ELECTRIC MOTOR TEMPERATURE PREDICTION USING IBM WATSON

Submitted By Team
Joseph Bejoy & Navaneeth
Union Christian College, Aluva, Kerala

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

The thermal loss and cooling modes of the permanent magnet synchronous motor (PMSM) directly affect its temperature rise. The heat loss of the PMSM mainly includes copper loss, iron loss and mechanical loss. The iron loss mainly depends on the stator's voltage, and the mechanical loss mainly depends on the rotor speed. Different from iron loss and mechanical loss, the copper loss of a permanent magnet motor stator directly affects the heating degree of the stator winding. On one hand, the heat of the stator winding is first transferred to the insulation. On the other hand, compared with the winding and core, the insulation in the motor is the material with the worst heat resistance among all materials of the motor. In the engineering field, the selection of an insulation grade of PMSM depends entirely on the temperature of the stator winding. When the temperature of the motor stator winding is too high, the insulation will be thermally aged, and decortication will even occur, which will seriously threaten the safe operation of the motor. In addition, if the permanent magnet motor winding heating cannot be effectively controlled, the heat of the stator winding will be further transmitted to the rotor side through the air gap, which will cause irreversible demagnetization of the permanent magnet. In conclusion, accurate evaluation and prediction of stator winding temperature is of great significance to the safety and reliability of permanent magnet motors.

2.2 Proposed solution

Model is proposed to predict the temperature of the motor based on several input parameters like ambient temperature, coolant temperature, voltage d-component, voltage q-component, motor speed, current d-component, current q-component. The dataset is trained on various machine learning algorithms and their performance is analyzed to find out which one would be the best to effectively predict temperature of the motor. The accuracies obtained from each algorithm is plotted to show the comparative analysis of each algorithm.

First data collection is done followed by data pre-processing to obtain a cleaned dataset with no null values or duplicate values for better training and higher accuracy. Then data visualization is performed which gives a clear idea about the dataset through the visualization graphs and makes it easier to identify the patterns, trends and outliers. After this, the dataset is split into training and testing datasets and fed into the various classification models to obtain the prediction. The confusion matrix along with the accuracies of the models are obtained to find out the most effective algorithm that could be used for the prediction. The model has also been trained using a custom input value to check for the accuracy.

3. HADWARE / SOFTWARE DESINING

- **Tools**

- For application development, the following Software Requirements are:

1. **Operating System:** Windows 7 and above
2. **Language**:Python, Html
3. **Tools**: Microsoft Excel (Optional).
4. **Technologies used**:Flask application

- **Software requirements**

1. **Operating System**:Any OS with clients to access the internet
2. **Network**:Wi-Fi Internet or cellular Network
3. **Visio Studio**:Create and design Data Flow and Block Diagram
4. **GitHub**:Versioning Control
5. **Google Chrome**:Medium to display and run the html cod

- **Hardware Requirements**

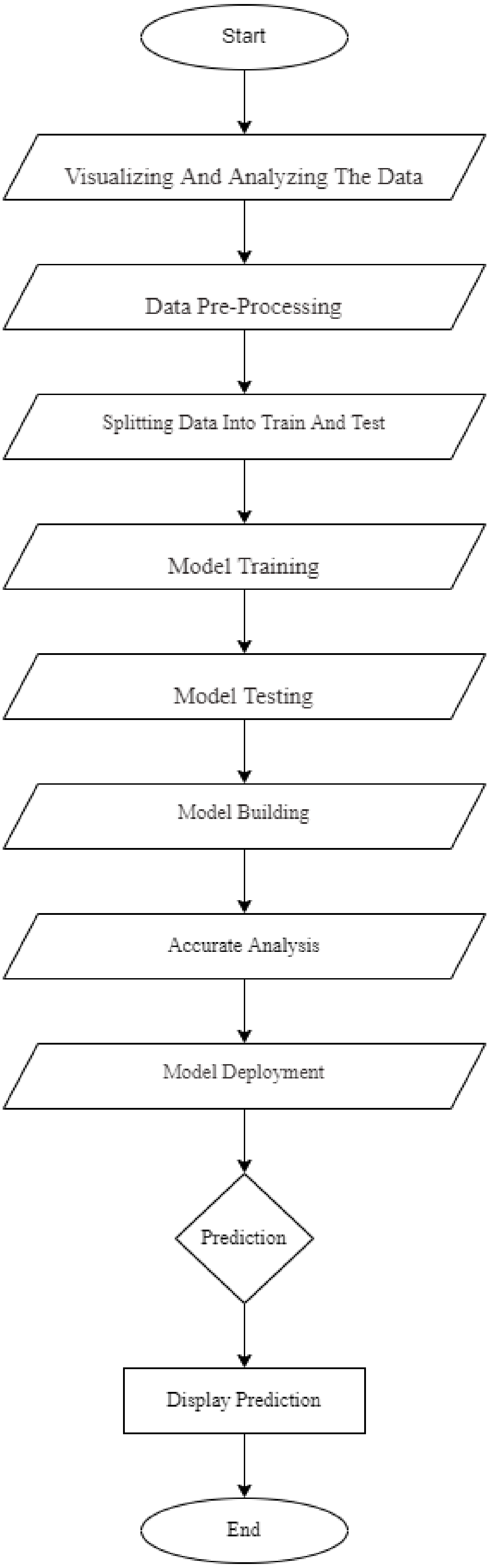
For application development, the following Software Requirements are:

1. **Processor**: Intel or high.
2. **RAM**: 1024 MB.
3. **Space on disk**: minimum 100mb.
4. **For running the application**: Device: Any device that can access the internet.
5. **Minimum space to execute**: 20 MB

4. EXPERIMENTAL INVESTIGATIONS

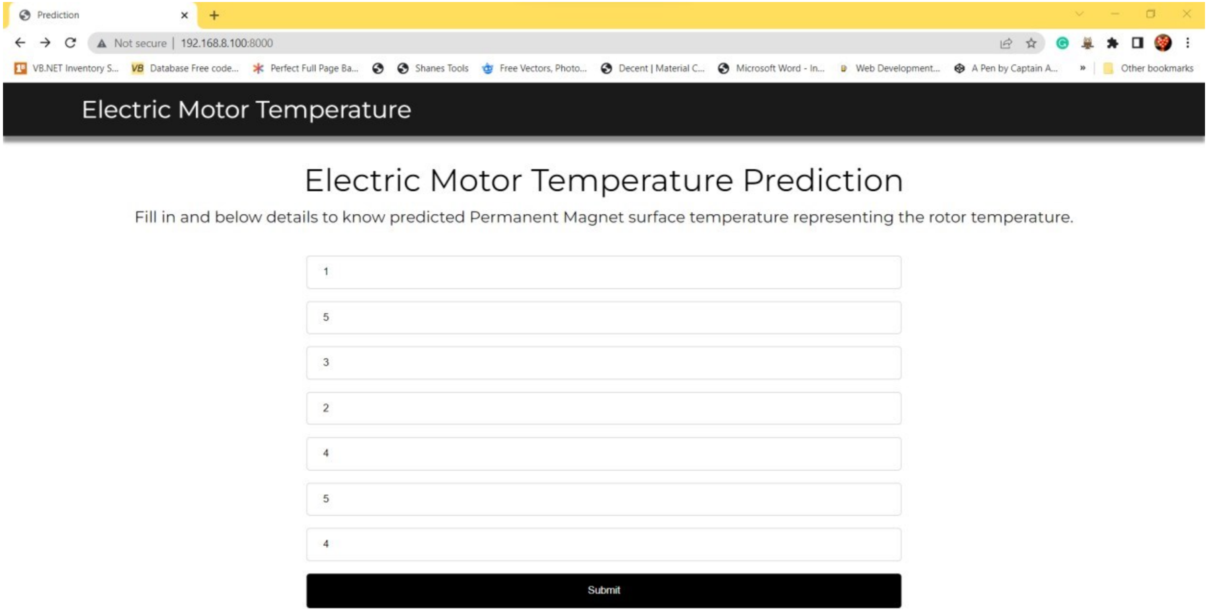
The traditional motor temperature prediction model mainly uses the finite element method. The specific method is to simulate the transient temperature by using the finite element method, establish the temperature field, and then predict the motor temperature. This method can only be used to calculate and process the current linear data, but it cannot deal with a large number of nonlinear 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.
The proposed solution is based on 7 variables. That are ambient temperature (coolant temperature voltage d-component, voltage q-component, motor speed, current d-component, current q-component.

5. Flow Chart


Start
Visualizing And Analyzing The Data
Data Pre-Processing
Splitting Data Into Train And Test
Model Training
Model Testing
Model Building
Accurate Analysis
Model Deployment
Prediction
Display Prediction
End

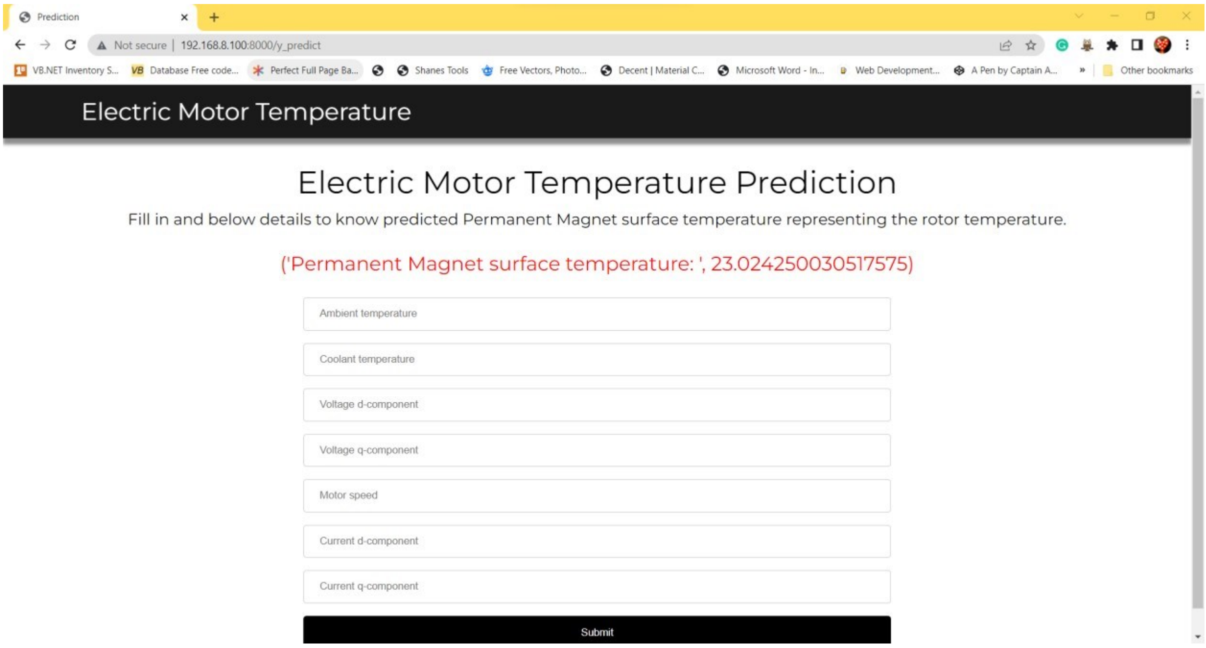
6. RESULT

6.1 Input Page



All the 7 fields as shown in above picture are independent variables used to predict the temperature

6.2 Output Page



After giving all the inputs the predictions is shown in red color in the picture above which says that Permanent Magnet Surface temperature

7. ADVANTAGES & DISADVANTAGES

❖ **Advantages**

- Increased accuracy for effective motor temperature prediction.
- Handles roughest(enormous) amount of data using random forest algorithm and feature selection.
- Cost effective
- This suggestion is promising as data modelling and analysis tools, e.g., data mining, have the potential to generate a knowledge-rich environment which can help to significantly improve the quality of mechanical decisions.

❖ **Disadvantages**

- Data mining techniques does not help to provide effective decision making.

8. CONCLUSION

In this paper, test bench data from a permanent magnet synchronous motor (PMSM) was used to estimate the temperatures of the rotor, the stator yoke, the stator tooth and the stator winding in the PMSM by applying machine learning techniques. Linear regression (LR), k nearest neighbours (kNN), random forest (RF) and decision tree (DT) algorithms were employed on the dataset in two different experiments with an objective of determining the extent to which these temperatures can be accurately estimated. .

9. FUTURE SCOPE

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.

10. BIBILOGRAPHY

1. Zhang, H.T.; Dou, M.F.; Deng, J. Loss-Minimization Strategy of Nonsinusoidal Back EMF PMSM in Multiple Synchronous Reference Frames
2. Chen, S.A.; Jiang, X.D.; Yao, M.; Jiang, S.M.; Chen, J.; Wang, Y.X. A dual vibration reduction structure-based self-powered active suspension system with PMSM-ball screw actuator via an improved H-2/H-infinity control.
3. Wallscheid, O.; Specht, A.; Bcker, J. Observing the Permanent-Magnet Temperature of Synchronous Motors Based on Electrical Fundamental Wave Model Quantities.

11. APPENDIX

❖ **Source code of python in Jupyter notebook**

```
# Import Libraries

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

# Read The Data Set

import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3
def__iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your
credentials.
# You might want to remove those credentials before you share the notebook.
client_6e31851029094998aaf1215b90f37c02 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='wDqvAhBRyREvo9TBi3_2jf46q6GmifZVzeRxUrZ_cv1D',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')
```

```
body =
client_6e31851029094998aaf1215b90f37c02.get_object(Bucket='electricmotortemperaturepr
edictio-donotdelete-pr-jnjxnwzabu5xsf',Key='measures_v2.csv')['Body']
# add missing___iter___method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter___= types.MethodType( __iter__, body )

df = pd.read_csv(body)
df.head()

# Uni-Variate Analysis
#Bar Graph
plt.figure(figsize=(15,6))
df['profile_id'].value_counts().sort_values().plot(kind='bar')
df.columns

#Plotting Distribution and Boxplot for all the features to check for skewness

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()

# Multi-Variate Analysis
fig,axes = plt.subplots(2,4, figsize=(20,5),sharey=True)
sns.scatterplot(df['ambient'],df['pm'],ax=axes[0][0])
sns.scatterplot(df['coolant'],df['pm'],ax=axes[0][1])
sns.scatterplot(df['motor_speed'],df['pm'],ax=axes[0][2])
sns.scatterplot(df['i_d'],df['pm'],ax=axes[0][3])
sns.scatterplot(df['u_q'],df['pm'],ax=axes[1][0])
sns.scatterplot(df['u_d'],df['pm'],ax=axes[1][1])
sns.scatterplot(df['i_q'],df['pm'],ax=axes[1][2])

plt.figure(figsize=(14,7))
sns.heatmap(df.corr(),annot=True);

#For a random measurement, we can try to compare the temperatures of the 3 stator
components.

plt.figure(figsize=(20,5))
df[df['profile_id'] == 20]['stator_yoke'].plot(label = 'stator yoke')
df[df['profile_id'] == 20]['stator_tooth'].plot(label = 'stator tooth')
df[df['profile_id'] == 20]['stator_winding'].plot(label = 'stator winding')
plt.legend();
```

```
df = df[(df['profile_id'] != 65) & (df['profile_id'] != 72)]
df_test = df[(df['profile_id'] == 65) | (df['profile_id'] == 72)]

df.drop('profile_id',axis = 1, inplace=True)
df_test.drop('profile_id',axis = 1,inplace=True)

# Descriptive Analysis

df.info()
df.describe()

# Data Pre-Processing
df.head()

#Drop Unwanted Features
df.drop(['stator_yoke','stator_tooth','stator_winding','torque'],axis = 1)

#Handling Missing Values
df.isnull().sum()

#Normalizing The Values

from sklearn.preprocessing import MinMaxScaler
X = df.drop(['pm','stator_yoke','stator_tooth','stator_winding','torque'],axis = 1)
X_df_test = df_test.drop(['pm','stator_yoke','stator_tooth','stator_winding','torque'],axis = 1)
X

names = X.columns
mm = MinMaxScaler()
X = mm.fit_transform(X)
y = df['pm']

X=pd.DataFrame(X,columns = names)
X.shape
y.shape

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

#Splitting Data Into Train And Test
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)

# Model Building
from sklearn.linear_model import LinearRegression
```

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from xgboost import XGBClassifier


#Liner Regression
lr=LinearRegression()

#Decision Tree Model
dr=DecisionTreeRegressor()

#Random Forest Model
rf=RandomForestRegressor()
xgb=xgboost.XGBRegressor()

lr.fit(X_train,y_train)

dr.fit(X_train,y_train)

xgb.fit(X_train,y_train)


# Compare The Model

from sklearn import metrics

p1=lr.predict(X_test)
p1

p2=dr.predict(X_test)
p2

p3=xgb.predict(X_test)
p3

print(metrics.r2_score(y_test,p1))
print(metrics.r2_score(y_test,p3))

#Evaluating Performance Of The Model

from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test,p1))

# Save The Model
import joblib
```

```
joblib.dump(dr,"model.save")

# IBM Machine learning
pip install ibm_watson_machine_learning
import ibm_watson_machine_learning

# Authenticating and setting up the space
from ibm_watson_machine_learning import APIClient
wml_credentials = {
                 "url": "https://us-south.ml.cloud.ibm.com",
                 "apikey":"seSzSNtAIEtMzm3WtkJQvds67-jNUBoHCgFladiYQeoq"
                }

wml_client = APIClient(wml_credentials)
wml_client.spaces.list()

SPACE_ID = "e4104233-d541-441a-88f6-4f90036df939"
#spaceId from space in model

wml_client.set.default_space(SPACE_ID) #setting default space
MODEL_NAME = "Electric"
DEPLOYMENT_NAME = "electric_deploy"
BEST_MODEL = lr

#setting python version
software_spec_uid = wml_client.software_specifications.get_id_by_name("default_py3.8")
#setup model meta- model properties
model_props = {
wml_client.repository.ModelMetaNames.NAME:MODEL_NAME,
wml_client.repository.ModelMetaNames.TYPE:"scikit-learn_0.23",
  wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid
}

#saving the model
model_details = wml_client.repository.store_model(
    model = BEST_MODEL,
    meta_props = model_props,
    training_data = X_train,
    training_target = y_train
 )

software_spec_uid
model_uid = wml_client.repository.get_model_id(model_details)
model_uid

# Deploying model
```

```
#set meta
deployment_props = {
   wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
   wml_client.deployments.ConfigurationMetaNames.ONLINE:{}
}

#the actual deployment
deployment =
wml_client.deployments.create(artifact_uid=model_uid,meta_props=deployment_props)

deployment_uid =wml_client.deployments.get_uid(deployment)
deployment_uid

model_id
```

- **Flask Application coding**

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

import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM
Cloud account.
API_KEY = "5oxeVZLWX5gnKlvAGUbMqVHvuzwKbJEk-LUefGKA31iv"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
```