Electric Motor Temperature Prediction Using Machine Learning

1 Introduction

1.1 Overview

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.

1.2 Purpose

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.

2 LITERATURE SURVEY

2.1 Existing problem

To predict the temperature of the rotor accurately, the mechanism of heat generation and conduction for the motor should be researched. Considering the complexity of the thermal characteristics on the actual motor work condition, the enery transfer paths inside the motor system were simplified.

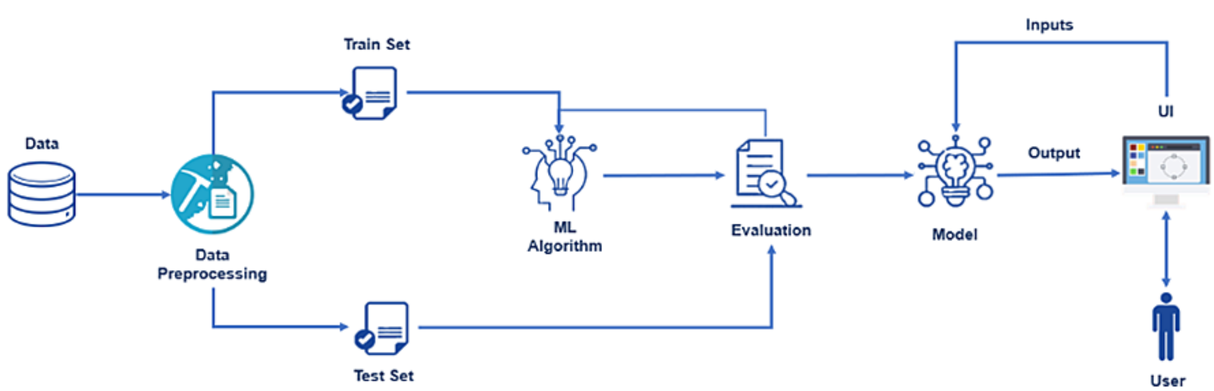
2.2 Proposed solution

- 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. o One Hot Encoding. o Feature Scaling. o Splitting Data into Train and Test.
- Model Building o Training and testing the model o Evaluation of Model(decision tree classification) • Application Building o Create an HTML file o Build a Python Code

3 THEORITICAL ANALYSIS

3.1 Block diagram



3.2 Hardware and software requirements of the project

Hardware

Processor : Processor Intel CORE i3 and above

Internet Connection : Existing telephone lines, Data card, Fiber net

RAM : 4 GB

Software designing

Operating System : Windows, Mac, Linux

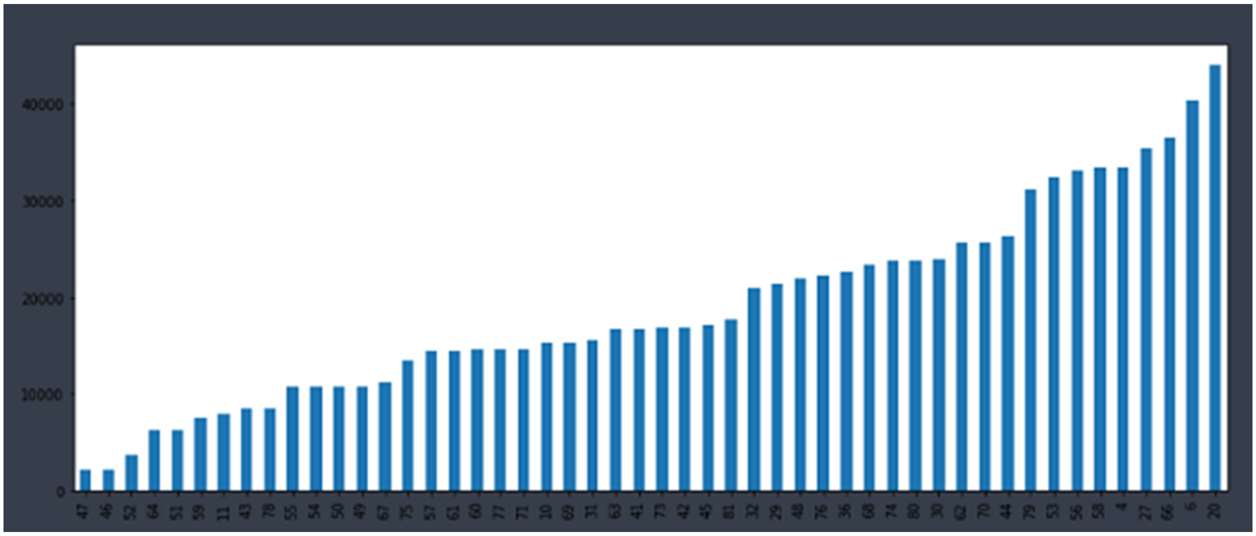
Language : R Programming – R-4.1.1

GUI : R Studio

4 EXPERIMENTAL INVESTIGATIONS

4.1 Analysis or the investigation made while working on the solution.

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.



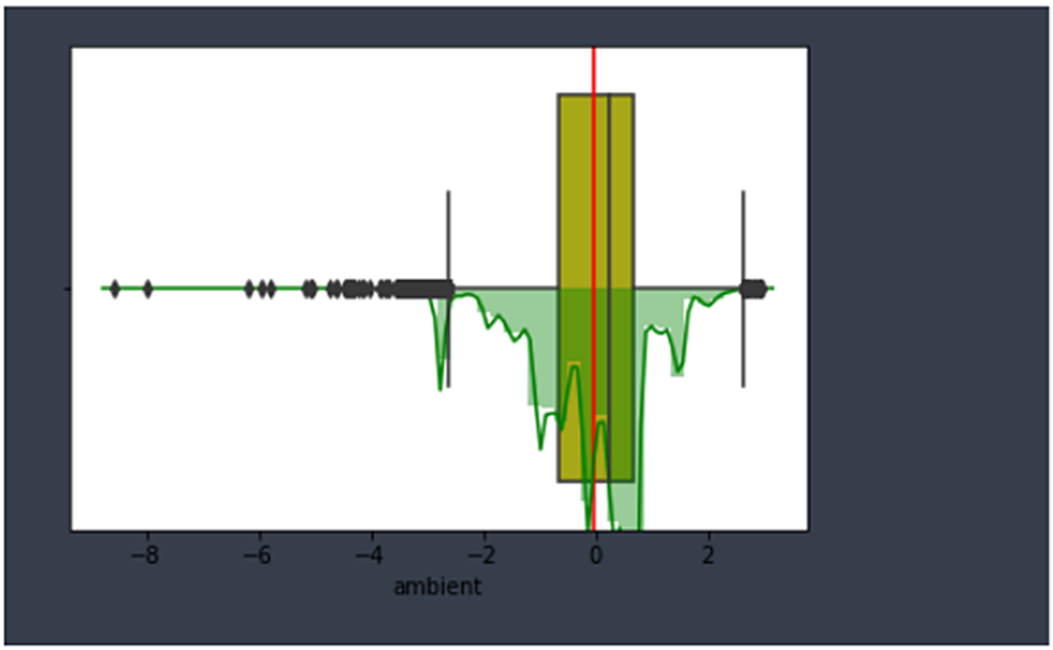
As we can see, session ids 66, 6, and 20 have the most number of measurements recorded.

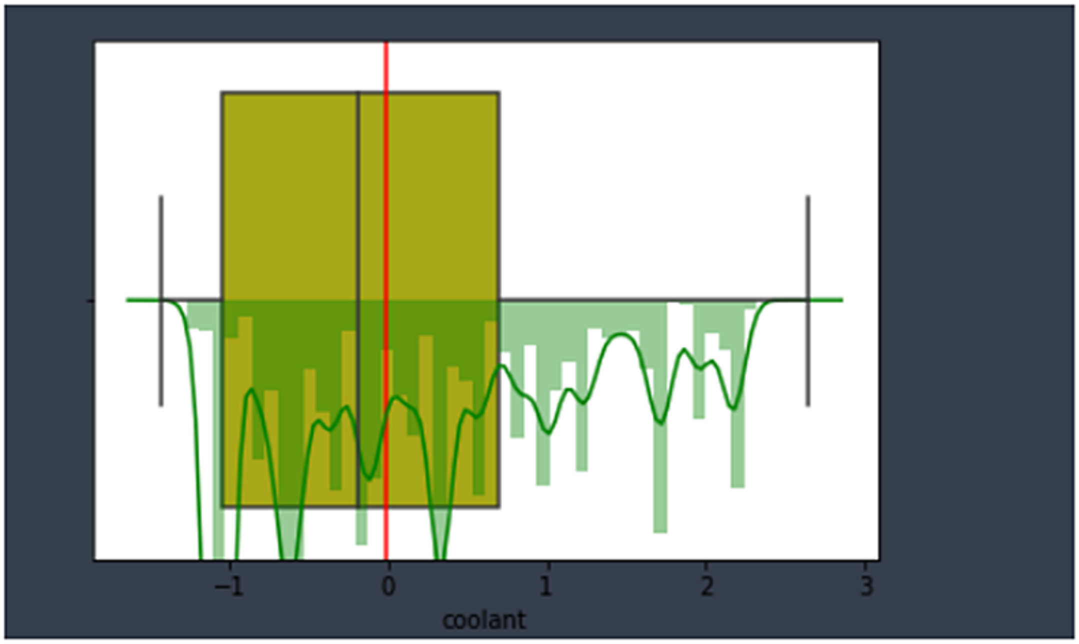
·Box plot:

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.



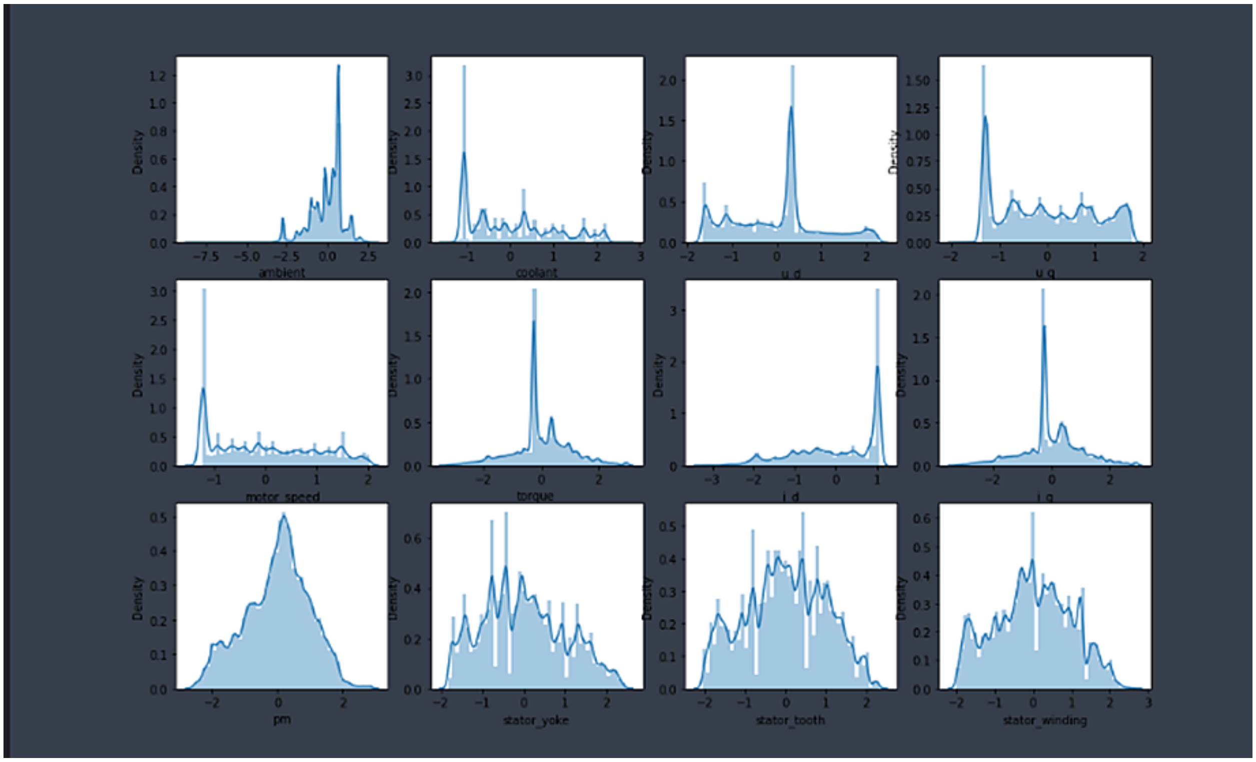


All features boxplots are plotted and the following conclusions are drawn.

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.

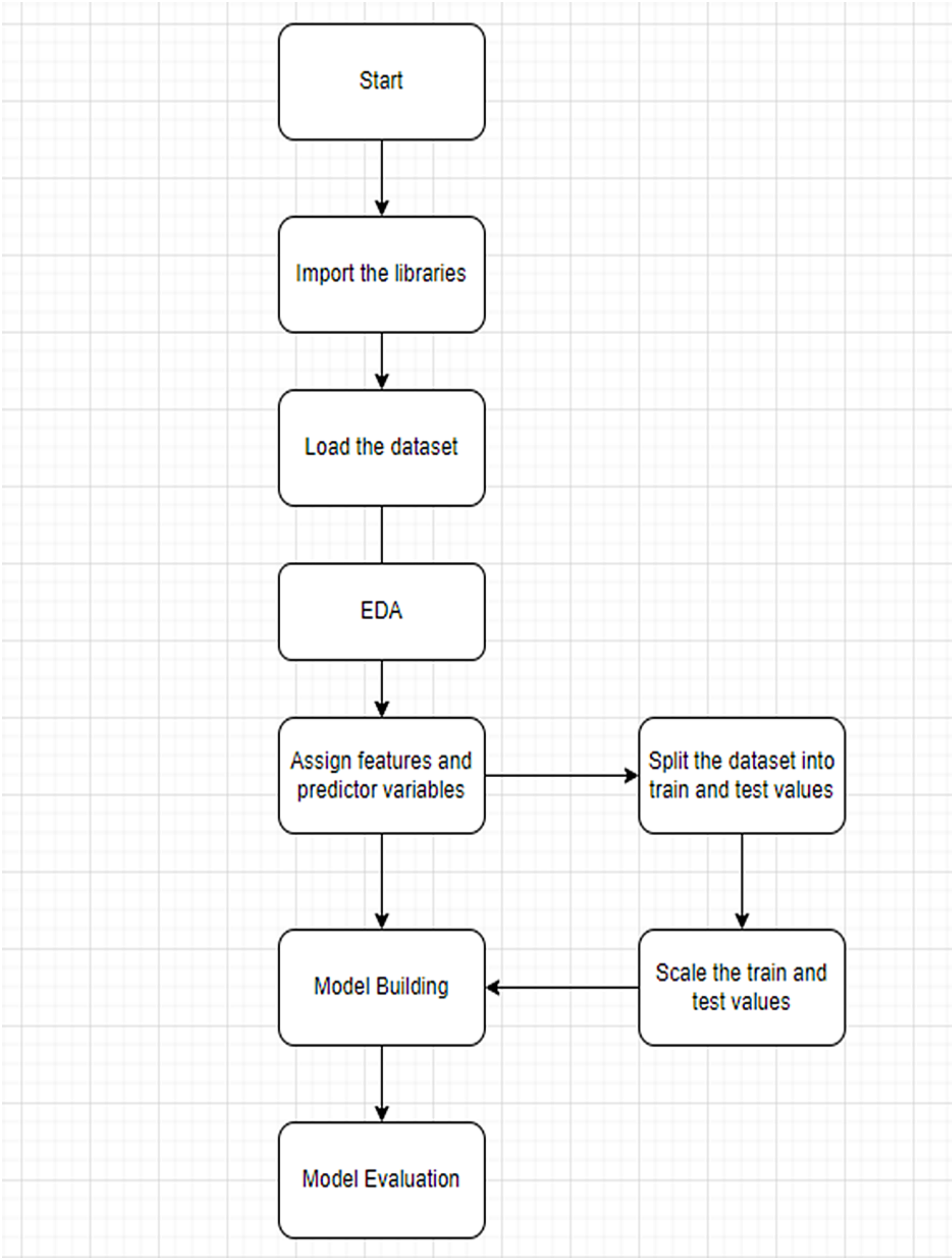
In [9]:

```
plt.figure(figsize = (15,10))

for i in range(len(df.columns)-1):
    plt.subplot(3,4,i+1)
    sns.distplot(df.iloc[:,i])

plt.show()
```



ambient
coolant
u_d
u_q
motor_speed
torque
i_d
i_q
pm
stator_yoke
stator_tooth
stator_winding
Density

5 FLOWCHART


Start
Import the libraries
Load the dataset
EDA
Assign features and predictor variables
Split the dataset into train and test values
Scale the train and test values
Model Building
Model Evaluation

6 RESULT

RESULT Final findings (Output) of the project along with screenshots.

Decision Tree model Accuracy

```
y_pred,r2score,mse = get_r2score_mse(tree_model,x_test,y_test)
display(r2score,mse)
```

```
Accuracy : 99.87%
Mean Squared Error : 0.0013198958720427526
```

7 Advantages of PMSM Motors

1. **Higher efficiency** than **Brushless DC Motors**
2. No torque ripple when motor is commutated
3. Higher torque and better performance
4. More reliable and less noisy, than other asynchronous motors
5. High performance in both high and low speed of operation
6. Low rotor inertia makes it easy to control
7. Efficient dissipation of heat
8. Reduced size of the motor

Disadvantages of PMSM Motors

- There is a risk of demagnetization of the poles which may be caused by a large armature current. Demagetizatiob cab also occurs due to excessive heating and also when the motor is an overload for a long period of time.
- Extra ampere Cannot be added to reduce the armature reaction.
- The magnetic field of the PMDC motor is preset at all times, even when the motor is not being used.
- The permanent magnet produces a high flux density as that an externally supplied shunt field does. Therefore, a PMDC motor has lower induced torque per ampere-turns of armature current the shunt than a shunt motor of the same rating.
- Permanent magnet motor solutions tend to need a higher initial cost than the use of AC induction motors so more difficult to starts up than AC induction motors.

8 Application

- Servo Mechanism in Automobiles: Servo mechanisms is a set of motors and motor controllers that produce motion at a higher energy level than the input applied. PMSM motors are the first choice of Motors to support such mechanism.This is because PMSM Motors are highly efficient, produce less noise and are resistant to wear and tear. One example is the servo brake that amplifies the force used by the driver on the brake pedals. Another example is the Servo Steering which is one step ahead of the regular power steering. This also makes use of a PMSM motor.
- Electric Vehicle Drivetrain: Baring a few Electric Vehicles which use BLDC motors, most OEMs are deploying AC motors to power the EV drivetrain.And PMSM is the preferred choice. Reasons being high power density and the availability of efficient PMSM motor control solutions.

9 CONCLUSION

The purpose oh this work was to predict the temperature of synchronous motor with a permanent magnet using machine learning methods. First of all, in this work a study of the subject area and a review of publications with research established in the laboratory of the university of Paderborn SDPM. With the help of the unified UML language, diagrams are designed that show how the implementation model can act as a part of the control system of the mechanism that works on SDPM. To implement this project, the method of machine learning Random Forest Regression is chosen. The implementation tool is python programming language with add-ins such as NumPy, Pandas, Scikit-learn, matplotlib, and seaborn. A comparison of the machine learning method and the means of implementation with analogues is also performed. Because of this this project, a study of data from the SDPM sensory data set is conducted, the predicted feature, which was the temperature of the stator winding, is determined, and this feature was predicted. After analysing the obtained forecasting results, we can conclude that the purpose and main objectives of this work are achieved and the trained model can be used to accurately estimate and predict the stator and rotor temperatures of a synchronous motor on permanent magnets.

10 Future Scope

New features are being introduced in the vehicles at an unprecedented rate. And motors, especially smart motor systems are at the core of such innovations.

The scope of this work was restricted to the application of simple machine learning algorithms to a PMSM due to data availability. Future work could however aim to generalize across different motor types which will require collecting data more representative of the diverse types of motors.

11 Biblography

References

1. I. Boldea, L. N. Tutelea, L. Parsa, and D. Dorrell, “Automotive electric propulsion systems with reduced or no permanent magnets: an overview,” *IEEE Transactions on Industrial Electronics*, vol. 61, no. 10, pp. 5696–5711, 2014.View at: Publisher Site | Google Scholar

2. H. Liang, Y. Chen, S. Liang, and C. Wang, “Fault detection of stator inter-turn short-circuit in PMSM on stator current and vibration signal,” *Applied Sciences*, vol. 8, no. 9, p. 1677, 2018.View at: Publisher Site | Google Scholar

3. D. Huang, W. Li, Y. Wang, and Z. Cao, “Influence of magnetic slot wedge on rotor losses and temperature field of PMSM,” *Electric Machines and Control.Magnetics*, vol. 20, pp. 60–66, 2016.View at: Google Scholar

4. Y. Chen, S. Liang, W. Li, H. Liang, and C. Wang, “Faults and diagnosis methods of permanent magnet synchronous motors: a review,” *Applied Sciences*, vol. 9, no. 10, p. 2116, 2019.View at: Publisher Site | Google Scholar

5. W. Li, Y. Chen, X. Li, and S. Liang, “Matching quality detection system of synchronizer ring and cone,” *Applied Sciences*, vol. 9, no. 17, p. 3622, 2019.View at: Publisher Site | Google Scholar

6. J. Dong, Y. Huang, L. Jin et al., “Thermal optimization of high-speed permanent motor,” *IEEE Transactions on Magnetics*, vol. 50, Article ID 7018504, 2014.View at: Publisher Site | Google Scholar

7. K.-S. Kim, B.-H. Lee, and H.-J. Kim, “Thermal analysis of outer rotor type IPMSM using thermal equivalent circuit,” in *Proceedings of the 15th International Conference on Electrical Machines and Systems*, pp. 1–4, Sapporo, Japan, October 2012.View at: Google Scholar

12 Appendix

12.1 Dataset and Importing Libraries

12.2 PreProsessing

```
In [1]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.svm import SVR
import pickle
```

```
In [2]:
df = pd.read_csv(".\dataset\pmsm_temperature_data.csv")
df
```

	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	s
0	-0.752143	-1.118446	0.327935	-1.297858	-1.222428	-0.250182	1.029572	-0.245860	-2.522071	-1.831422	-2.066143	-2
1	-0.771263	-1.117021	0.329665	-1.297686	-1.222429	-0.249133	1.029509	-0.245832	-2.522418	-1.830969	-2.064859	-2
2	-0.782892	-1.116681	0.332771	-1.301822	-1.222428	-0.249431	1.029448	-0.245818	-2.522673	-1.830400	-2.064073	-2
3	-0.780935	-1.116764	0.333700	-1.301852	-1.222430	-0.248636	1.032845	-0.246955	-2.521639	-1.830333	-2.063137	-2
4	-0.774043	-1.116775	0.335206	-1.303118	-1.222429	-0.248701	1.031807	-0.246610	-2.521900	-1.830498	-2.062795	-2
...	...	...	...	...	...	...	...	...	...	...	...	...
998065	-0.047497	0.341638	0.331475	-1.246114	-1.222428	-0.255640	1.029142	-0.245723	0.429853	1.018568	0.836084	0.
998066	-0.048839	0.320022	0.331701	-1.250655	-1.222437	-0.255640	1.029148	-0.245736	0.429751	1.013416	0.834438	0.
998067	-0.042350	0.307415	0.330946	-1.246852	-1.222430	-0.255640	1.029191	-0.245701	0.429439	1.002906	0.833936	0.
998068	-0.039433	0.302082	0.330987	-1.249505	-1.222432	-0.255640	1.029147	-0.245727	0.429558	0.999157	0.830504	0.
998069	-0.043803	0.312666	0.330830	-1.246590	-1.222431	-0.255640	1.029141	-0.245722	0.429166	0.987163	0.828046	0.

998070 rows × 13 columns

12.3 Training Model

```
In [30]:
x = df.drop(["pm","stator_yoke","stator_tooth","stator_winding","profile_id","torque"],axis = 1).values
x.shape
```

```
(998070, 7)
```

```
In [31]:
y = df.iloc[:,[8]].values
y
```

```
array([[-2.522071  ],
       [-2.5224178 ],
       [-2.5226731 ],
       ...,
       [ 0.4294391 ],
       [ 0.42955777],
       [ 0.4291662 ]])
```

```
In [32]:
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2)
```

```
In [33]:
scale = StandardScaler()
x_train = scale.fit_transform(x_train)
x_test = scale.transform(x_test)
```

Linear Regression Model

```
In [34]:
MLG_model = LinearRegression()
MLG_model.fit(x_train,y_train)
```

```
LinearRegression()
```

Decision Tree Rgeression Model

```
In [35]:
tree_model = DecisionTreeRegressor()
tree_model.fit(x_train,y_train)
```

```
DecisionTreeRegressor()
```

Random Forest Regression Model

```
In [36]:
rand_tree = RandomForestRegressor(n_estimators = 11)
rand_tree.fit(x_train,y_train)
```

```
C:\Users\Mohamed Adnan\AppData\Local\Temp\ipykernel_8\2741476015.py:2: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using ravel().
  rand_tree.fit(x_train,y_train)


RandomForestRegressor(n_estimators=11)
```

12.4 Evaluation Metrics

Evaluation Metrics

In [37]:
```
def get_r2score_mse(model,x_test,y_test):
    y_pred = model.predict(x_test)
    r2score = r2_score(y_test,y_pred)
    mse = mean_squared_error(y_test,y_pred)
    return y_pred,r2score,mse
```

In [38]:
```
def get_mse(y_test,y_pred):
    mse = mean_squared_error(y_test,y_pred)
    return mse
```

In [39]:
```
def display(r2score,mse):
    print("Accuracy : {:.2f}%".format(r2score*100))
    print("Mean Squared Error : {}".format(mse))
```

Linear model Accuracy

In [40]:
```
y_pred,r2score,mse = get_r2score_mse(MLG_model,x_test,y_test)
display(r2score,mse)
```

```
Accuracy : 47.08%
Mean Squared Error : 0.5242079356001519
```

Decision Tree model Accuracy

In [41]:
```
y_pred,r2score,mse = get_r2score_mse(tree_model,x_test,y_test)
display(r2score,mse)
```

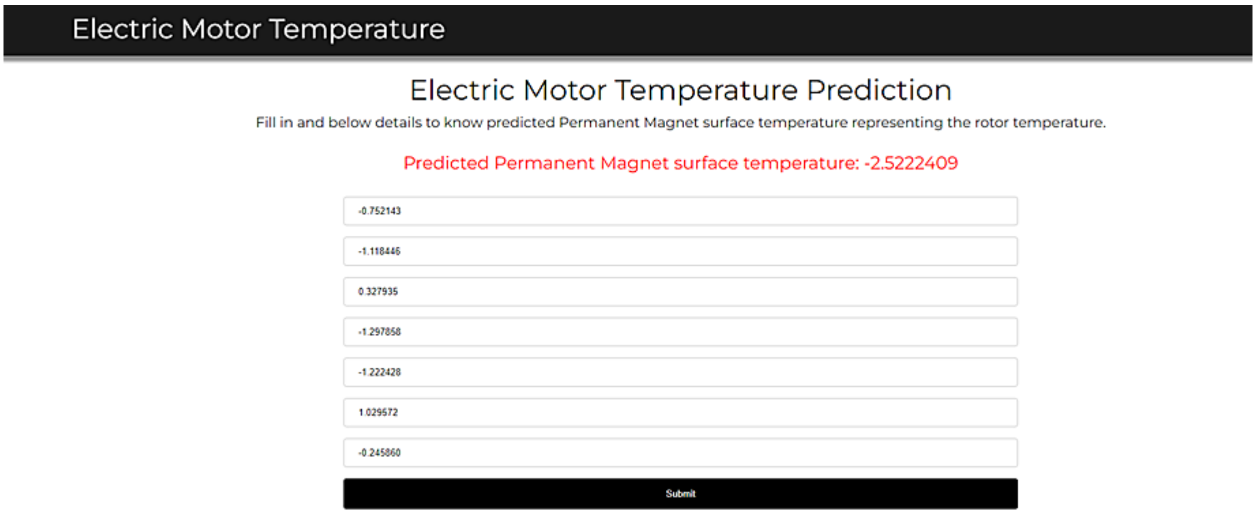
```
Accuracy : 97.05%
Mean Squared Error : 0.02922133103788543
```

Random Forest model Accuracy

In [42]:
```
y_pred,r2score,mse = get_r2score_mse(rand_tree,x_test,y_test)
display(r2score,mse)
```

```
Accuracy : 98.41%
Mean Squared Error : 0.015753181721280776
```

12.5 Flask Implementation


Electric Motor Temperature
Electric Motor Temperature Prediction
Fill in and below details to know predicted Permanent Magnet surface temperature representing the rotor temperature.
Predicted Permanent Magnet surface temperature: -2.5222409
-0.752143
-1.118446
0.327935
-1.297858
-1.222428
1.029572
-0.245860
Submit