

ELECTRIC MOTOR TEMPERATURE
PREDICTION USING IBM WATSON

SUBMITTED BY-

JIGYASU PANT
AMAN LAKHMANI
AYUSH SINGH
ATHARV ARYA

1. INTRODUCTION:-

Overview:-

Electric motor is a device which is used to transform mechanical energy into electrical energy. The principle of the electric motor involves the process of interaction between the electric and magnetic fields. As far as temperature in electric motor is concerned, it is undoubtedly an important contributory factor to the performance of the motor. Elevated temperatures may cause insulation in the winding wires, which may result in short circuits or causing the motor winding to burn out. Moreover, certain important properties such as magnetic properties of iron may be degraded by excessive heat. Owing to these problems associated with temperature in electric motors, the proposed work performs electric motor temperature prediction using IBM Watson.

Purpose:-

Considering temperature as the primary factor affecting the performance of an electric motor, it is of utmost importance that an efficient mechanism is proposed for this purpose. With an increase in the motor temperature, the resistance increases and the torque constant and voltage constant decreases. Motor winding resistance is the key reason behind heat generation within the motor. Using machine learning proves to be significant as firstly, performing testing at initial stage results in prevention of damage to the motors, thereby identifying the perfect temperature intervals for the electric motors to work. Moreover the appropriate machine learning models assist in observing the areas where modification pertaining to the motors is required.

2. LITERATURE SURVEY:-

2.1 Existing Problem:-

Overheating is known to cause excessive damage to electric motors and is usually caused by elevated temperatures in the environment where the motor operates. Around 55 % of insulating failures in electric motors are associated with overheating. Each 10 degree Celsius rise in the temperature of the motor causes the insulation life to decrease to half. Furthermore, low resistance is the most common reason behind failure in electric motors. If the separation between the motor windings is not sufficient, it eventually paves the path to several problems such as leakages, short circuits and finally, failure of the electric motor.

2.2 Proposed solution:-

In the proposed work, prediction of the motor temperature is enabled considering the significance of temperature monitoring in electric motors. A machine learning model is prepared based on various parameters as input like coolant temperature, ambient temperature, voltage q-component, motor speed, current d-component and current q-component. The dataset taken into consideration is trained as per several machine learning algorithms. In the steps involved, the first and foremost is data collection. Thereafter, the second step is data visualisation wherein the graphic representation of data is performed, thus allowing us to analyse and obtain information about the patterns, outliers etc. This is followed by data pre-processing to obtain a clean dataset. This is followed by splitting of the dataset into training and testing models to get the prediction done. The most effective algorithm is found by performing a comparative analysis between their accuracies. Then, a web application is built for integration with the model built. By creating a web application, a user is allowed to enter the values for prediction. At the final stage, the model is deployed on IBM cloud.

3. THEORETICAL ANALYSIS:-

HADWARE / SOFTWARE DESIGNING

Software Requirements:-

- Operating System: Windows 10
- Platforms Used: -Google Colab, Flask, IBM Cloud, GitHub
- Languages Used: - Python, HTML

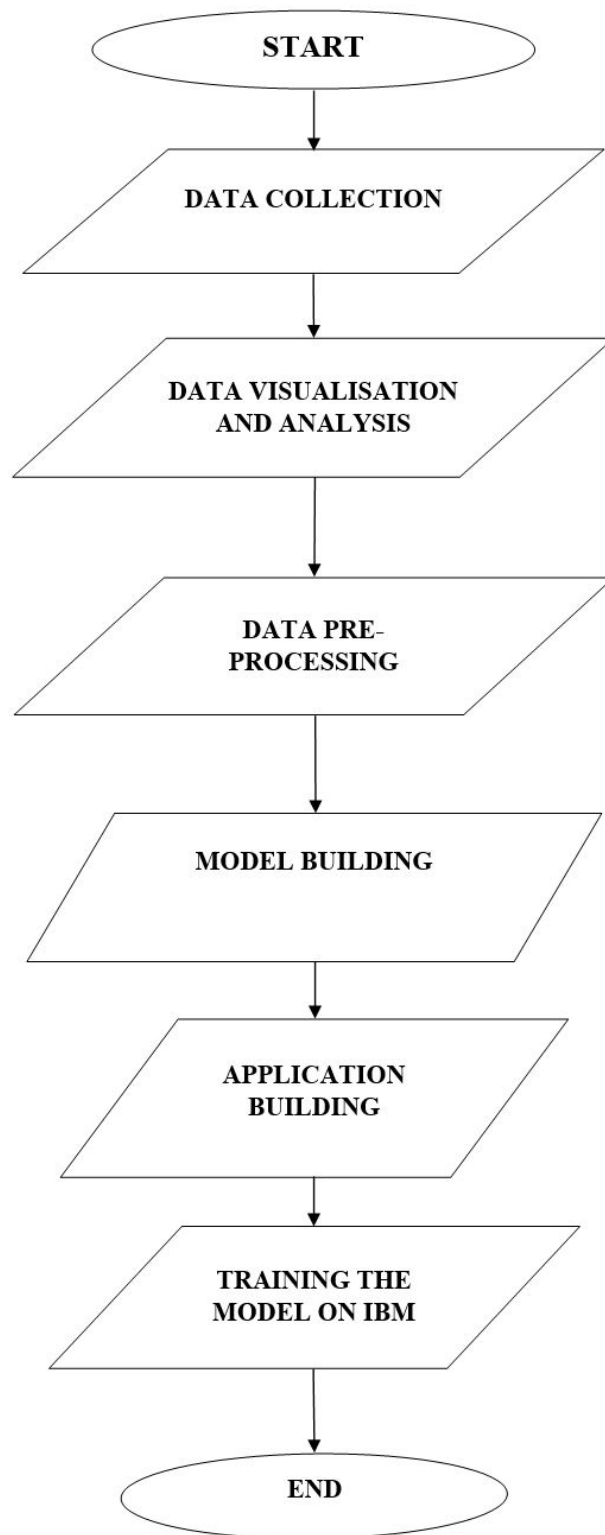
Hardware Requirements: -

- Processor: Intel Core (i5).
- RAM: 8 GB.
- Space on disk: minimum 100 MB.
- Minimum space required for execution: 20 MB
- Device: - Any device with internet access

4. EXPERIMENTAL INVESTIGATIONS:-

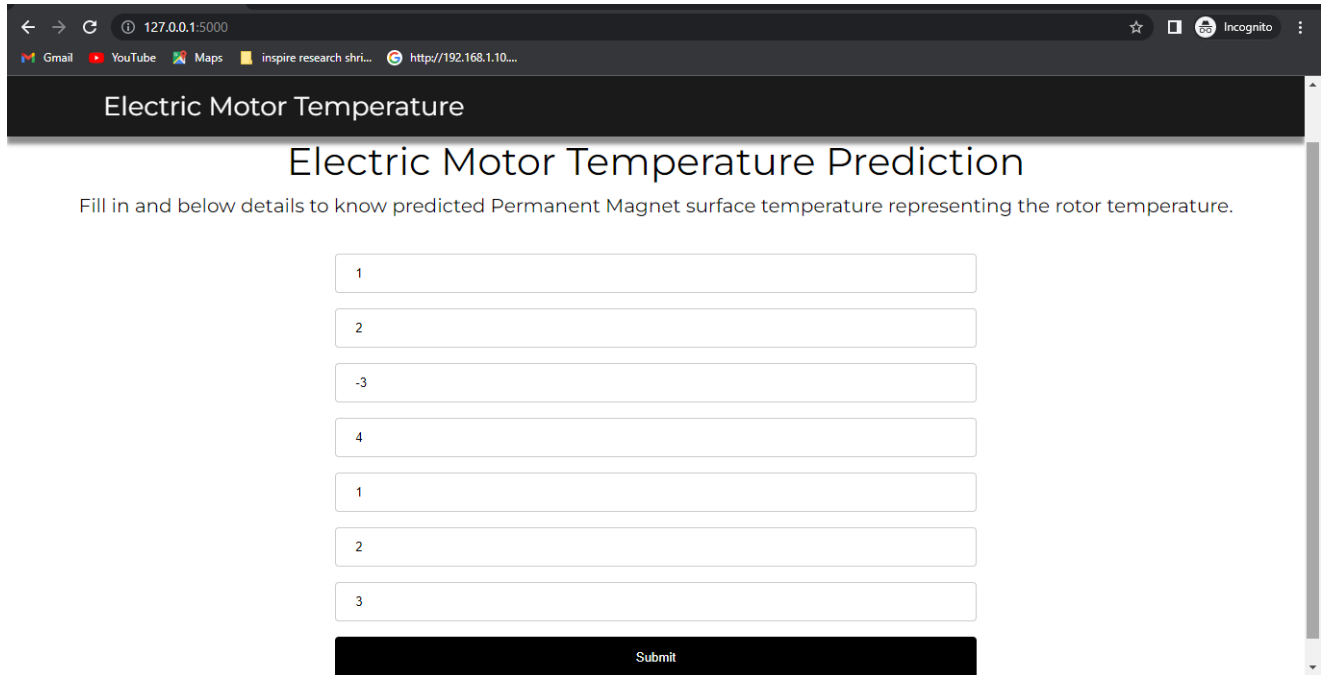
It is a must that we must be capable of dealing with huge as well as non linear data as far as temperature prediction of motors is concerned. In conventional methods, the process of temperature prediction is linear. Thus, the proposed work proves to be a better alternative over the traditional methods. Searching and identifying the best machine learning algorithm further helps in ensuring accuracy in temperature prediction and monitoring. Making use of input parameters in the dataset such as ambient temperature, coolant temperature voltage d-component, voltage q-component etc. helps us in realising the process of temperature prediction in a holistic manner.

5. FLOWCHART: -



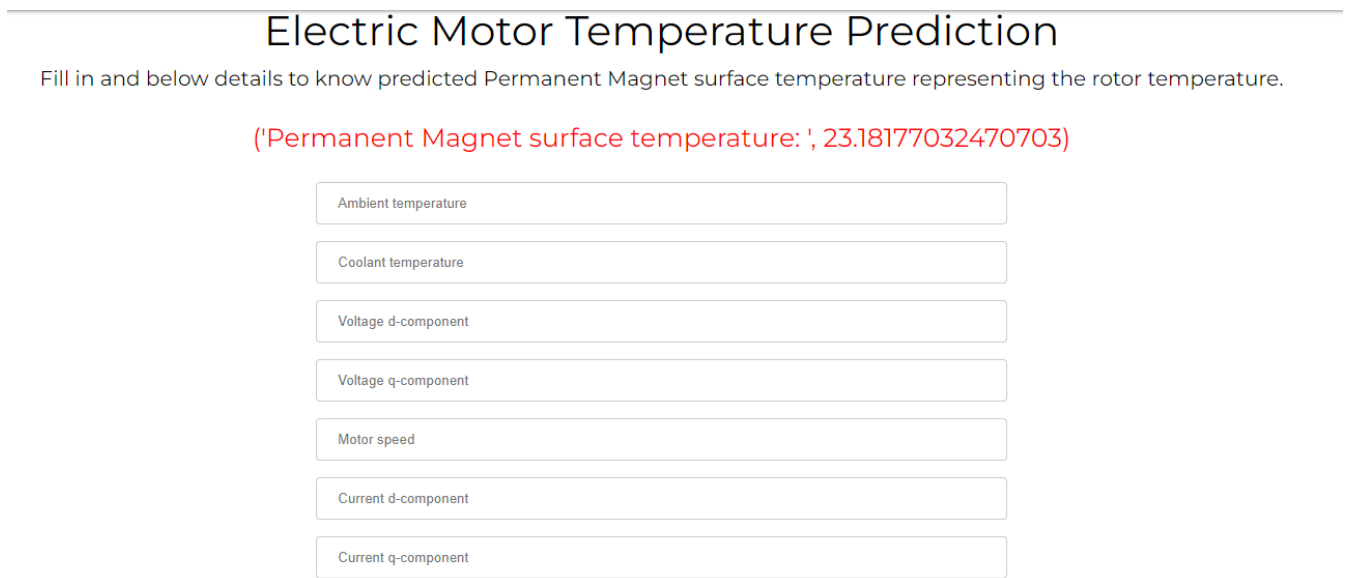
6. RESULTS:-

SCREENSHOT WITH INPUT VALUES: -



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000'. The page title is 'Electric Motor Temperature'. The main heading is 'Electric Motor Temperature Prediction'. Below the heading, there is a instruction: 'Fill in and below details to know predicted Permanent Magnet surface temperature representing the rotor temperature.' There are seven input fields, each containing a number: 1, 2, -3, 4, 1, 2, and 3. At the bottom, there is a black 'Submit' button.

SCREENSHOT OF OUTPUT: -



The screenshot shows the same web application interface as the previous one, but with the output displayed. The heading 'Electric Motor Temperature Prediction' and the instruction 'Fill in and below details to know predicted Permanent Magnet surface temperature representing the rotor temperature.' are still present. Below the instruction, the output is displayed in red text: '('Permanent Magnet surface temperature: ', 23.18177032470703)'. There are seven input fields, each containing a label: 'Ambient temperature', 'Coolant temperature', 'Voltage d-component', 'Voltage q-component', 'Motor speed', 'Current d-component', and 'Current q-component'.

7. ADVANTAGES AND DISADVANTAGES: -

Advantages:-

1. Optimised models are prepared which have high accuracy of prediction due to continuous learning.
2. Patterns and outliers not easily identifiable can be simply found as machine learning is used.
3. Huge as well as non linear data can be easily dealt with.
4. Human attention or interference is not constantly needed.

Disadvantages:-

1. Good amount of time may be required for allowing the algorithms to learn.
2. Vulnerability to errors is also a problem.

8. APPLICATIONS:-

1. As we do temperature prediction in motors, the same can be applicable to temperature prediction for climate.
2. Analysis of engine temperatures for vehicles may take place with the help of the proposed work.
3. Evaluation of the thermal environment of underground mines is also an area of application.

9. CONCLUSION:-

Hence, we have successfully developed the machine learning models for temperature prediction. Machine learning techniques made the process efficient significantly. Important algorithms such as linear regression, random forest, decision tree algorithms, k nearest neighbours are used and compared for choosing the most effective out of them.

10.FUTURE SCOPE:-

The future scope of the project involves its utility in several areas such as mining, vehicles, room temperature, climate prediction etc. wherever determination of temperature is a crucial factor. Moreover, in the manufacturing sector, wherever monitoring and prediction of temperature is a necessity for the products or devices, the scope of the project can extend to that application.

11.BIBLIOGRAPHY:-

1. <https://www.plantengineering.com/articles/when-it-comes-to-motors-how-hot-is-hot/>
2. <https://www.neuraldesigner.com/learning/examples/electric-motor-temperature-digital-twin>
3. <https://circuitglobe.com/electric-motor.html>
4. <https://newbedev.com/anaconda-prompt-crashes-as-soon-as-i-activate-tensorflow-env#:~:text=Anaconda%20prompt%20crashes%20as%20soon%20as%20I%20activate,k%20eras%20C%20the%20tensorflow%20and%20tensorboard%20versions%20gets%20changed>

APPENDIX: -

A. SOURCE CODE: -

```
import numpy as np
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.gridspec as gridspec
import os
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import svm
import tensorflow as tf
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
data.describe()
plt.figure(figsize=(15,6))
data['profile_id'].value_counts().sort_values().plot(kind='bar')
for i in data.columns:
    sns.distplot(data[i],color='g')
    sns.boxplot(data[i],color='y')
    plt.vlines(data[i].mean(),ymin=-1,ymax=1,color='r')
plt.show()
fig,axes=plt.subplots(2,4,figsize=(20,5),sharey=True)
sns.scatterplot(data['ambient'],data['pm'],ax=axes[0][0])
sns.scatterplot(data['coolant'],data['pm'],ax=axes[0][1])
sns.scatterplot(data['motor_speed'],data['pm'],ax=axes[0][2])
sns.scatterplot(data['i_d'],data['pm'],ax=axes[0][3])
sns.scatterplot(data['u_q'],data['pm'],ax=axes[1][0])
sns.scatterplot(data['u_d'],data['pm'],ax=axes[1][1])
sns.scatterplot(data['i_q'],data['pm'],ax=axes[1][2])
plt.figure(figsize=(14,7))
sns.heatmap(data.corr(),annot=True)
plt.figure(figsize=(20,5))
data[data['profile_id']==20]['stator_yoke'].plot(label='stator yoke')
data[data['profile_id']==20]['stator_tooth'].plot(label='stator tooth')
data[data['profile_id']==20]['stator_winding'].plot(label='stator winding')
plt.legend()
data = data[(data['profile_id'] != 65) & (data['profile_id'] != 72)]
data_test = data[(data['profile_id'] == 65) | (data['profile_id'] == 72)]
```

```

data.drop('profile_id',axis = 1, inplace=True)
data_test.drop('profile_id',axis = 1,inplace=True)
data.isnull().sum()
data.info()
# Data Pre-Processing
data.head()
#Drop Unwanted Features
data.drop(['stator_yoke','stator_tooth','stator_winding','torque'],axis = 1)
#Handling Missing Values
data.isnull().sum()

```

#Normalizing The Values

```

from sklearn.preprocessing import MinMaxScaler
X = data.drop(['pm','stator_yoke','stator_tooth','stator_winding','torque'],axis = 1)
X_data_test=data_test.drop(['pm','stator_yoke','stator_tooth','stator_winding','torque'],axis= 1)
X
names = X.columns
scale = StandardScaler()
X = scale.fit_transform(X)
y = data['pm']
X=pd.DataFrame(X,columns = names)
X.shape
y.shape
import joblib
joblib.dump(scale,'transform.save')
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=3)
scale = StandardScaler()
X_train = scale.fit_transform(X_train)
X_test = scale.transform(X_test)
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn import metrics
lr=LinearRegression()
dr=DecisionTreeRegressor()
lr.fit(X_train,y_train)
dr.fit(X_train,y_train)
Y_pred = lr.predict(X_test)
LogReg = round(lr.score(X_test, y_test), 2)
mae_lr = round(metrics.mean_absolute_error(y_test, Y_pred), 4)
mse_lr = round(metrics.mean_squared_error(y_test, Y_pred), 4)
print(mae_lr)
print(mse_lr)

```

```
import joblib
joblib.dump(dr,'model.save')
```

#Deployment (IBM MACHINE LEARNING)

```
!pip install -U ibm-watson-machine-learning
from ibm_watson_machine_learning import APIClient
import json
import numpy as np
```

#Authenticate and Set Space

```
wml_credentials={"apikey":"qZ1L8puFkx5OiafXhv2t5HHXK-Roa3E2c7FbwRrnza6n",
"url":"https://us-south.ml.cloud.ibm.com"}
wml_client=APIClient(wml_credentials)
wml_client.spaces.list()
model_details
model_id=wml_client.repository.get_model_id(model_details)
model_id
deployment_props={
wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
}
deployment=wml_client.deployments.create(
artifact_uid=model_id,
meta_props=deployment_props
)
```

#FLASK APPLICATION CODE

```
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')
```

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

```
@app.route('/y_predict',methods=['POST','GET'])
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)
print(pred)

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

```

```

if __name__ == "__main__":
    app.run(debug=False)

```

#FLASK APPLICATION CODE FOR IBM MACHINE LEARNING

```

# -*- coding: utf-8 -*-
"""

```

Created on Sun Jun 5 11:50:45 2022

```

@author: ayush
"""

```

```

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

```

```

import requests

```

```

# NOTE: you must manually set API_KEY below using information retrieved from your IBM
Cloud account.

```

```

API_KEY = "qZ1L8puFkx5OiafXhv2t5HHXK-Roa3E2c7FbwRrnza6n"
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}
app = Flask(__name__)
#model = joblib.load("model.save")
trans=joblib.load('transform.save')

```

```

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

@app.route('/y_predict',methods=['POST','GET'])
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)
    #print(pred)
    payload_scoring = {"input_data": [{"fields":["f0","f1","f2","f3","f4","f5","f6"]},
    "values":[[1,2,3,2,1,4,5]]]}
    response_scoring= requests.post ('https://us-
    south.ml.cloud.ibm.com/ml/v4/deployments/03741e21-6e84-48b6-8fe7-
    520288d195ad/predictions?version=2022-06-09', json=payload_scoring,
    headers={'Authorization': 'Bearer ' + mltoken})
    print("Scoring response")
    print(response_scoring.json())
    pred=response_scoring.json()
    output=pred['predictions']
    print(output)
    outpt=pred['predictions'][0]['values'][0][0]
    print(outpt)

    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=False)

```