Electric Motor Temperature Prediction using Machine Learning

Project Description:

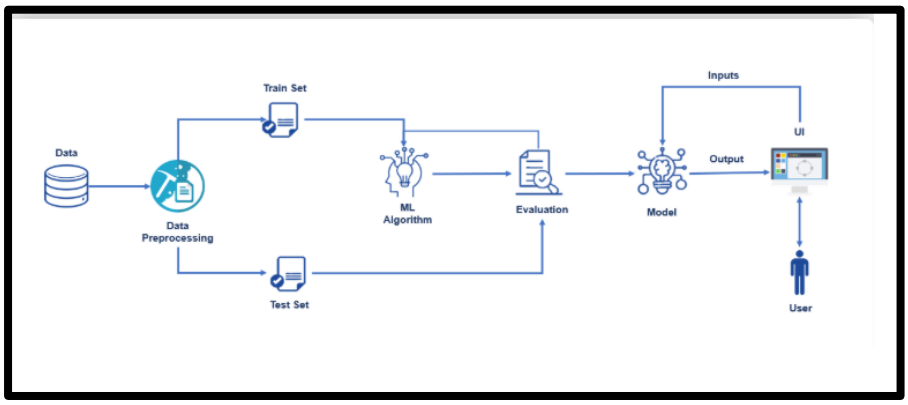
Electric Motor Temperature Prediction using Machine Learning is a predictive maintenance solution aimed at forecasting the temperature of electric motors in industrial settings. By analyzing historical operational data such as motor load, voltage, current, and environmental conditions, combined with machine learning algorithms, this project aims to predict motor temperatures accurately. The goal is to prevent overheating, avoid equipment failures, optimize maintenance schedules, and improve overall operational efficiency.

Scenario 1: Preventive Maintenance Manufacturing plants can use the temperature predictions to implement proactive maintenance strategies. By identifying potential overheating issues before they occur, maintenance teams can schedule timely inspections, replace worn-out components, and prevent costly downtime due to motor failures.

Scenario 2: Energy Efficiency Facility managers can leverage temperature predictions to optimize energy consumption. By maintaining motors at optimal temperature levels, they can reduce energy wastage, improve equipment performance, and lower operational costs over time.

Scenario 3: Equipment Reliability Industries relying heavily on electric motors, such as automotive production or HVAC systems, can benefit from accurate temperature predictions. Ensuring motors operate within safe temperature ranges enhances equipment reliability, prolongs lifespan, and minimizes the risk of unexpected breakdowns during critical operations.

Technical Architecture:



Pre requisites:

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

- **Anaconda navigator:**
  - Refer 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 scikit-learn"** and click enter.
  - Type **"pip install matplotlib"** and click enter.
  - Type **"pip install pickle-mixin"** and click enter.
  - Type **"pip install seaborn"** and click enter.
  - Type **"pip install Flask"** and click enter.

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- **ML Concepts**
  - Supervised learning: https://www.youtube.com/watch?v=QeKshry8pWQ
  - Unsupervised learning: https://www.youtube.com/watch?v=D6gtZrsYi6c
  - Regression: https://www.youtube.com/watch?v=NUXdtN1W1FE
  - Evalution Metrics: https://www.youtube.com/watch?v=YSB7FtzeicA&t=1s
  - 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 pre-processing techniques.
- Applying different algorithms according to the dataset
- You will be able to know how to find the accuracy of the model.
- You will be able to build web applications using the Flask framework.

Project Flow:

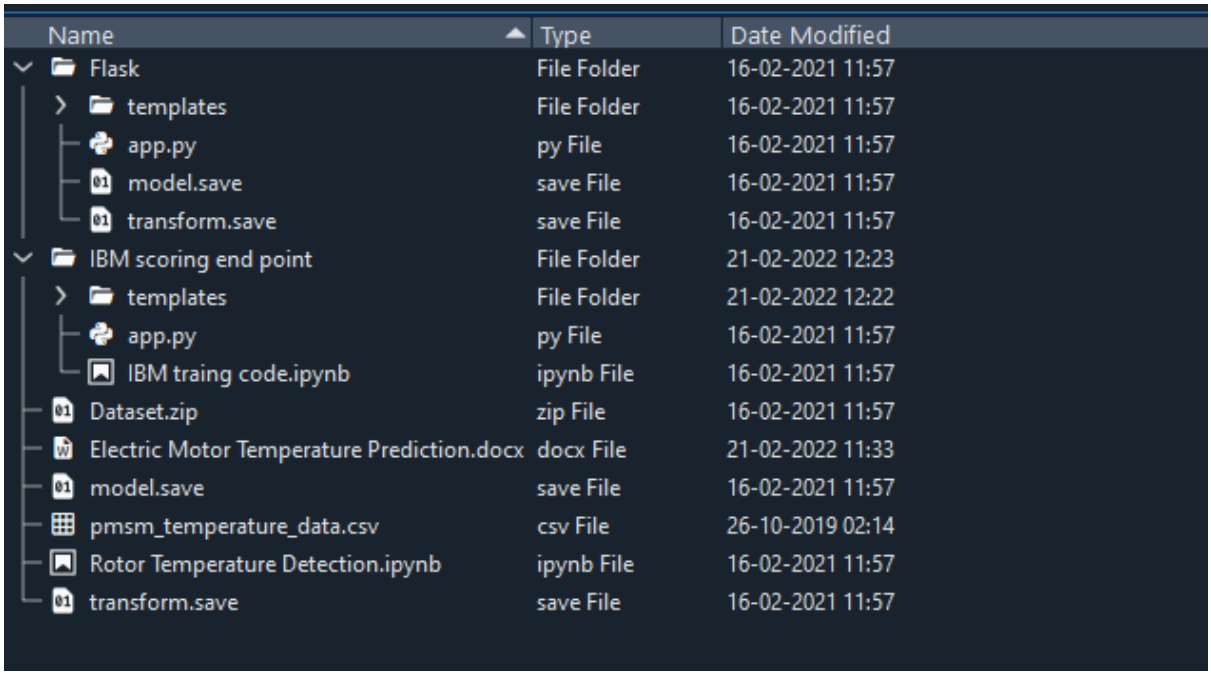
- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Data collection
  - Collect the dataset or create the dataset
- Visualizing and analyzing data
  - Univariate analysis
  - Multivariate analysis
  - Descriptive analysis
- Data pre-processing
  - Drop unwanted features
  - Checking for null values
  - Remove negative data
  - Handling outlier
  - Handling categorical data
  - Handling Imbalanced data
  - Splitting data into train and test
- Model building
  - Import the model building libraries
  - Initializing the model
  - Training and testing the model
  - Evaluating performance of model
  - Save the model
- Application Building
  - Create an HTML file
  - Build python code

Project Structure:

Create the Project folder which contains files as shown below



- Rotor Temperature Detection.ipynb is the jupyter notebook file where the model is built.
- Dataset.zip is the dataset file used in this project.
- model.save is the model file that generates when the notebook file is executed.
- transform.save is the transformation file used while building the model.
- Flask folder is the application folder where the web application and server-side program are present.
- IBM scoring endpoint is the folder that contains IBM training code and flask files related to IBM

Milestone 1: Data Collection

ML depends heavily on data, It is most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Activity 1: Download the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

Link: https://www.kaggle.com/datasets/wkirgsn/electric-motor-temperature

Milestone 2: Visualizing and analysing the data

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

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image To know about the packages refer to the link given on prerequisites.

```
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 Dataset

Read the datasets

The dataset is read as a data frame (df in our application) using the pandas' library (pd is the alias name given to the pandas package).

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

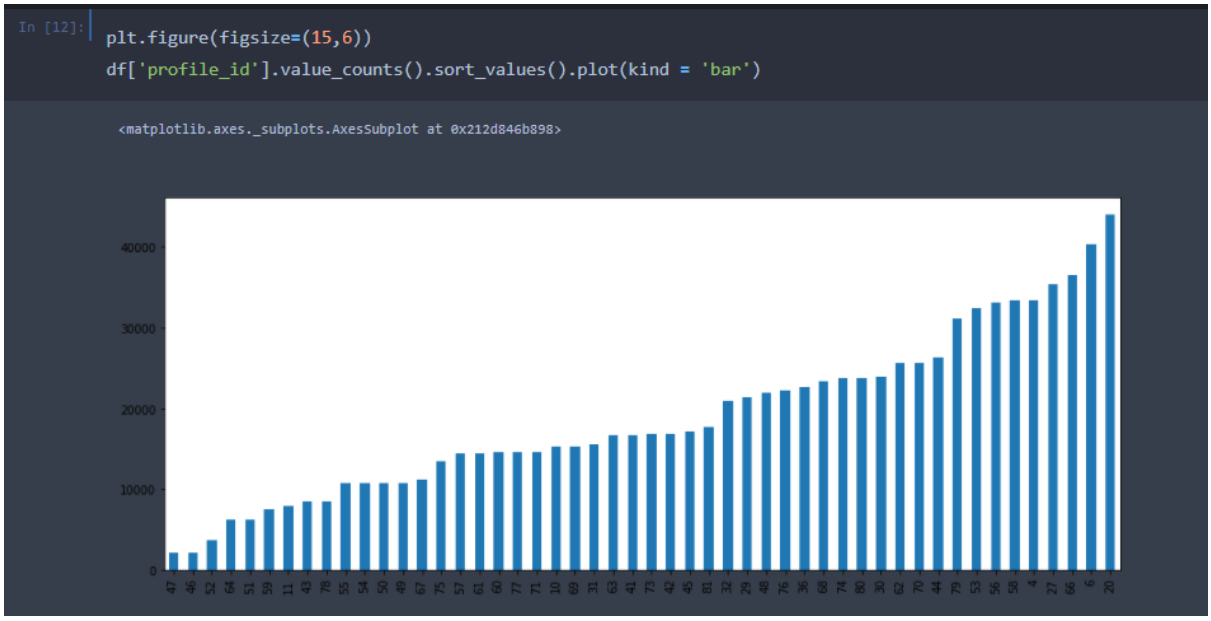
	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	stator_
0	-0.752143	-1.118446	0.327935	-1.297858	-1.222428	-0.250182	1.029572	-0.245860	-2.522071	-1.831422	-2.066143	-2.0180
1	-0.771263	-1.117021	0.329665	-1.297686	-1.222429	-0.249133	1.029509	-0.245832	-2.522418	-1.830969	-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

Activity 3: Uni-variate analysis

Here we get to know about our data

Bar Graph:

A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.



As we can see, session ids 66, 6, and 20 have the most number of measurements recorded.

Box plot:

A boxplot is a standardized way of displaying the distribution of data based on a five-number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”). It can tell you about your outliers and what their values are.

Distribution plot:

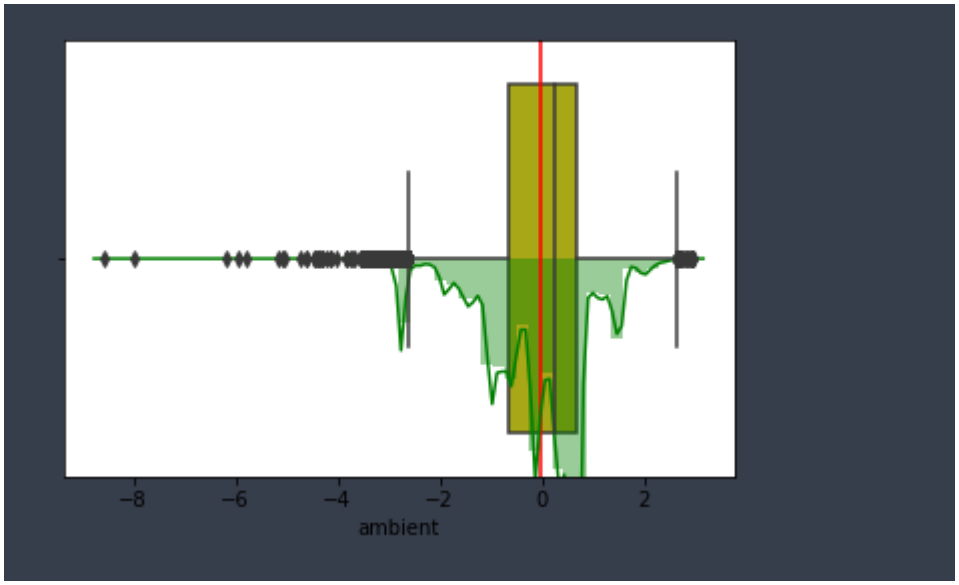
The distribution plot is suitable for comparing range and distribution for groups of numerical data. Data is plotted as value points along an axis.

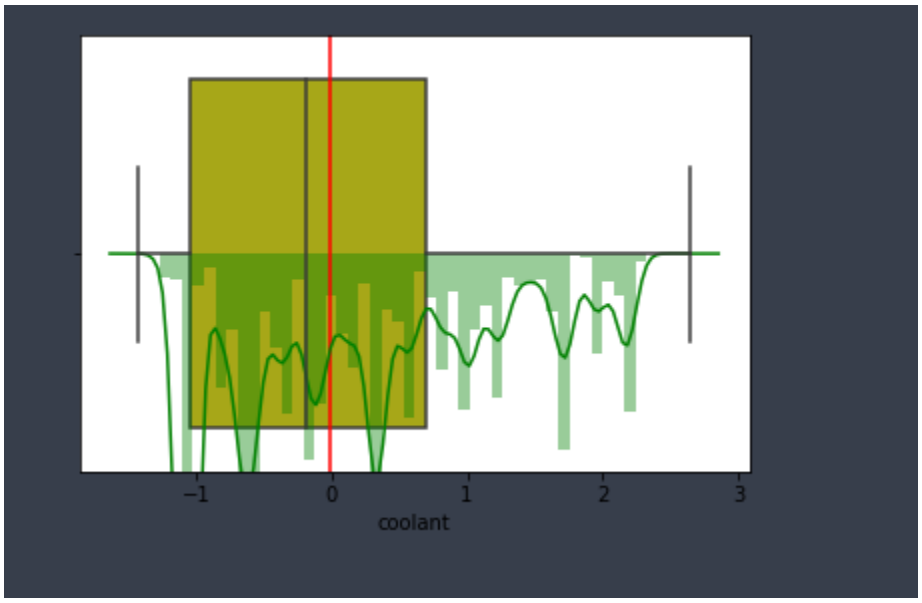
In [13]:
```
df.columns
```

```
Index(['ambient', 'coolant', 'u_d', 'u_q', 'motor_speed', 'torque', 'i_d',
       'i_q', 'pm', 'stator_yoke', 'stator_tooth', 'stator_winding',
       'profile_id'],
      dtype='object')
```

In [14]:
```
#Plotting Distribution and Boxplot for all the features to check for skewness
```

In [15]:
```
for i in df.columns:
    sns.distplot(df[i],color='g')
    sns.boxplot(df[i],color = 'y')
    plt.vlines(df[i].mean(),ymin = -1,ymax = 1,color = 'r')#drawing the mean line
    plt.show()
```





All features boxplots are plotted and the following conclusions are drawn.

As we can see from the above plots, the mean and median for most of the plots are very close to each other. So the data seems to have low skewness for almost all variables.

Activity 4: Multi-variate analysis

Multivariate analysis (MVA) is a Statistical procedure for the analysis of data involving more than one type of measurement or observation. It may also mean solving problems where more than one dependent variable is analyzed simultaneously with other variables.

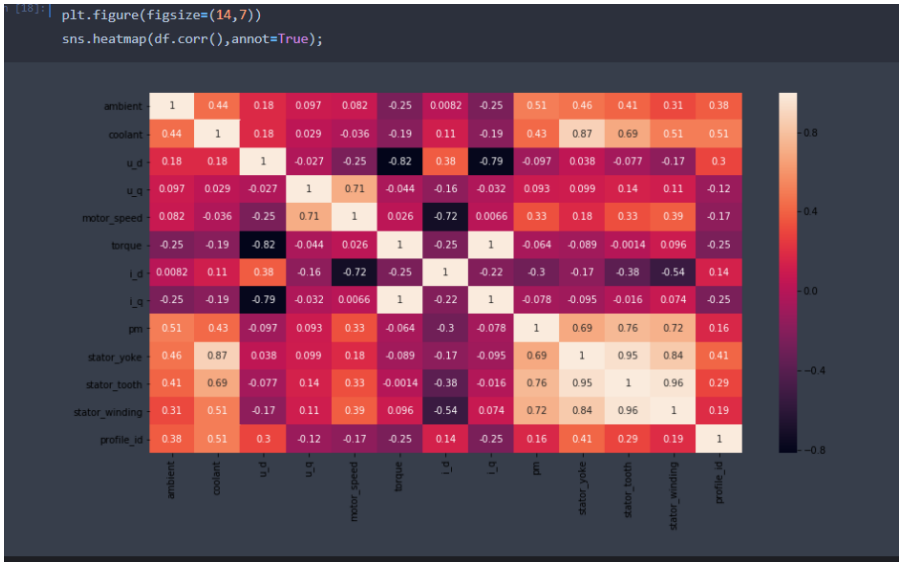
Scatterplot:

A scatter plot (also called a scatterplot, scatter graph, scatter chart, scattergram, or scatter diagram) is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.

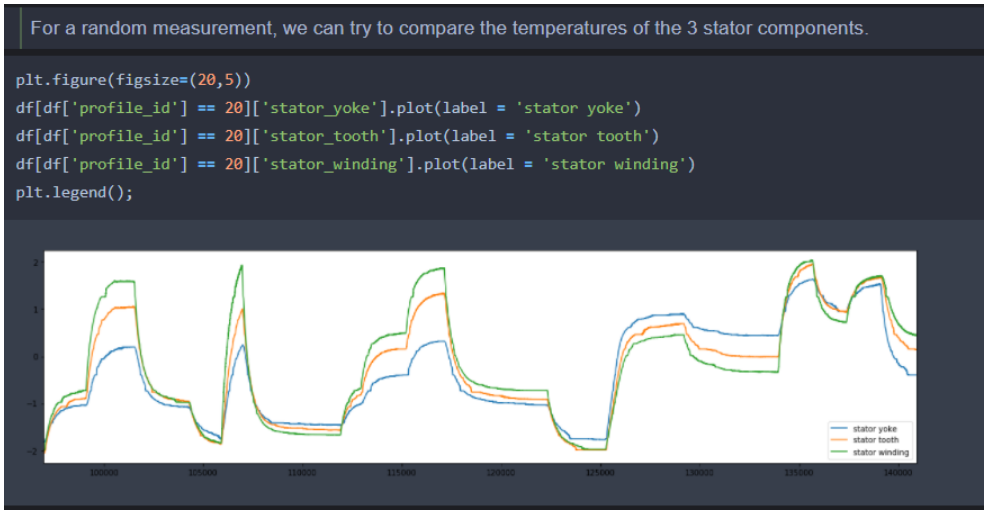
As we want to predict the temperatures of stator components and rotor(pm), we will drop these values from our dataset for regression. Also, torque is a quantity, which is not reliably measurable in field applications, so this feature shall be omitted in this modeling.

Heat-map:

A heat map is a data visualization technique that shows the magnitude of a phenomenon as color in two dimensions. The variation in color may be by hue or intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space.



From the heatmap above, we can see that torque and q component of current are almost perfectly correlated. Also, there seems to be a very high correlation between temperature measurements of stator yoke, stator tooth, and stator windings.



- As we can see from the plot, all three stator components follow a similar measurement variance.
- As the dataset author mentioned, the records in the same profile id have been sorted by time, we can assume that these recordings have been arranged a series of times.
- Due to this, we can infer that there has not been much time given for the motor to cool down in between recording the sensor data as we can see that initially the stator yoke temperature is low as compared to the temperature of stator winding but as we progress in time, the stator yoke temperature goes above the temperature of the stator winding.

- As profile_id is an id for each measurement session, we can remove it from any further analysis and model building.

```
In [20]: df.drop('profile_id',axis = 1,inplace=True)
         df_test.drop('profile_id',axis = 1,inplace=True)
```

Activity 5: Descriptive analysis

We'll see which particular variables contribute to the rotor temperature individually by checking their statistical significance.Descriptive analysis is to study the basic features of data with the statistical process. Here pandas have a worthy function called describe. With this described function we can understand the unique, top, and frequent values of categorical features. And we can find mean, std, min, max, and percentile values of continuous features.

df.info():

This function is used to display a brief introduction about the data set such as the. of rows and columns, the Data type of each column, whether the null values are present in the column or not.

```
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
```

df.describe()

This function is used to analyze the descriptive statistics of the data such as mean, median, quartile values, maximum and minimum values of each column.

```
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

Milestone 3: Data Pre-processing

As we have understood how the data is lets pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling outliers
- Scaling Techniques
- Splitting dataset into training and test set

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 1: Drop unwanted features

As we want to predict the temperatures of stator components and rotor(pm), we will drop these values from our dataset for regression. Also, torque is a quantity, which is not reliably measurable in field applications, so this feature shall be omitted in this modeling.
Dropping the columns from the dataset is being concluded with the help of a scatter plot, which is available in the data analysis part.

```
df.drop(['stator_yoke','stator_tooth','stator_winding','torque'],axis = 1)
```

Activity 2: Handling missing values

For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it. From the below image we found that the education column and previous year rating column have null values.

```
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
```

There are no null values in the dataset, we can skip this step.

Activity 3: Handling outliers

With the help of a boxplot, outliers are visualized (refer to activity 3 univariate analysis). And here we are going to find the upper bound and lower bound of the Na_to_K feature with some mathematical formula.

- To find the upper bound we have to multiply IQR (Interquartile range) with 1.5 and add it with 3rd quantile. To find a lower bound instead of adding, subtract it with 1st quantile. Take the image attached below as your reference.
- If outliers are removed, we lose more data. It will impact model performance.
- Here removing outliers is impossible. So, the capping technique is used on outliers.
- Capping: Replacing the outliers with upper bound values.
  Note: In our Dataset all the values are in the same range, so outliers replacing is not necessary.

Activity 4: Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.
To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project, we are using feature mapping and label encoding.

Note: In our dataset, there is no categorical data type, so we can skip this step.

Activity 5: Normalizing the values

As we want to predict the temperatures of stator components and rotor(pm), we will drop these values from our dataset for regression. Also, torque is a quantity, which is not reliably measurable in field applications, so this feature shall be omitted in this modeling.
We are using minmax scaler ,which is a function in preprocessing module in sklearn library

```
In [52]: mm = MinMaxScaler()
         X = mm.fit_transform(X)
         X_df_test = mm.fit_transform(X_df_test)
         y = df['pm']
         y_df_test = df_test['pm']
         X = pd.DataFrame(X,columns = ['ambient', 'coolant', 'u_d', 'u_q', 'motor_speed', 'i_d','i_q'])
         X_df_test = pd.DataFrame(X_df_test,columns = ['ambient', 'coolant', 'u_d', 'u_q', 'motor_speed', 'i_d','i_q
         y.reset_index(drop = True,inplace = True)
         y_df_test.reset_index(drop = True,inplace = True)
```

Saving the transformation

```
import joblib
joblib.dump(mm,'transform.save')
```

```
['transform.save']
```

Activity 6: Splitting data into train and test

Now let's split the Dataset into train and test sets. For splitting training and testing data, we are using the train_test_split() function from sklearn. As parameters, we are passing x_resample, y_resample, test_size, random_state.

For deep understanding refer to this link

```
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 4: Model Building

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project, we are applying four Regression algorithms. The best model is saved based on its performance. To evaluate the performance of the model, we use root mean square error and r-square value.

Activity 1: Linear Regression

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

Activity 2: Decision tree model

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

Activity 3: Random forest model

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, the RandomForestRegressor 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 evaluate the performance of the model, we use root mean square error and r-square value.

Activity 4: 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)
```

Now let's see the performance of all the models and save the best model

Activity 5: Compare the model

```
from sklearn import metrics

print(metrics.r2_score(y_test,p1))
print(metrics.r2_score(y_test,p2))
print(metrics.r2_score(y_test,p3))
print(metrics.r2_score(y_test,p4))
```

```
0.9698725757690718
0.4775746977817083 6
```

Out of all the models. The decision Tree regressor is giving an r2-score of 96%, it means the model is able to explain 96% of the data. so we will select the decision tree model and save it.

To know more about R2-score, please refer to the below link

Activity 6: Evaluating performance of the model

Evaluating the model by using RMSE (Root Mean Squared Error)

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

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

         0.030187256377134934
```

From the above picture, we can infer that the RMSE value is 0.03, which is very low.

It means our predicted values and actual values are almost equal. The difference between actual values and predicted values is very less. so we are considering this model.

Activity 7: Save the model

```
In [27]: import joblib

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

         ['model.save']
```

Milestone 5: Application Building

In this section, we will be building a web application that is integrated into 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

After the model is built, we will be integrating it into a web application

Activity1: Build the python flask app

In the flask application, the user values are taken from the HTML page

```
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)
```

Activity2: 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:

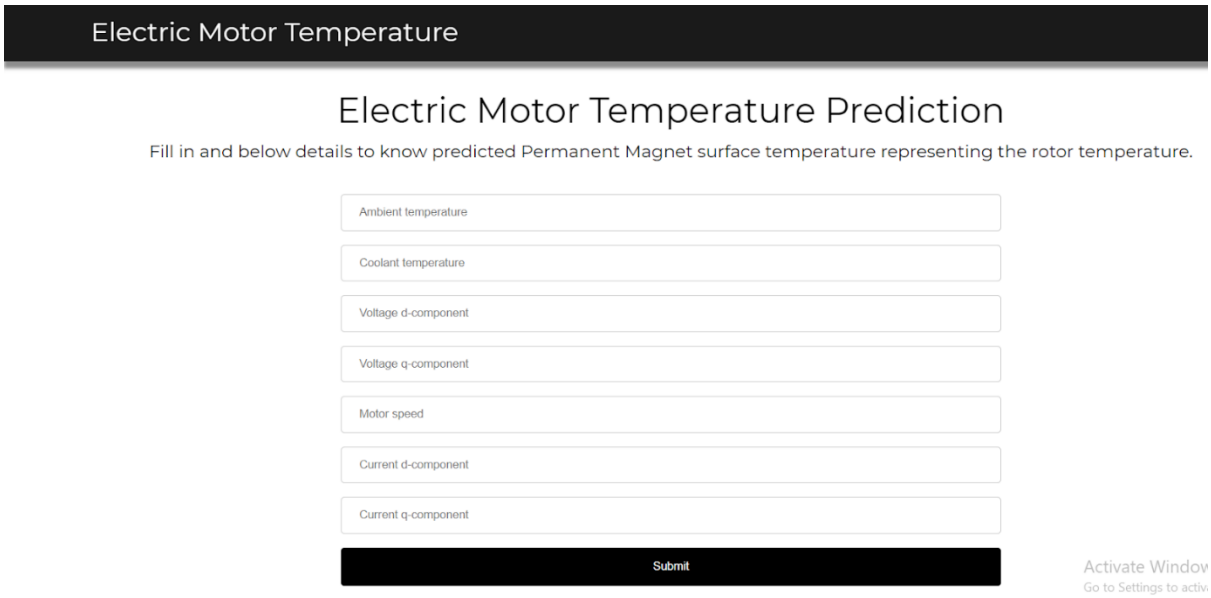
For this project, create three HTML files namely

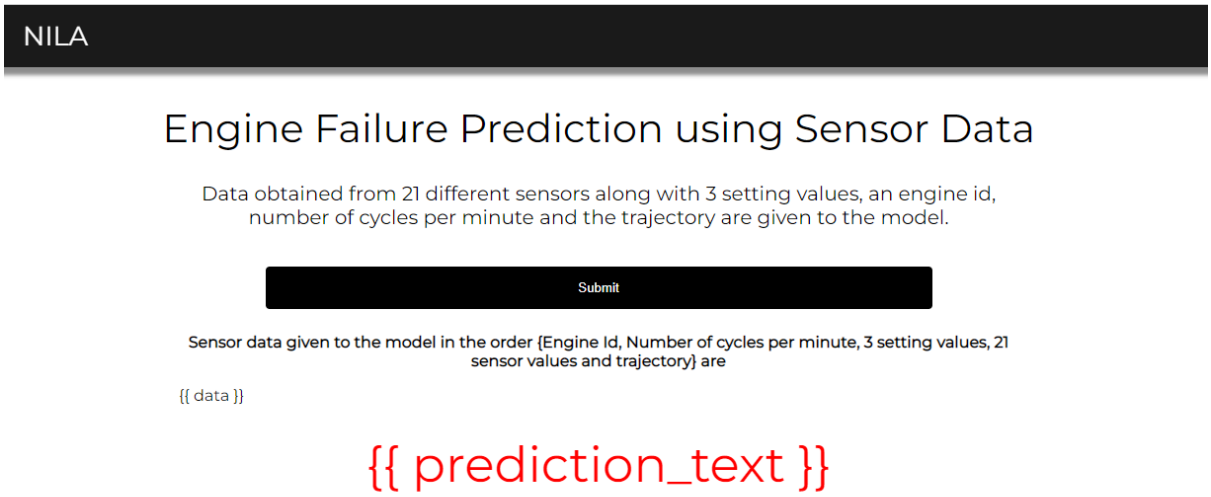
- Manual_predict.html
- Sensor_predict.html

and save them in the templates folder.

For more information regarding HTML: Link

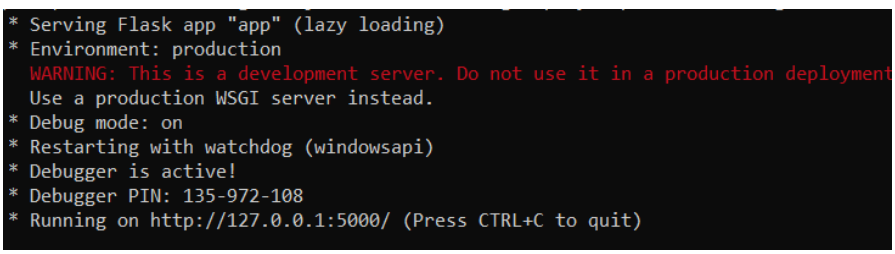
Let’s see how our home.html page looks like



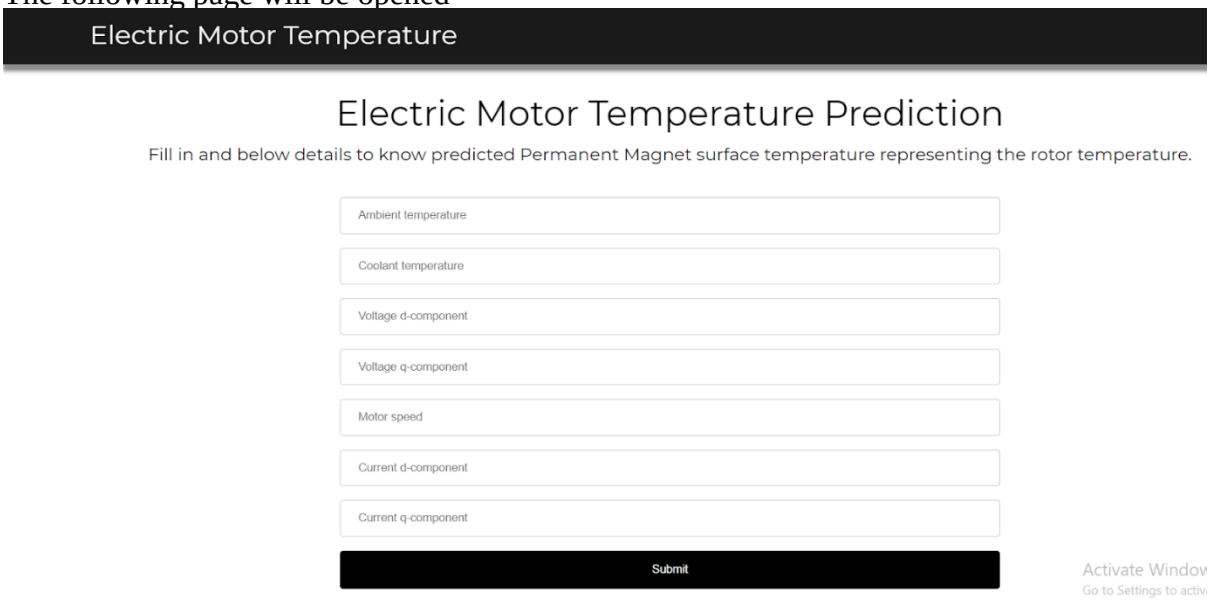


Activity 3: Run The Application

Open the anaconda prompt go to the project folder and in that go to the flask folder and run the python file by using the command “python app.py”



copy the HTTP link and paste it into a browser tab.
The following page will be opened



Enter the values and click on predict button, it will predict the temperature of an electric motor