Electric Motor Temperature Prediction Using Machine Learning

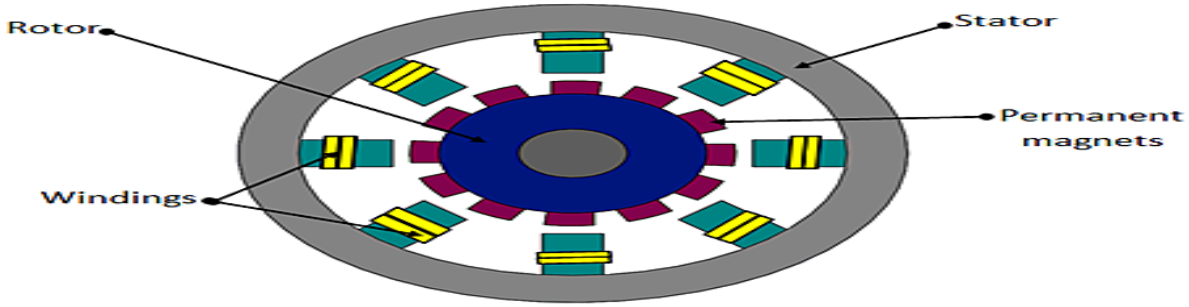
Project Description:

Introduction:

The permanent-magnet synchronous machine (PMSM) drive is one of the best choices for a full range of motion control applications. For example, the PMSM is widely used in robotics, machine tools, actuators, and it is being considered in high-power applications such as industrial drives and vehicular propulsion. It is also used for residential/commercial applications. The PMSM is known for having low torque ripple, superior dynamic performance, high efficiency, and high-power density.

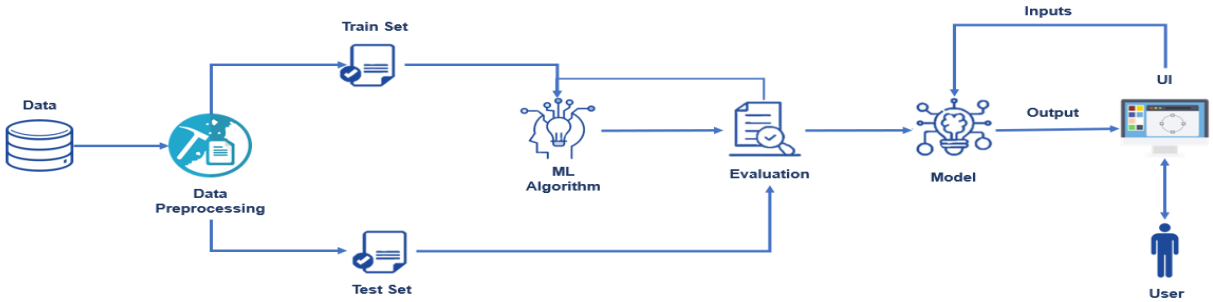
The task is to design a model with appropriate feature engineering that estimates the target temperature of a rotor.

In this project, we will be using algorithms such as Linear Regression, Decision Tree, Random Forest and SVM. We will train and test the data with these algorithms and select the best model. The best algorithm will be selected and saved in pkl format. We will be doing flask integration and IBM deployment.



[Text Wrapping Break]

Technical Architecture:[Text Wrapping Break]



[Text Wrapping Break]**Pre requisites:**

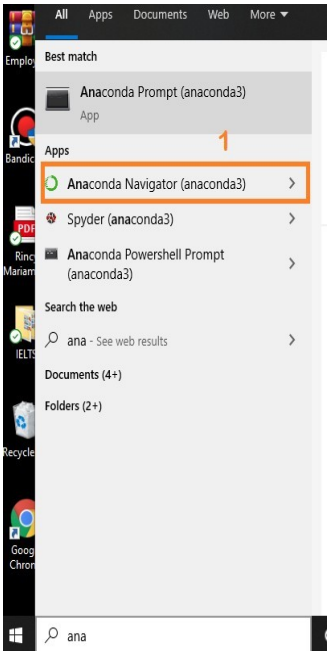
To complete this project, you must require following software's concepts and packages

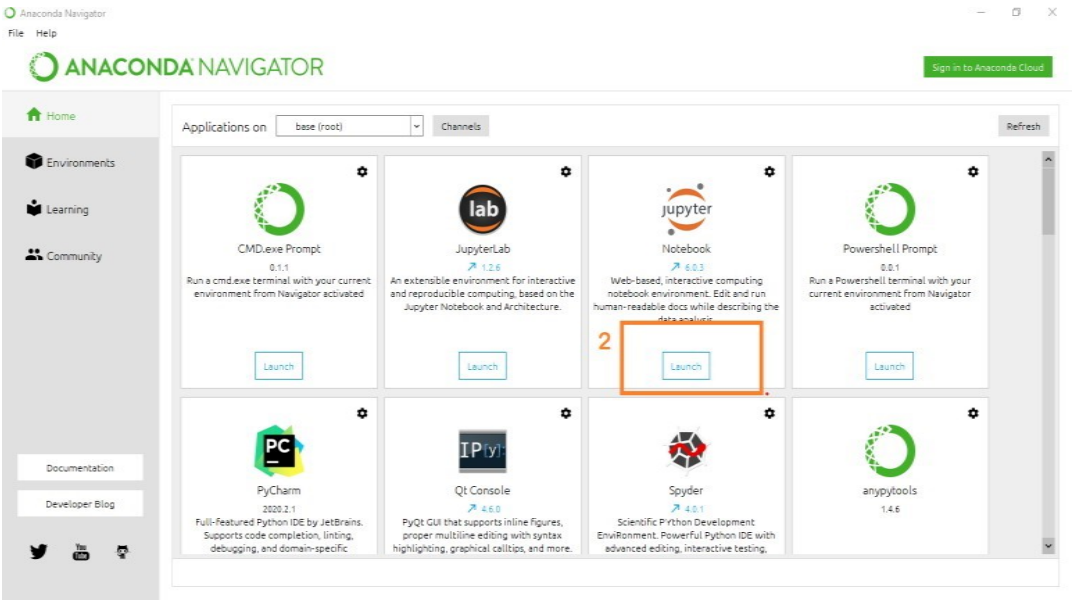
- **Anaconda navigator:**
- Refer to the link below to download anaconda navigator
- **Link : https://www.youtube.com/watch?v=5mDYijMfSzs**
- **Python packages:**

  Open anaconda prompt as administrator.
- Type "pip install numpy" and click enter.
- Type "pip install pandas" and click enter.
- Type "pip install matplotlib" and click enter.
- Type "pip install scikit-learn" and click enter.
- Type "pip install Flask" and click enter.

  The above steps allow you to install the packages in the anaconda environment

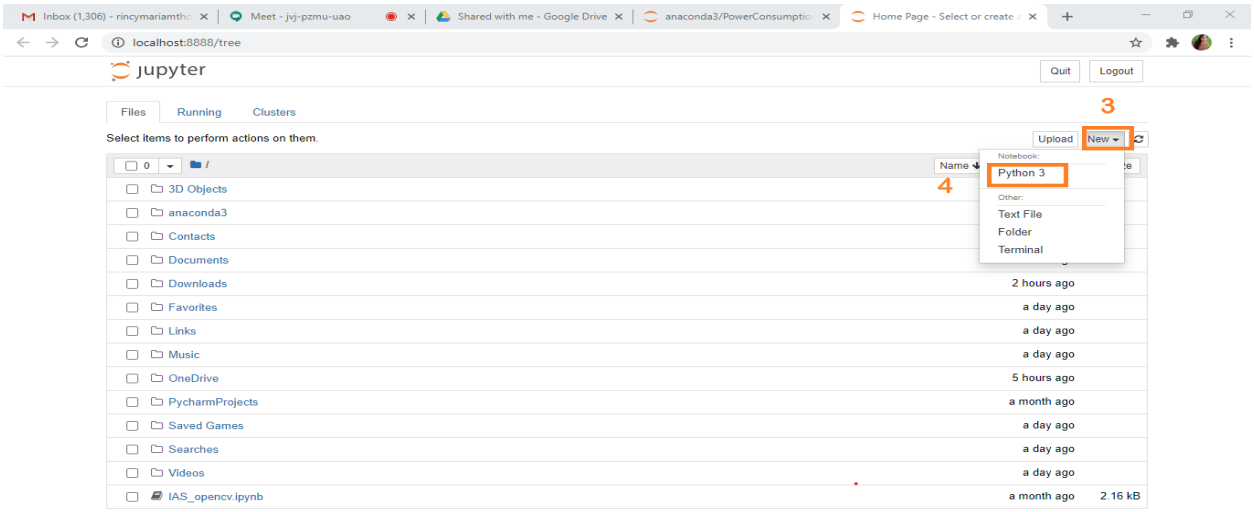
- **Launch Jupyter**





Search for Anaconda Navigator and open Launch Jupyter notebook.

- Then you will be able to see that the jupyter notebook runs on localhost:5000.
- To Create a new file Go to New →Python3.The file in jupyter notebook is saved with .ipynb extension.



- Flask Basics : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives:

By the end of this project:

- You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.

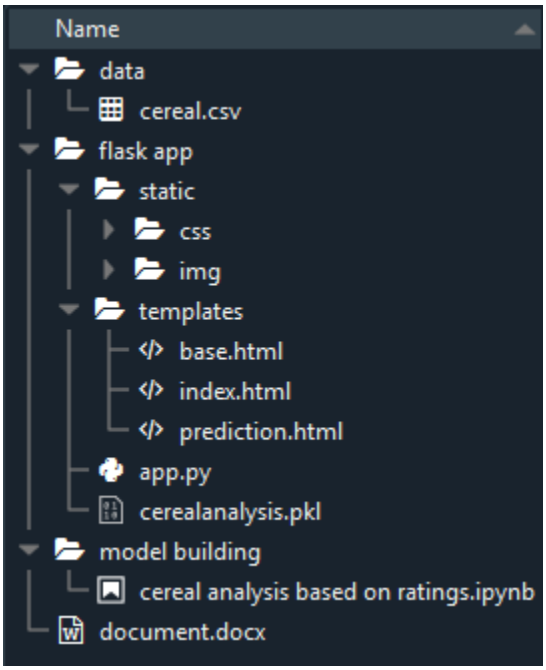
- You will be able to know how to pre-process / clean the data using different data preprocessing techniques.
- You will be able to analyse or get insights of data through visualization.
- Applying different algorithms according to dataset and based on visualization.
- You will be able to know how to find the accuracy of the model.
- You will be able to know how to build a web application using the Flask framework.

Project Flow:

- User interacts with the UI (User Interface) to enter the input values
- Entered input values are analyzed by the model which is integrated
- Once model analyses the input the prediction is showcased on the UI To accomplish this, we have to complete all the activities and tasks listed below
- Data Collection.
- Collect the dataset or Create the dataset
- Data Preprocessing.
- Import the Libraries.
- Importing the dataset.
- Checking for Null Values.
- Data Visualization.
- Taking care of Missing Data.
- Label Encoding.
- OneHot Encoding.
- Splitting Data into Train and Test.
- Feature Scaling.
- Model Building
- Import the model building Libraries
- Initializing the model
- Training and testing the model
- Evaluation of Model
- Save the Model
- Application Building
- Create an HTML file
- Build a Python Code

Project Structure:

Create a Project folder which contains files as shown below



- A python file called app.py for server side scripting.
- We need the model which is saved and the saved model in this content is **cerealanalysis.pkl**
- Templates folder which contains base.HTML file, index.HTML file, prediction.HTML file.
- Static folder which contains css folder which contains style.css .

Milestone 1: Data Collection:

ML depends heavily on data, without data, it is impossible for an "AI" to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training **data set.** It is the actual **data set** used to train the model for performing various actions.

Activity1: Download the dataset

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository etc.

The dataset used for this project was obtained from Kaggle . Please refer to the link given below to download the data set and to know about the dataset

https://www.kaggle.com/crawford/80-cereals

Milestone 2: Data Preprocessing

Data Pre-processing includes the following main tasks

- Import the Libraries.
- Importing the dataset.
- Checking for Null Values.
- Data Visualization.

- Label Encoding.
- OneHot Encoding.
- Splitting Data into Train and Test.

Activity 1: Import The Libraries

- It is important to import all the necessary libraries such as pandas, numpy, matplotlib.
- **Numpy**- It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.
- **Pandas**- It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- **Seaborn**- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **Matplotlib**- Visualisation with python. It is a comprehensive library for creating static,animated, and interactive visualizations in Python

```
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Activity 2: Read the Datasets

- You might have your data in .csv files, .excel files

- Let's load a .csv data file into pandas using **read_csv() function.**We will need to locate the directory of the CSV file at first (it's more efficient to keep the dataset in the same directory as your program).

```
In [4]: df = pd.read_csv('pmsm_temperature_data.csv')
        df.head()
```

	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	stator_
0	-0.752143	-1.118446	0.327935	-1.297858	-1.222428	-0.250182	1.029572	-0.245860	-2.522071	-1.831422	-2.066143	-2.0180
1	-0.771263	-1.117021	0.329665	-1.297686	-1.222429	-0.249133	1.029509	-0.245832	-2.522418	-1.830969	-2.064859	-2.0176
2	-0.782892	-1.116681	0.332771	-1.301822	-1.222428	-0.249431	1.029448	-0.245818	-2.522673	-1.830400	-2.064073	-2.0173
3	-0.780935	-1.116764	0.333700	-1.301852	-1.222430	-0.248636	1.032845	-0.246955	-2.521639	-1.830333	-2.063137	-2.0176
4	-0.774043	-1.116775	0.335206	-1.303118	-1.222429	-0.248701	1.031807	-0.246610	-2.521900	-1.830498	-2.062795	-2.0181

- If your dataset is in some other location ,Then
  Data=pd.read_csv(r"File_location")
  Note:r stands for "raw" and will cause backslashes in the string to be interpreted as actual backslashes rather than special characters.

- If the dataset is in the same directory of your program, you can directly read it, without giving raw as r.

- Our Dataset pmsm_temperature contains following Columns

1. ambient
2. coolant
3. U_d
4. U_q
5. Motor_speed
6. torque
7. I_d
8. I_q
9. pm
10. Stator_yoke
11. Stator_tooth
12. stator

The output column to be predicted is **rating** .Based on the input variables we predict the temperature

Activity 3: Analyse the data

- head() method is used to return top n (5 by default) rows of a DataFrame or series.


```
In [4]: df = pd.read_csv('pmsm_temperature_data.csv')
        df.head()
```

	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	stator_
0	-0.752143	-1.118446	0.327935	-1.297858	-1.222428	-0.250182	1.029572	-0.245860	-2.522071	-1.831422	-2.066143	-2.0180
1	-0.771263	-1.117021	0.329665	-1.297686	-1.222429	-0.249133	1.029509	-0.245832	-2.522418	-1.830969	-2.064859	-2.0176
2	-0.782892	-1.116681	0.332771	-1.301822	-1.222428	-0.249431	1.029448	-0.245818	-2.522673	-1.830400	-2.064073	-2.0173
3	-0.780935	-1.116764	0.333700	-1.301852	-1.222430	-0.248636	1.032845	-0.246955	-2.521639	-1.830333	-2.063137	-2.0176
4	-0.774043	-1.116775	0.335206	-1.303118	-1.222429	-0.248701	1.031807	-0.246610	-2.521900	-1.830498	-2.062795	-2.0181

- describe() method computes a summary of statistics like count, mean, standard deviation, min, max and quartile values.

```
data.describe()
```

The output is as shown below

```
In [9]: df.describe()
```

	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm
count	942677.000000	942677.000000	942677.000000	942677.000000	942677.000000	942677.000000	942677.000000	942677.000000	942677.
mean	-0.030010	-0.008238	-0.021835	0.007720	0.003210	0.020170	-0.002465	0.019958	-0.0050
std	1.007729	1.009503	0.994308	0.996054	0.996456	0.999063	1.000106	0.999683	1.00141
min	-8.573954	-1.429349	-1.655373	-1.861463	-1.371529	-3.345953	-3.245874	-3.341639	-2.6319
25%	-0.634542	-1.041886	-0.854390	-0.881885	-0.951878	-0.265042	-0.760764	-0.254672	-0.6678
50%	0.242495	-0.185147	0.225416	-0.089520	-0.140245	-0.121022	0.196713	-0.091449	0.09915
75%	0.681905	0.698607	0.356361	0.859836	0.855827	0.561778	1.013952	0.543750	0.67784
max	2.967117	2.649032	2.274734	1.793498	2.024164	3.016971	1.060937	2.914185	2.91745

info() gives information about the data

```
In [8]: df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 942677 entries, 0 to 982769
Data columns (total 13 columns):
ambient           942677 non-null float64
coolant           942677 non-null float64
u_d               942677 non-null float64
u_q               942677 non-null float64
motor_speed       942677 non-null float64
torque            942677 non-null float64
i_d               942677 non-null float64
i_q               942677 non-null float64
pm                942677 non-null float64
stator_yoke       942677 non-null float64
stator_tooth      942677 non-null float64
stator_winding    942677 non-null float64
profile_id        942677 non-null int64
dtypes: float64(12), int64(1)
memory usage: 100.7 MB
```

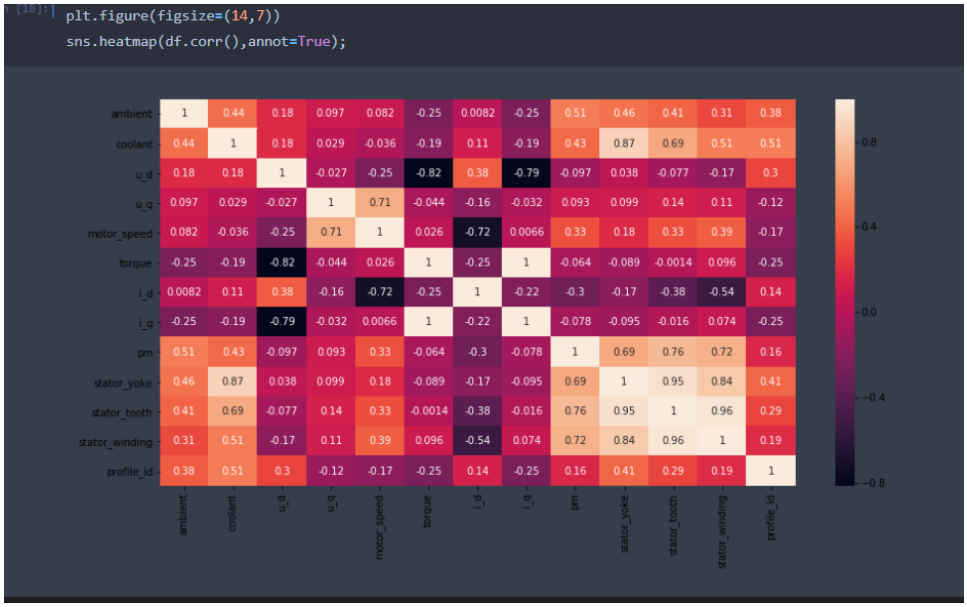
Activity 4: Handling Missing Values

1. After loading it is important to check the complete information of data as it can indication many of the hidden information such as null values in a column or a row
2. Check whether any null values are there or not. if it is present then following can be done,
3. a. Imputing data using Imputation method in sklearn
   b. Filling NaN values with mean, median and mode using fillna() method.

```
In [11]: df.isnull().sum()

ambient          0
coolant          0
u_d              0
u_q              0
motor_speed      0
torque           0
i_d              0
i_q              0
pm               0
stator_yoke      0
stator_tooth     0
stator_winding   0
profile_id       0
dtype: int64
```

4. Heatmap:Ity of representing the data in 2-D form.It gives coloured visual summary of the data

```
[18]: plt.figure(figsize=(14,7))
      sns.heatmap(df.corr(),annot=True);
```



From the heatmap, we see that there are no missing values in the dataset

Activity 5: Data Visualisation

- Data visualization is where a given data set is presented in a graphical format. It helps the

detection of patterns, trends and correlations that might go undetected in text-based data.

- Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine learning models will perform poorly on data that wasn't visualized and understood properly.
- To visualize the dataset we need libraries called Matplotlib and Seaborn.
- The Matplotlib library is a Python 2D plotting library which allows you to generate plots, scatter plots, histograms, bar charts etc.

Let's visualize our data using Matplotlib and seaborn library.

Before diving into the code, let's look at some of the basic properties we will be using when plotting.

xlabel: Set the label for the x-axis.

ylabel: Set the label for the y-axis.

Visualizing And Analyzing The Data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There is n number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

title: Set a title for the axes.

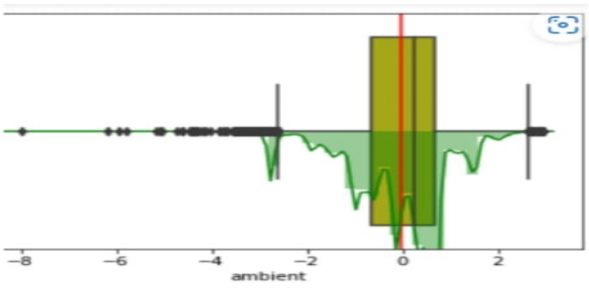
Legend: Place a legend on the axes.

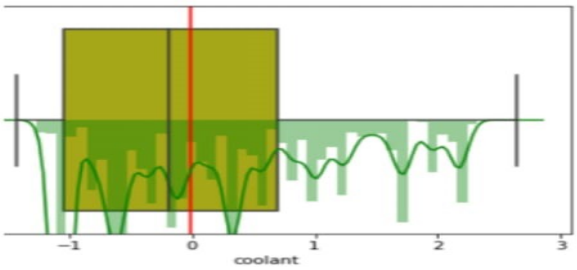
1. data.corr() gives the correlation between the columns

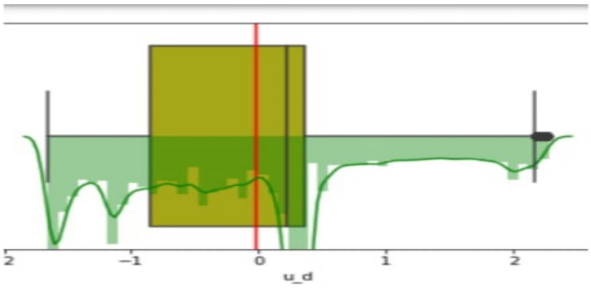
Out[51]:

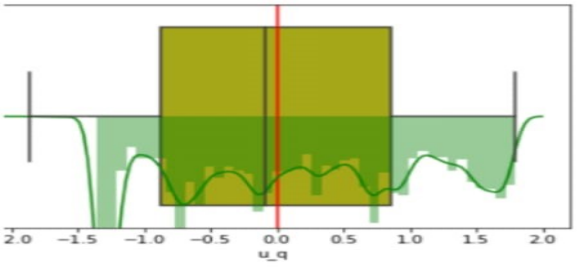
	ambient	coolant	u_d	u_q	motor_speed	i_d	i_q
505495	0.270204	0.288679	0.462250	-0.247227	-0.678016	0.968799	-0.573988
580705	-0.123966	1.937632	0.299465	-1.293001	-1.222430	1.029135	-0.245726
740606	0.688609	-0.171957	1.522243	1.061857	1.528602	-0.845077	-0.820782
225890	-1.158055	-1.042933	-0.469943	1.475568	1.834745	-0.777329	-0.018343
540496	0.688188	1.747456	0.947134	0.657506	-0.151493	0.873817	-0.907738
334701	0.567773	2.179385	0.184844	1.701340	1.753579	-0.491803	-0.253110
570735	-0.503207	0.792959	0.336665	-1.322363	-1.222430	1.029150	-0.245701
963858	0.062732	1.690038	0.299669	-1.284062	-1.222428	1.029160	-0.245717
865515	0.328729	-0.636870	2.218716	-0.557558	0.580682	-1.731411	-1.690000
507954	1.094502	-0.602857	-0.792172	-0.053149	-0.469331	0.081380	1.510606
534163	0.686783	-1.211159	2.228326	-0.769602	0.799096	-1.937165	-1.545722
20574	-0.085315	-1.054022	-0.964634	0.641086	-0.140245	0.386210	1.136766
115315	0.340695	-1.043654	-1.607185	-0.728241	1.888851	-2.021641	0.407147
81292	-1.111783	-1.054915	0.516815	-0.159968	-0.646163	0.994297	-0.617409
743875	0.688610	0.561473	1.872186	0.562416	0.082065	0.177971	-1.729435
25052	0.101937	-1.089592	-1.026388	0.927615	1.212482	-0.837578	0.322729
481042	0.706319	2.178387	0.947590	1.590360	1.070865	-0.067005	-0.592533
4766	-0.928489	-1.038371	-1.249375	0.485853	2.024120	-1.356766	0.236092
550415	2.135019	0.609588	0.985079	1.573049	0.621382	0.329669	-0.674377
359475	0.246263	1.145789	0.296599	-0.179645	-0.681333	1.029104	-0.245706
459595	0.663342	-0.451665	-1.492780	-0.129559	0.051545	-1.120119	1.489307
506467	0.700919	-0.394141	-0.457578	-0.766979	-0.844126	-0.963513	2.395487
772133	0.425667	-0.612044	0.305696	-1.215681	-1.204866	1.022718	-0.198039
907962	0.297616	1.345934	0.315753	-1.248850	-1.222430	1.029157	-0.245717
520543	0.578506	1.403250	1.500266	1.174667	0.196951	0.678656	-1.222573
700071	0.433112	0.501680	0.301654	-1.298501	-1.222431	1.029151	-0.245702
350402	-0.960295	1.089865	0.207570	-0.702521	-0.951871	0.961426	0.131018
839899	1.326457	-1.087187	2.039411	0.102632	0.602405	-1.025754	-1.457595
629347	0.188282	2.175059	0.312571	-1.325941	-1.222430	1.029137	-0.245717
862138	-0.054431	-0.639774	0.316296	-1.325994	-1.222431	1.029147	-0.245713
...	...	...	...	...	...	...	...
904590	0.293567	0.387088	-0.745101	1.237165	0.110186	0.710653	0.642105
240556	-2.291222	-1.042645	-1.610038	-0.653718	0.822903	-1.939363	0.821465

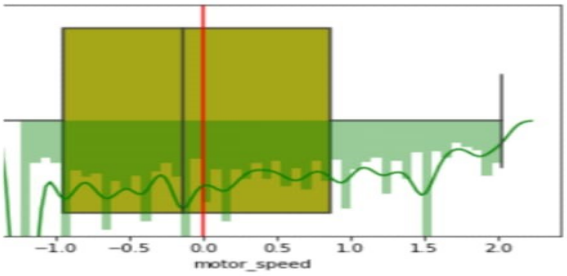
2. Box plot

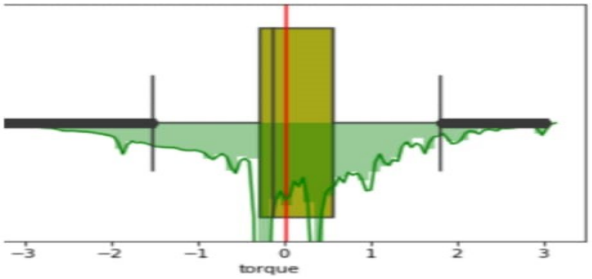

−8
−6
−4
−2
0
2
ambient

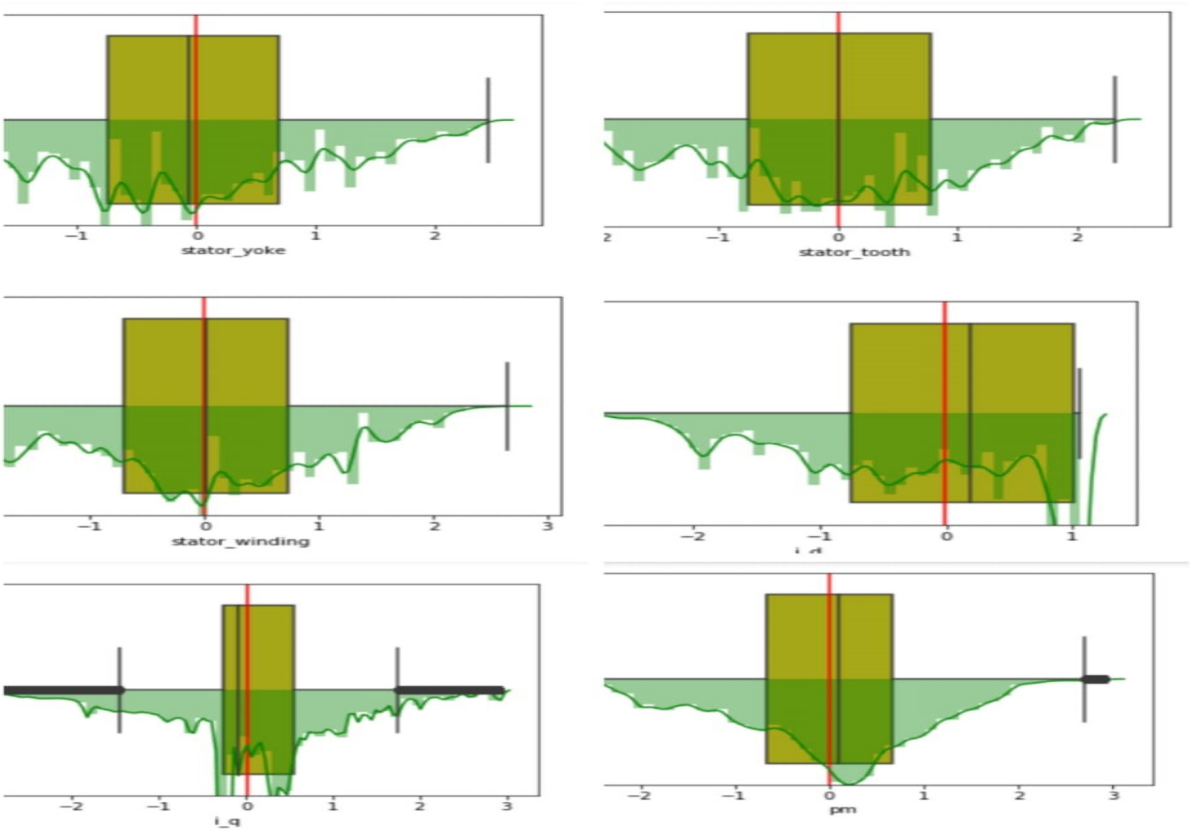

−1
0
1
2
3
coolant


2
−1
0
1
2
u_d

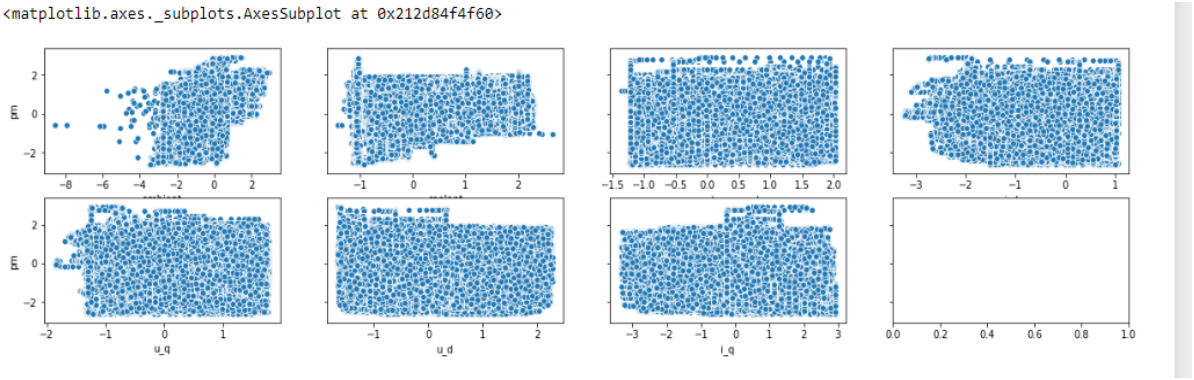

2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
u_q


−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
motor_speed


−3
−2
−1
0
1
2
3
torque




```
<matplotlib.axes._subplots.AxesSubplot at 0x212d84f4f60>
```



Activity 6: Label Encoding

In machine learning, we usually deal with datasets which contain multiple labels in one or more than one column. These labels can be in the form of words or numbers. To make the data understandable or in human readable form, the training data is often labelled in words.

Label Encoding refers to converting the labels into numeric form so as to convert it into the machine- readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important preprocessing step for the structured dataset in supervised learning.

```
df = pd.read_csv('pmsm_temperature_data.csv')
df.head()
```

Data after label encoding

	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	stator_winding	profile_id
0	-0.752143	-1.118446	0.327935	-1.297858	-1.222428	-0.250182	1.029572	-0.245860	-2.522071	-1.831422	-2.066143	-2.018033	4
1	-0.771263	-1.117021	0.329665	-1.297686	-1.222429	-0.249133	1.029509	-0.245832	-2.522418	-1.830969	-2.064859	-2.017631	4
2	-0.782892	-1.116681	0.332771	-1.301822	-1.222428	-0.249431	1.029448	-0.245818	-2.522673	-1.830400	-2.064073	-2.017343	4
3	-0.780935	-1.116764	0.333700	-1.301852	-1.222430	-0.248636	1.032845	-0.246955	-2.521639	-1.830333	-2.063137	-2.017632	4
4	-0.774043	-1.116775	0.335206	-1.303118	-1.222429	-0.248701	1.031807	-0.246610	-2.521900	-1.830498	-2.062795	-2.018145	4

All the data is converted into numerical values.

Activity 7: Splitting the Dataset into Dependent and Independent variable

- In machine learning, the concept of dependent variable (y) and independent variables(x) is important to understand. Here, Dependent variable is nothing but output in the dataset and the independent variable is all inputs in the dataset.
- With this in mind, we need to split our dataset into the matrix of independent variables and the vector or dependent variable. Mathematically, Vector is defined as a matrix that has just one column.

To read the columns, we will use **iloc** of pandas (used to fix the indexes for selection) which takes two parameters — [row selection, column selection].

Let's split our dataset into independent and dependent variables.

1. The independent variable in the dataset would be considered as 'x' and name, mfr, type, calories , protein , fat, sodium , fiber, carbo, sugars , potass , vitamins , shelf , weight , cups columns would be considered as independent variables.
2. The dependent variable in the dataset would be considered as 'y' and the ' rating ' column is considered as dependent variable.

Now we will split the data of independent variables,

```
x= data.iloc[:,0:14].values
y= data.iloc[:,14:15].values
```

From the above code **":"** indicates that you are considering all the rows in the dataset and **"0:18"** indicates that you are considering columns 0 to 8 such as sex, job and purpose as input values and assigning them to variable x. In the same way in the

second line **":"** indicates you are considering all the rows and **"18:19"** indicates that you are considering only the last column as output value and assigning them to variable y.

After splitting we see the data as

below x

```
x

array([[  3.  ,   0.  ,  70.  , ...,   3.  ,   1.  ,   0.33],
       [  5.  ,   0.  , 120.  , ...,   3.  ,   1.  ,   1.  ],
       [  2.  ,   0.  ,  70.  , ...,   3.  ,   1.  ,   0.33],
       ...,
       [  6.  ,   0.  , 100.  , ...,   1.  ,   1.  ,   0.67],
       [  1.  ,   0.  , 100.  , ...,   1.  ,   1.  ,   1.  ],
       [  1.  ,   0.  , 110.  , ...,   1.  ,   1.  ,   0.75]])
```

Y

```
y

array([[68.402973],
       [33.983679],
       [59.425505],
       [93.704912],
       [34.384843],
       [29.509541],
       [33.174094],
       [37.038562],
       [49.120253],
       [53.313813],
       [18.042851],
       [50.764999],
       [19.823573],
       [40.400208],
       [22.736446],
       [41.445019],
       [45.863324],
       [35.782791],
       [22.396513],
       [40.448772],
       [64.533816],
       [46.895644],
```

Activity 8: OneHot Encoding

Sometimes in datasets, we encounter columns that contain numbers of no specific order of preference. The data in the column usually denotes a category or value of the category and also when the data in the column is label encoded. This confuses the machine learning model, to avoid this, the data in the column should be One Hot encoded.

One Hot Encoding –

It refers to splitting the column which contains numerical categorical data to many columns depending on the number of categories present in that column. Each column contains “0” or “1” corresponding to which column it has been placed.

```
from sklearn.preprocessing import OneHotEncoder
one = OneHotEncoder()
a = one.fit_transform(x[:,0:1]).toarray()
x = np.delete(x,[0],axis=1)
x=np.concatenate((a,x),axis=1)
```

Activity 9: Splitting the data into Train and Test

- When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test dataset. For this, you will have a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.
- But the question is, how do you split the data? You can’t possibly manually split the dataset into two sets. And you also have to make sure you split the data in a random manner. To help us with this task, the Scikit library provides a tool, called the Model Selection library. There is a class in the library which is,**‘train_test_split.’** Using this we can easily split the dataset into the training and the testing datasets in various proportions.
- The train-test split is a technique for evaluating the performance of a machine learning algorithm.
- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.
- In general you can allocate 80% of the dataset to training set and the remaining 20% to test set.We will create 4 sets— X_train (training part of the matrix of features), X_test (test part of the matrix of features), Y_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices), Y_test (test part of the dependent variables associated with the X test sets, and therefore also the same indices.
- There are a few other parameters that we need to understand before we use the class:
- **test_size** — this parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset
- **train_size** — you have to specify this parameter only if you’re not specifying the test_size. This is the same as test_size, but instead you tell the class what percent of the dataset you want to split as the training set.
- **random_state** — here you pass an integer, which will act as the seed for the random number generator during the split. Or, you can also pass an instance of the Random_state class, which

will become the number generator. If you don't pass anything, the Random_state instance used by np.random will be used instead.

- Now split our dataset into a train set and test using train_test_split class from scikit learn library.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
```

Milestone 3: Model Building:

Model building includes the following main tasks

- Import the model building Libraries
- Initializing the model
- Training and testing the model
- Evaluation of Model
- Save the Model

Activity 1: Training and Testing the Model

- Once after splitting the data into train and test, the data should be fed to an algorithm to build a model.
- There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have may be Classification algorithms or Regression algorithms.

  1.Linear Regression

  2.Decision Tree

  Regressor 3.Random

  Forest Regressor

We're going to use x_train and y_train obtained above in train_test_split section to train our decision tree regression model. We're using the fit method and passing the parameters as shown below.

We are using the algorithm from Scikit learn library to build the model as shown below,

Support Vector Machine Model

A function named SVR is created and train and test data are passed as the parameters. Inside the function, SVR algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the performance of the model, we use root mean square error and r-square value.

```
In [51]: from sklearn.linear_model import LinearRegression
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.svm import SVR

In [52]: lr=LinearRegression()
         dr=DecisionTreeRegressor()
         rf =RandomForestRegressor()
         svm =SVR()
```

```
In [*]: lr.fit(X_train,y_train)
        dr.fit(X_train,y_train)
        rf.fit(X_train,y_train)
        svm.fit(X_train,y_train)
```

Activity 2: Model Evaluation

After training the model, the model should be tested by using the test data which has been separated while splitting the data for checking the functionality of the model.

Regression Evaluation Metrics:

These model evaluation techniques are used to find out the accuracy of models built in regression type of machine learning models. We have three types of evaluation methods.

1. **R – Square** (R^2)

   Formula for calculating (R^2) is given by:

$$R2 = \frac{TSS - RSS}{TSS}$$

- Total Sum of Squares (TSS) : TSS is a measure of total variance in the response/ dependent variable Y and can be thought of as the amount of variability inherent in the response before the regression is performed.
- Residual Sum of Squares (RSS) : RSS measures the amount of variability that is left unexplained after performing the regression.
- (TSS – RSS) measures the amount of variability in the response that is explained (or removed) by performing the regression

  Where N is the number of observations used to fit the model, σx is the standard deviation of x, and σy is the standard deviation of y.

  - R^2 ranges from 0 to 1.
  - R^2 of 0 means that the dependent variable cannot be predicted from the independent variable
  - R^2 of 1 means the dependent variable can be predicted without error from the independent variable
  - An R^2 between 0 and 1 indicates the extent to which the dependent variable is predictable. An R^2 of 0.20 means that 20 percent of the variance in Y is predictable from X; an R^2 of 0.40 means that 40 percent is predictable; and so on.

2. **Root Mean Square Error (RMSE)**

   RMSE tells the measure of dispersion of predicted values from actual values. The formula for calculating RMSE is

$$R^2 = \{(1) * \sum (x - mean(x)) * (y - mean(y))]/(\sigma * \sigma)\}^2$$

N

I x y

N:Total number of observation

```
In [20]: from sklearn.metrics import mean_squared_error

In [26]: print(mean_squared_error(y_test,p1))

         0.030187256377134934
```

N : Total number of observations

Though RMSE is a good measure for errors, the issue with it is that it is susceptible to the range of your dependent variable. If your dependent variable has a thin range, your RMSE will be low and if the dependent variable has a wide range RMSE will be high. Hence, RMSE is a good metric to compare between different iterations of a model.

3. **Mean Absolute Percentage Error (MAPE)**

To overcome the limitations of RMSE, analysts prefer MAPE over RMSE which gives error in terms of percentages and hence comparable across models. Formula for calculating MAPE can be written as:

$$RMSE = \sqrt{\frac{\sum (Y_{Actual} - Y_{Predicted})^2}{N}}$$

N : Total number of observations

For testing the model we use the below method,

```
from sklearn.metrics import r2_score
r2_score(y_test,lr_pred)
```

```
0.9999999999999992
```

Activity 3: Save the Model

After building the model we have to save the model.

Pickle in **Python** is primarily **used** in serializing and deserializing a **Python** object structure. In other words, it's the process of converting a **Python** object into a byte stream to store it .

file/database, maintain program state across sessions, or transport data over the network. wb indicates write method and rd indicates read method.

This is done by the below code

```
In [27]: import joblib

In [65]: joblib.dump(dr,"model.save")

['model.save']
```

Milestone 4 : Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server side script

Activity 1: Build HTML Code

- In this HTML page, we will create the front end part of the web page. In this page we will accept input from the user and Predict the values.

  For more information regarding HTML

  https://www.w3schools.com/html/

  In our project we have 3 HTML files ,they are

1. base.html
2. index.html
3. prediction.html

base.html

Build An HTML Page

We Build an HTML page to take the values from the user in a form and upon clicking on the predict button we get the temperature predicted. The values predicted are normalized values according to the dataset. Hence units are not considered. You can get these files from the project folder.

Building Html Pages:

For this project, create three HTML files namely

- Manual_predict.html
- Sensor_predict.html

and save them in the templates folder.

For more information regarding HTML: Link

Let's see how our home.html page looks like

The html page looks

Electric Motor Temperature

Electric Motor Temperature Prediction

Fill in and below details to know predicted Permanent Magnet surface temperature representing the rotor temperature.

Ambient temperature

Coolant temperature

Voltage d-component

Voltage q-component

Motor speed

Current d-component

Current q-component

Activity 2: Main Python Script

Let us build an app.py flask file which is a web framework written in python for server-side scripting. Let's see the step by step procedure for building the backend application.

In order to develop web api with respect to our model, we basically use the Flask framework which is written in python.

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import joblib
app = Flask(__name__)
model = joblib.load("model.save")
trans=joblib.load('transform.save')


app = Flask(__name__)
```

Load the home page

```
app = Flask(__name__)

@app.route('/')
def predict():
    return render_template('Manual_predict.html')
```

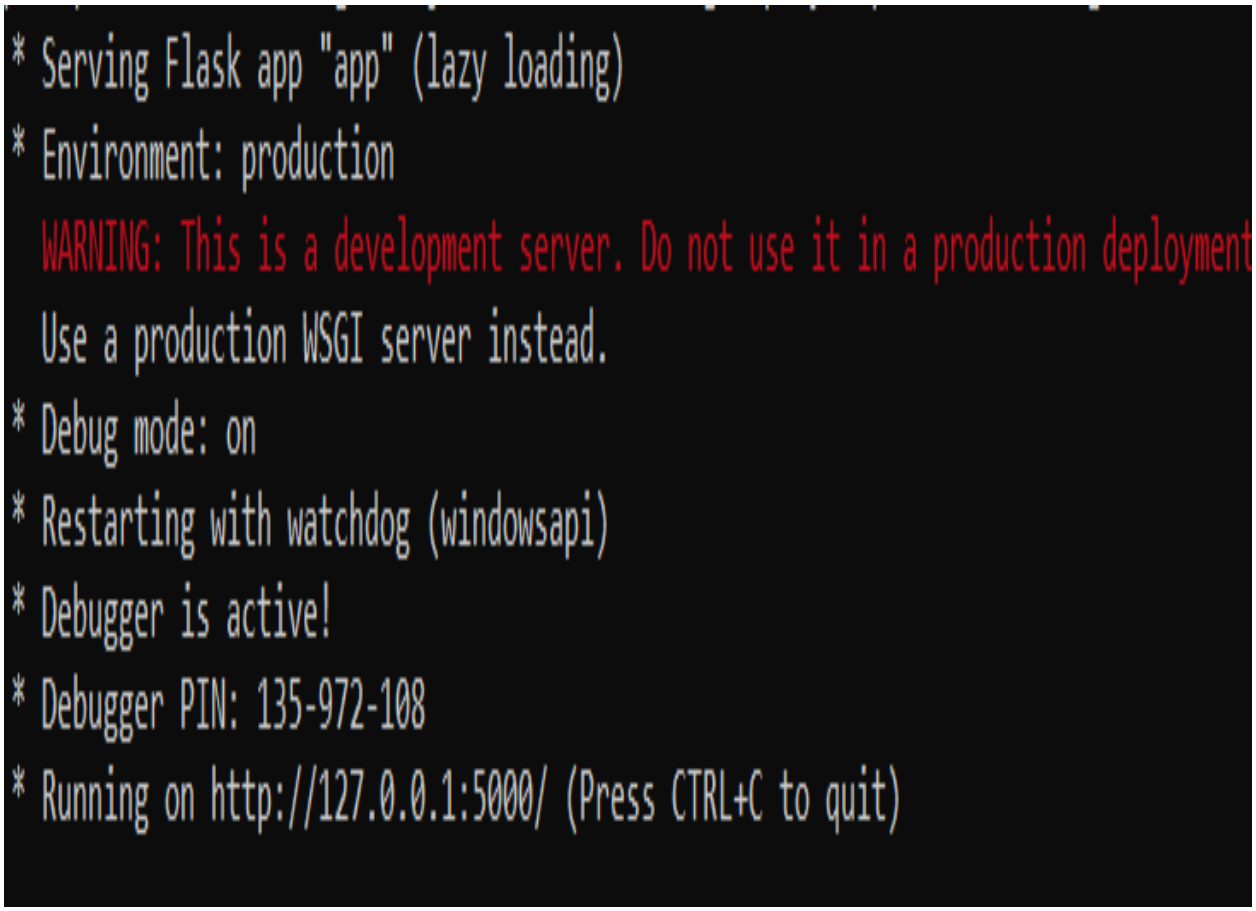
Prediction function

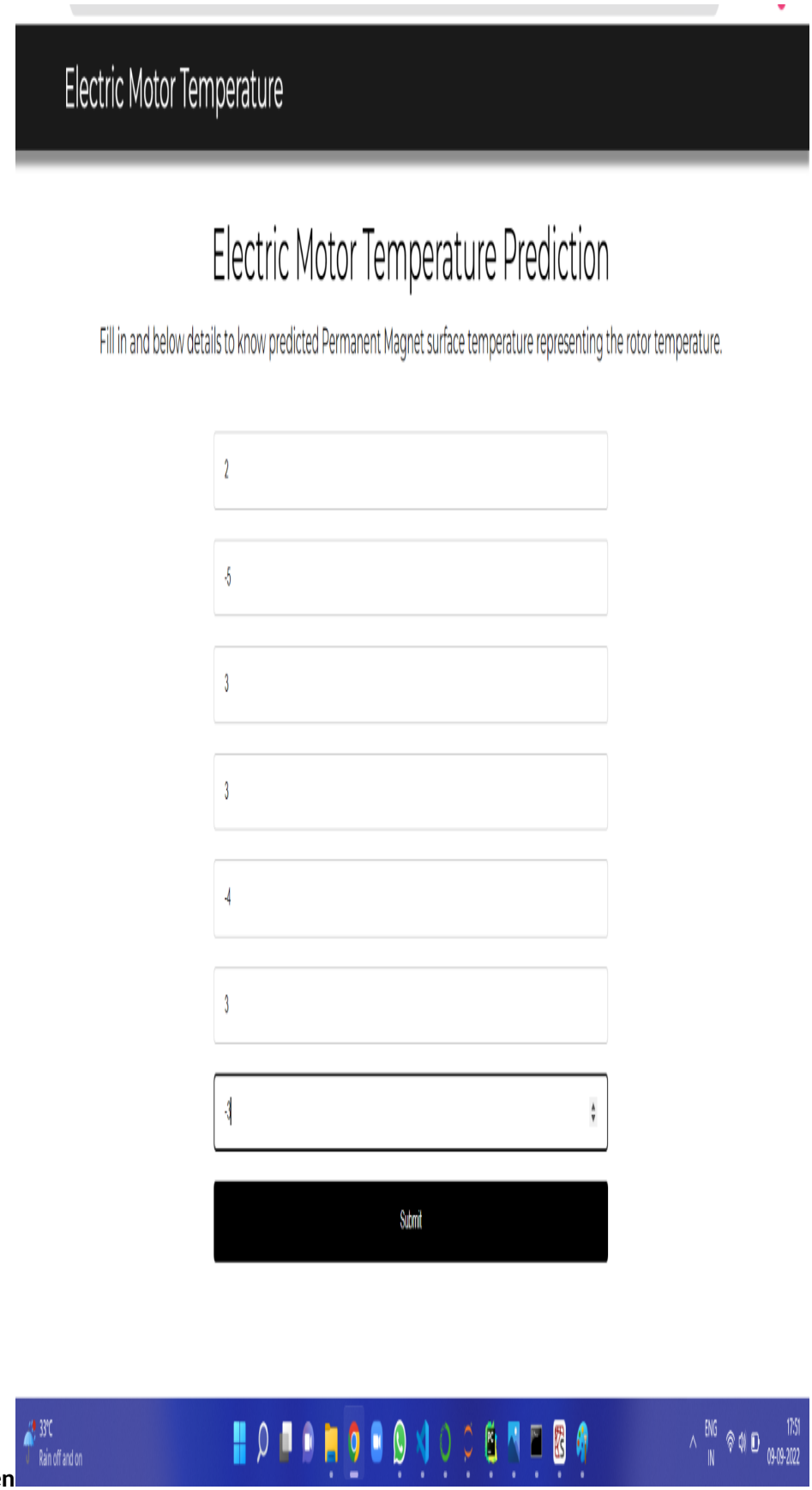
```
@app.route('/y_predict',methods=['POST'])
def y_predict():
    x_test = [[float(x) for x in request.form.values()]]
    print('actual',x_test)
    x_test=trans.transform(x_test)
    print(x_test)
    pred = model.predict(x_test)

    return render_template('Manual_predict.html', prediction_text=('Permanent Magnet surface temperature: ',pred[0]))


if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

Activity 3: Run the App

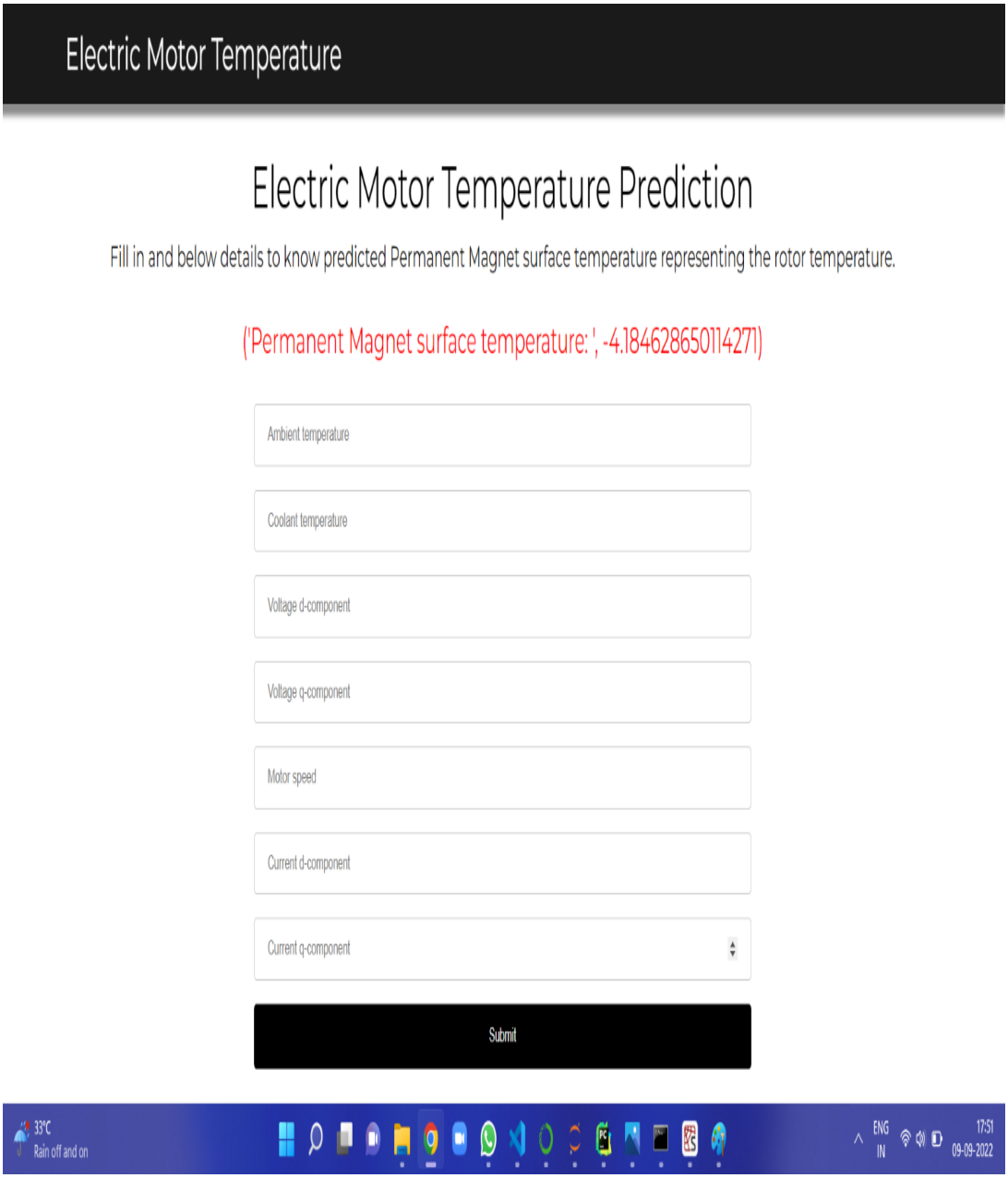
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page

Then it will run on localhost:5000


* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment
Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 135-972-108
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)



Output Screen


Electric Motor Temperature
Electric Motor Temperature Prediction
Fill in and below details to know predicted Permanent Magnet surface temperature representing the rotor temperature.
('Permanent Magnet surface temperature: ', -4.184628650114271)
Ambient temperature
Coolant temperature
Voltage d-component
Voltage q-component
Motor speed
Current d-component
Current q-component
Submit
33°C
Rain off and on
ENG
IN
17:51
09-09-2022