridge	0.923947	0.003129	0.041637	0.923918	0.007322
ElasticNet	2.22E-16	0.041131	0.168421	-0.0002	0.008195
Lars	0.835437	0.00677	0.057383	0.835377	0.009842
LarsCV	0.888621	0.004582	0.051742	0.888568	0.120207
Lasso	2.22E-16	0.041131	0.168421	-0.0002	0.007112
LASSoLarsCV	0.927315	0.002991	0.040319	0.927278	0.128553
Bayesian	0.927313	0.002991	0.04032	0.927276	0.013427
ARd	0.927314	0.002991	0.040321	0.927277	0.022931
DecisionTree	0.999488	2.11E-05	0.000913	0.999488	0.454782
ExtraTree	0.999076	3.80E-05	0.001028	0.999076	0.094728
AdaBoost	0.96048	0.00171	0.035508	0.958426	9.517552
GradientBoost	0.995243	0.000196	0.010097	0.995237	12.69882
RandomForest	0.796481	0.008372	0.064959	0.796411	5.407051

Figure 6.5: **Regressor performance of the raw dataset before scaling**

Regressor	EVS	MSE	MAE	RScore	Running Time (ms)
Linear	0.927315	0.056735	0.175612	0.927278	0.008356
Ridge	0.927192	0.05683	0.17589	0.927156	0.007014
ElasticNet	0.328568	0.523929	0.596371	0.328437	0.00884
Lars	0.835437	0.128433	0.249937	0.835377	0.010961
LarsCV	0.891132	0.08497	0.222619	0.891086	0.114237
Lasso	0	0.780317	0.733576	-0.0002	0.008824
LASSoLarsCV	0.927315	0.056735	0.175612	0.927278	0.132557
Bayesian	0.927313	0.056737	0.175619	0.927276	0.013464
ARd	0.927314	0.056736	0.175622	0.927277	0.025343
DecisionTree	0.999537	0.000361	0.003925	0.999537	0.481415
ExtraTree	0.999872	0.0001	0.004148	0.999872	0.108875
AdaBoost	0.960786	0.031864	0.149446	0.959157	7.702117
GradientBoost	0.995162	0.003779	0.044563	0.995156	10.12313
RandomForest	0.796481	0.158832	0.282938	0.796411	4.293062

Figure 6.6: **Regressor performance of the raw dataset after scaling**

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

An attempt is made to find the performance analysis of the electric motor temperature dataset in forecasting the ambient temperature of the electric motor by applying various activation layers with convolutional neural network sequential model. The empirical feature examination is done and the relation of motor speed and ambient temperature of the motor is visualized. The correlation of each features in the dataset is extricated and the distribution of target variable with respect to other features are analyzed. Experimental results shows that the Conv1D-Softsign activation layer tends to reach the RScore of 99.842 with the step loss of 0.0001155

7.2 Future Enhancements

The raw data set is subjected to feature selection to find the important features with anova test analysis. The resampled dataset after anova is fitted to all the regressors with and without the presence of feature scaling and performance.

Chapter 8

PLAGIARISM REPORT

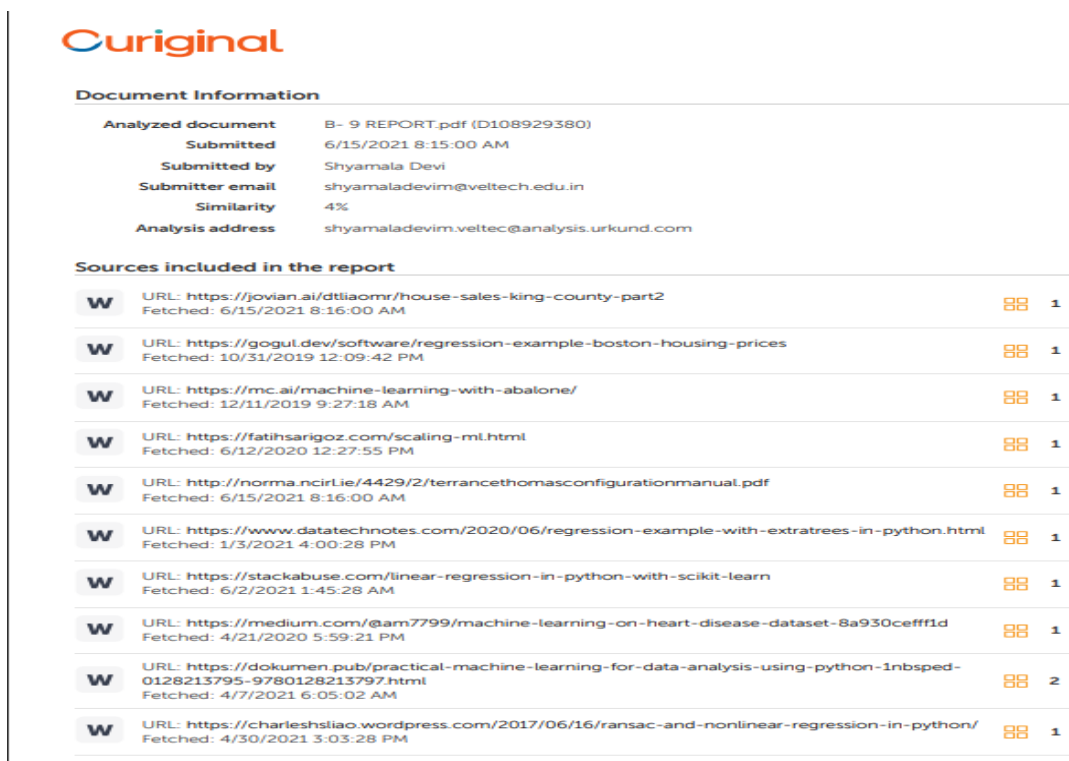


Figure 8.1: Poster Presentation

Chapter 9

SOURCE CODE & POSTER PRESENTATION

9.1 Source code

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import openpyxl
import matplotlib.pyplot as plt
from tkPDFViewer import tkPDFViewer as pdf
import matplotlib.pyplot as plt plotting
from sklearn.model selection import cross val score
from sklearn.model selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model selection import train test split
from sklearn.metrics import classification report
from sklearn.preprocessing import StandardScaler
```

```

from sklearn.feature_selection import VarianceThreshold, RFE, SelectKBest, f_regression, mutual_info_regression

import warnings
from sklearn.preprocessing import MinMaxScaler

warnings.filterwarnings("ignore")

from matplotlib import rcParams

from matplotlib.cm import rainbow

from sklearn.svm import SVR

from sklearn.feature_selection import SelectKBest, chi2

import sklearn.metrics

from sklearn.linear_model import LinearRegression, LogisticRegression

from sklearn.metrics import explained_variance_score

from sklearn.metrics import mean_squared_error

from sklearn.metrics import mean_absolute_error

from sklearn.metrics import mean_squared_log_error

from sklearn.metrics import r2_score

from sklearn.tree import DecisionTreeRegressor, ExtraTreeRegressor

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, BaggingRegressor, GradientBoostingRegressor

from sklearn.svm import SVR

from sklearn.svm import LinearSVR, NuSVR

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear_model import BayesianRidge, ARDRegression

from sklearn.linear_model import Ridge, RidgeCV, SGDRegressor, ElasticNet, ElasticNetCV, Lars, LarsCV, Lasso, LassoCV, LassoLars, LassoLarsCV, LassoLarsIC

from sklearn.preprocessing import StandardScaler

```

```

from sklearn.cross_decomposition import PLSRegression
import matplotlib.pyplot as plt
import time
from matplotlib.colors import Normalize
import matplotlib.cm as cm
import scipy.stats as stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, load_model
import keras
from keras.models import Sequential
from tensorflow.keras.models import Sequential
from keras.layers import Dense, Conv1D, Flatten
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

```

```

    dataset1 = pd.readcsv('motornew.csv')
df=dataset1
df.info()
profid = 6

```

```

    currdf = df[df['profile id'] == prof id]

```



```

currdf = curr df.drop('profile id', axis = 1)
columns = curr df.columns.tolist()
scaler = MinMaxScaler()
scurr df = pd.DataFrame(scaler.fittransform(currdf), columns= columns)
scurr df.head()
plt.title('Regression between motor speed and pm')
sns.regplot(x=currdf['motor speed'],y=currdf['pm'], color='g')
plt.show()
plt.figure(figsize=(8, 8))
plt.figure(figsize=(15,15))
sns.heatmap(df.corr(),cmap='PiYG',annot=True)

```

Before Feature Scaling

```

X = currdf.drop(['pm'], axis = 1)
y = currdf['pm']
Xtrain, Xtest, ytrain, ytest = traintestsplit(X,y, testsize = 0.2, randomstate = 0)
from sklearn.linear model import LinearRegression
regressor = LinearRegression()
regressor.fit(X train, y train)

```

predicting the test set result

```

ypredlogreg = regressor.predict(Xtest)

```

from sklearn.metrics import explained variance score

```

from sklearn.metrics import max error

```

```

from sklearn.metrics import mean absolute error
from sklearn.metrics import mean squared error
from sklearn.metrics import median absolute error
from sklearn.metrics import r2 score
EVC=explainedvariance score(ytest, ypredlogreg)
ME=maxerror(ytest, ypredlogreg)
MAE=meanabsoluteerror(ytest, ypredlogreg)
MSE=meansquarederror(ytest, ypredlogreg)
MDAE=medianabsoluteerror(ytest, ypredlogreg)
RS=r2score(ytest, ypredlogreg)
dfreg=pd.DataFrame(columns = ['Regressor','EVC','ME', 'MAE','MSE','MDAE','RS'])
L2=['LinearRegressor','explained variance score','max error', 'mean absolute er-
ror','mean
squared error','median absolute error','r2 score']
dfreg.loc[len(dfreg),:]=L2
L2=['LinearRegressor',EVC,ME,MAE,MSE,MDAE,RS]
dfreg.loc[len(dfreg),:]=L2

```

```

from sklearn.ensemble import AdaBoostRegressor
adaregressor = AdaBoostRegressor()
adaregressor.fit(Xtrain, ytrain)

```

```

    predicting the test set result
ypredadareg = adaregressor.predict(Xtest)

```

```

EVC=explainedvariance score(ytest, ypredadareg)

```

```

ME=maxerror(ytest, ypred adareg)
MAE=mean absoluteerror(ytest, ypredadareg)
MSE=mean squarederror(ytest, ypredadareg)
MDAE=medianabsoluteerror(ytest, ypredadareg)
RS=r2score(ytest, ypredadareg)
L2=["AdaBoostRegressor",EVC,ME,MAE,MSE,MDAE,RS]
dfreg.loc[len(dfreg),:]=L2

```

```

from sklearn.ensemble import BaggingRegressor
baregressor = BaggingRegressor()
baregressor.fit(Xtrain, ytrain)
predicting the test set result
ypred bareg = baregressor.predict(Xtest)

```

```

EVC=explainedvariancescore(ytest, ypredbareg)
ME=max error(ytest, ypredbareg)
MAE=meanabsoluteerror(ytest, ypredbareg)
MSE=meansquarederror(ytest, ypredbareg)
MDAE=medianabsoluteerror(ytest, ypredbareg)
RS=r2score(ytest, ypredbareg)
L2=["BaggingRegressor",EVC,ME,MAE,MSE,MDAE,RS]
dfreg.loc[len(dfreg),:]=L2

```

```

from sklearn.ensemble import ExtraTreesRegressor
etregressor = ExtraTreesRegressor()
etregressor.fit(Xtrain, ytrain)

```

predicting the test set result

EVC=explained variance score(ytest, ypred etreg)

ME=maxerror(y test, predetreg)

MAE=mean absolute error(y test, y pred etreg)

MSE=mean squared error(y test, y pred etreg)

MDAE=median absolute error(y test, y pred etreg)

RS=r2 score(y test, y pred etreg)

L2=['ExtraTreesRegressor',EVC,ME,MAE,MSE,MDAE,RS]

dfreg.loc[len(dfreg),:]=L2

from sklearn.ensemble import GradientBoostingRegressor

gbregressor = GradientBoostingRegressor()

gbregressor.fit(X train, y train)

predicting the test set result

y pred gbreg = gbregressor.predict(X test)

EVC=explained variance score(y test, y pred gbreg)

ME=max error(y test, y pred gbreg)

MAE=mean absolute error(y test, y pred gbreg)

MSE=mean squared error(y test, y pred gbreg)

MDAE=median absolute error(y test, y pred gbreg)

RS=r2 score(y test, y pred gbreg)

L2=['GradientBoostingRegressor',EVC,ME,MAE,MSE,MDAE,RS]

dfreg.loc[len(dfreg),:]=L2

```

from sklearn.ensemble import RandomForestRegressor
rfregressor = RandomForestRegressor()
rfregressor.fit(Xtrain, ytrain)
predicting the test set result
y_pred_rfreg = rfregressor.predict(Xtest)

```

```

EVC=explainedvariancescore(ytest, ypredrfreg)
ME=max error(y test, ypred rfreg)
MAE=mean absolute error(y test, y_pred_rfreg)
MSE=mean squared error(y test, y_pred_rfreg)
MDAE=median absolute error(y test, y_pred_rfreg)
RS=r2 score(y test, y_pred_rfreg)
L2=["RandomForestRegressor",EVC,ME,MAE,MSE,MDAE,RS]
dfreg.loc[len(dfreg),:]=L2

```

```

from sklearn.linear model import ElasticNet
from sklearn.datasets import make_regression
regr = ElasticNet(random state=0)
regr.fit(X train, y train)
y_pred_enreg = regr.predict(X test)

```

```

EVC=explained variance score(y test, y_pred_enreg)
ME=max error(y test, y_pred_enreg)
MAE=mean absolute error(y test, y_pred_enreg)
MSE=mean squared error(y test, y_pred_enreg)

```

```
MDAE=median absolute error(y test, y pred enreg)
RS=r2 score(y test, y pred enreg)
L2=["ElasticNet",EVC,ME,MAE,MSE,MDAE,RS]
dfreg.loc[len(dfreg),:]=L2
```

```
from sklearn.kernel ridge import KernelRidge
krregressor = KernelRidge()
krregressor.fit(X train, y train)
```

```
predicting the test set result
y pred krreg =krregressor.predict(X test)
```

```
EVC=explained variance score(y test, y pred krreg)
ME=max error(y test, y pred krreg)
MAE=mean absolute error(y test, y pred krreg)
MSE=mean squared error(y test, y pred krreg)
MDAE=median absolute error(y test, y pred krreg)
RS=r2 score(y test, y pred krreg)
L2=["KernelRidge",EVC,ME,MAE,MSE,MDAE,RS]
dfreg.loc[len(dfreg),:]=L2
```

```
from sklearn.svm import SVR
svr regressor = SVR(kernel='rbf', gamma='auto')
svr regressor.fit(X train, y train)
y pred svreg =svr regressor.predict(X test)
```

```

EVC=explained variance score(y test, y pred svreg)
ME=max error(y test, y pred svreg)
MAE=mean absolute error(y test, y pred svreg)
MSE=mean squared error(y test, y pred svreg)
MDAE=median absolute error(y test, y pred svreg)
RS=r2 score(y test, y pred svreg)
L2=["SVR",EVC,ME,MAE,MSE,MDAE,RS]
dfreg.loc[len(dfreg),:]=L2

```

```

from sklearn.tree import DecisionTreeRegressor
tree regressor = DecisionTreeRegressor(random state = 0)
tree regressor.fit(X train, y train)
y pred dtreg =tree regressor.predict(X test)

```

```

EVC=explained variance score(y test, y pred dtreg)
ME=max error(y test, y pred dtreg)
MAE=mean absolute error(y test, y pred dtreg)
MSE=mean squared error(y test, y pred dtreg)
MDAE=median absolute error(y test, y pred dtreg)
RS=r2 score(y test, y pred dtreg)
L2=["DecisionTreeRegressor",EVC,ME,MAE,MSE,MDAE,RS]
dfreg.loc[len(dfreg),:]=L2

```

WITH FEATURE SCALING

```

X = curr df.drop(['pm'], axis = 1)
y = curr df['pm']

```

```
sc X = StandardScaler()
```

```
X= sc X.fit transform(X)
```

```
X train, X test, y train, y test = train test split(X,y, test size = 0.2, random state = 0)
```

```
from sklearn.linear model import LinearRegression
```

```
regressor = LinearRegression()
```

```
regressor.fit(SX train, y train)
```

predicting the test set

```
y pred logreg = regressor.predict(SX test)
```

```
EVC=explained variance score(y test, y pred logreg)
```

```
ME=max error(y test, y pred logreg)
```

```
MAE=mean absolute error(y test, y pred logreg)
```

```
MSE=mean squared error(y test, y pred logreg)
```

```
MDAE=median absolute error(y test, y pred logreg)
```

```
RS=r2 score(y test, y pred logreg)
```

```
dfregfs=pd.DataFrame(columns = ['Regressor','EVC','ME', 'MAE','MSE','MDAE','RS'])
```

```
L2=['LinearRegressor','explained variance score','max error', 'mean absolute error', 'mean squared error', 'median absolute error', 'r2 score']
```

```
dfreg.loc[len(dfreg),:]=L2
```

```
L2=['LinearRegressor',EVC,ME,MAE,MSE,MDAE,RS]
```

```
dfregfs.loc[len(dfregfs),:]=L2
```

```
from sklearn.ensemble import AdaBoostRegressor
```



```
adaregressor = AdaBoostRegressor()
```

```
adaregressor.fit(SX train, y train)
```

predicting the test set result

```
y pred adareg = adaregressor.predict(SX test)
```

```
EVC=explained variance score(y test, y pred adareg)
```

```
ME=max error(y test, y pred adareg)
```

```
MAE=mean absolute error(y test, y pred adareg)
```

```
MSE=mean squared error(y test, y pred adareg)
```

```
MDAE=median absolute error(y test, y pred adareg)
```

```
RS=r2 score(y test, y pred adareg)
```

```
L2=[ 'AdaBoostRegressor',EVC,ME,MAE,MSE,MDAE,RS]
```

```
dfregfs.loc[len(dfregfs),:]=L2
```

```
from sklearn.ensemble import BaggingRegressor
```

```
baregressor = BaggingRegressor()
```

```
baregressor.fit(SX train, y train)
```

predicting the test set result

```
y pred bareg = baregressor.predict(SX test)
```

```
EVC=explained variance score(y test, y pred bareg)
```

```
ME=max error(y test, y pred bareg)
```

```
MAE=mean absolute error(y test, y pred bareg)
```

```
MSE=mean squared error(y test, y pred bareg)
```

```
MDAE=median absolute error(y test, y pred bareg)
RS=r2 score(y test, y pred bareg)
L2=["BaggingRegressor",EVC,ME,MAE,MSE,MDAE,RS]
dfregfs.loc[len(dfregfs),:]=L2
```

```
from sklearn.ensemble import ExtraTreesRegressor
etregressor = ExtraTreesRegressor()
etregressor.fit(SX train, y train)
```

```
predicting the test set result
y pred etreg = etregressor.predict(SX test)
```

```
EVC=explained variance score(y test, y pred etreg)
ME=max error(y test, y pred etreg)
MAE=mean absolute error(y test, y pred etreg)
MSE=mean squared error(y test, y pred etreg)
MDAE=median absolute error(y test, y pred etreg)
RS=r2 score(y test, y pred etreg)
L2=["ExtraTreesRegressor",EVC,ME,MAE,MSE,MDAE,RS]
dfregfs.loc[len(dfregfs),:]=L2
```

```
from sklearn.ensemble import GradientBoostingRegressor
gbregressor = GradientBoostingRegressor()
gbregressor.fit(SX train, y train)
```

```
predicting the test set result
```

```
y_pred_gbreg = gbregressor.predict(SX_test)
```

```
EVC=explained_variance_score(y_test, y_pred_gbreg)
```

```
ME=max_error(y_test, y_pred_gbreg)
```

```
MAE=mean_absolute_error(y_test, y_pred_gbreg)
```

```
MSE=mean_squared_error(y_test, y_pred_gbreg)
```

```
MDAE=median_absolute_error(y_test, y_pred_gbreg)
```

```
RS=r2_score(y_test, y_pred_gbreg)
```

```
L2=['GradientBoostingRegressor',EVC,ME,MAE,MSE,MDAE,RS]
```

```
dfregfs.loc[len(dfregfs),:]=L2
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
rfregressor = RandomForestRegressor()
```

```
rfregressor.fit(SX_train, y_train)
```

predicting the test set result

```
y_pred_rfreg = rfregressor.predict(SX_test)
```

```
EVC=explained_variance_score(y_test, y_pred_rfreg)
```

```
ME=max_error(y_test, y_pred_rfreg)
```

```
MAE=mean_absolute_error(y_test, y_pred_rfreg)
```

```
MSE=mean_squared_error(y_test, y_pred_rfreg)
```

```
MDAE=median_absolute_error(y_test, y_pred_rfreg)
```

```
RS=r2_score(y_test, y_pred_rfreg)
```

```
L2=['RandomForestRegressor',EVC,ME,MAE,MSE,MDAE,RS]
```

```
dfregfs.loc[len(dfregfs),:]=L2
```

```

from sklearn.linear model import ElasticNet
from sklearn.datasets import make regression
regr = ElasticNet(random state=0)
regr.fit(SX train, y train)
y pred enreg = regr.predict(SX test)

```

```

EVC=explained variance score(y test, y pred enreg)
ME=max error(y test, y pred enreg)
MAE=mean absolute error(y test, y pred enreg)
MSE=mean squared error(y test, y pred enreg)
MDAE=median absolute error(y test, y pred enreg)
RS=r2 score(y test, y pred enreg)
L2=["ElasticNet",EVC,ME,MAE,MSE,MDAE,RS]
dfregfs.loc[len(dfregfs),:]=L2

```

```

from sklearn.kernel ridge import KernelRidge
krregressor = KernelRidge()
krregressor.fit(SX train, y train)

```

```

predicting the test set result
y pred krreg =krregressor.predict(SX test)

```

```

EVC=explained variance score(y test, y pred krreg)
ME=max error(y test, y pred krreg)
MAE=mean absolute error(y test, y pred krreg)

```

```

MSE=mean squared error(y test, y pred krrreg)
MDAE=median absolute error(y test, y pred krrreg)
RS=r2 score(y test, y pred krrreg)
L2=["KernelRidge",EVC,ME,MAE,MSE,MDAE,RS]
dfregfs.loc[len(dfregfs),:]=L2

```

```

from sklearn.svm import SVR
svr regressor = SVR(kernel='rbf', gamma='auto')
svr regressor.fit(SX train, y train)
y pred svreg =svr regressor.predict(SX test)

```

```

EVC=explained variance score(y test, y pred svreg)
ME=max error(y test, y pred svreg)
MAE=mean absolute error(y test, y pred svreg)
MSE=mean squared error(y test, y pred svreg)
MDAE=median absolute error(y test, y pred svreg)
RS=r2 score(y test, y pred svreg)
L2=["SVR",EVC,ME,MAE,MSE,MDAE,RS]
dfregfs.loc[len(dfregfs),:]=L2

```

```

from sklearn.tree import DecisionTreeRegressor
tree regressor = DecisionTreeRegressor(random state = 0)
tree regressor.fit(SX train, y train)
y pred dtreg =tree regressor.predict(SX test)

```

```

EVC=explained variance score(y test, y pred dtreg)

```

ME=max error(y test, y pred dtreg)

MAE=mean absolute error(y test, y pred dtreg)

MSE=mean squared error(y test, y pred dtreg)

MDAE=median absolute error(y test, y pred dtreg)

RS=r2 score(y test, y pred dtreg)

L2=["DecisionTreeRegressor",EVC,ME,MAE,MSE,MDAE,RS]

dfregfs.loc[len(dfregfs),:]=L2

9.2 Poster Presentation

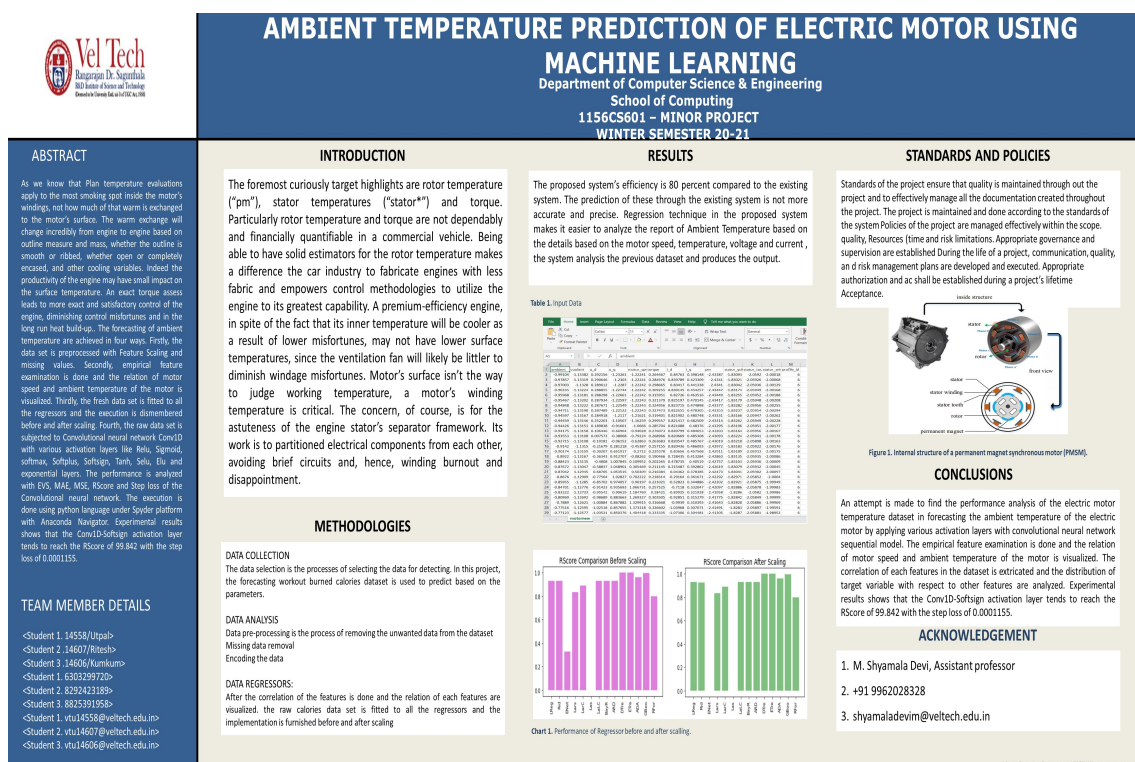


Figure 9.1: Poster Presentation

References

- [1] M. Anibal Valenzuela, and Pablo Reyes, “Simple and Reliable Model for the Thermal Protection of Variable-Speed Self-Ventilated Induction Motor Drives”, IEEE Transactions on Industry Applications, vol. 46, no. 2, 2016.

- [2] D. Mukherjee, S. Chakraborty, P. K. Guchhait and J. Bhunia, ”Application of Machine Learning for Speed and Torqu Prediction of PMS Motor in Electric Vehicles,” 2020 IEEE 1st International Conference for Convergence in Engineering (ICCE), Kolkata, India, 2020, pp. 129-133, doi: 10.1109/ICCE50343.2020.9290632.

- [3] Kaicheng Zhang, Akhil Guliani, Seda Ogrenci-Memik, Gokhan Memik, Kazutomo Yoshii, Rajesh Sankaran, Pete Beckman, “Machine Learning-Based Temperature Prediction for Runtime Thermal Management across System Components”, IEEE Transactions On Parallel And Distributed Systems, March 2016.

- [4] Cassiano Antunes Cezario, Hilton Penha Silva, “Electric Motor Winding Temperature Prediction Using a Simple Two-Resistance Thermal Circuit”, Proceedings of the 2008 International Conference on Electrical Machines, 2008.
- [5] Guo, Hai and Ding, Qun and Song, Yifan and Tang, Haoran and Wang, Likun and Zhao, Jingying, “Predicting Temperature of Permanent Magnet Synchronous Motor Based on Deep Neural Network”, Journal of Energies, vol 13, no. 18, 2020, <https://www.mdpi.com/1996-1073/13/18/4782> .
- [6] Yuanbin Hou, MeiE Gao, Chen Li, Hongyan Li, “ Temperature prediction based on double feedback of three-dimensional forming machine control system”, Proceedings of the International Symposium on Computer, Consumer and Control, 2016.
- [7] Ali Maknouninejad, Konrad Woronowicz and Alireza Safaee, “Enhanced Algorithm for Real Time Temperature Rise Prediction of A Traction Linear Induction Motor”, Journal of Institute of Energy Futures, Smart Power Networks, 2018..
- [8] B. Rojsca , S. Wilkins, J. Jacob, E.R.G. Hoedemaekersand S.P. van den Hoek, Modeling Of An Integrated Drive Unit In An Electric Vehicle”, International Journal of Engineering Research Technology, 2016.

- [9] Sunghyun Moon, and Sungho Lee, “High-Reliable Temperature Prediction Considering Stray Load Loss for Large Induction Machine”, IEEE Transactions On Magnetics, vol. 55, no. 6, 2019.