PROJECT TITLE

Electric motor temperature prediction

using machine learning

Artificial Intelligence & Machine Learning

TEAM ID: LTVIP2026TMIDS42348

TEAM MEMBERS(ROLES):

1. Shaik Rizwana– (Team Leader)
2. R Rajagopal (Team Member)
3. Samuel Morries (Team Member)
4. Pattikonda Shaik Shafi (Team Member)

Table of Contents

1. Introduction

Electric motors are widely used in industries, electric vehicles, manufacturing units, and household appliances. One of the major reasons for motor failure is overheating. High temperature affects insulation, reduces efficiency, and eventually damages the motor permanently.Traditional monitoring systems only alert after overheating happens. This project proposes a predictive system that estimates the motor temperature in advance using Machine Learning. By predicting temperature early, industries can perform preventive maintenance and avoid costly breakdowns.

The goal of this project is to develop a Machine Learning model that can predict the motor temperature based on sensor data such as:

Voltage

Current

Speed

Torque

Ambient temperature

The system helps operators monitor motor health and increases reliability and safety.

2. Project Overview

This project uses a supervised machine learning algorithm to predict the temperature of an electric motor in real-time.

Key Objectives:

Predict motor temperature before overheating occurs
Reduce equipment failure
Enable preventive maintenance
Improve motor life and efficiency

Technologies Used:

Python

Machine Learning (Regression Algorithms)

Scikit-learn

Pandas & NumPy

Flask (for web application/API)

Matplotlib (for visualization)

Workflow:

1. Collect sensor data

2. Preprocess dataset

3. Train ML model

4. Predict temperature

5. Display results on web interface domain.

3. Architecture

The system consists of four main components:

1. Data Input Layer

receives motor sensor parameters like Voltage, current, speed, torque, ambient temperature

2. Processing Layer

Data cleaning

Normalization

Feature selection

3. Machine Learning Model

Trained regression model (Random Forest / Linear Regression)

Predicts motor temperature

4. User Interface

User enters motor parameters

System displays predicted temperature

Flow:

Sensor Data → Preprocessing → ML Model → Temperature Prediction → User Display

.

3. PROJECT STRUCTURE

- 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

4. 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.

5. Setup Instructions

These steps to run the project on computer.

Step 1: Install Python

Download and install Python (version 3.8 or above).

Step 2: Install Required Libraries

Open terminal/command prompt and run:

pip install numpy
pip install pandas
pip install matplotlib
pip install scikit-learn
pip install flask

Step 3: Download Project Files

Extract the project folder.

Installation:

Clone the repository:

Navigate into the project directory:

cd electric temperature PredictionApp-

Install Python dependencies:

pip install Flask tensorflow opencv-python Pillow numpy (or list specific versions if needed in a requirements.txt file)

Download the dataset and pre-trained model:

The dataset (if needed for re-training or local setup) should be downloaded separately (e.g., from Kaggle, as mentioned in my demo video).

6. Folder Structure

dataset → contains training data

model → saved machine learning model

templates → web pages

static → CSS files

app.py → main application

train_model.py → model training script

templates/: Contains HTML files that define the user interface (e.g., index.html).

static/: (Expected if present) Would contain static assets like css/ for stylesheets, js/ for client-side JavaScript, and uploads/ for temporary uploaded images.

Server (Backend):

app.py: The main Flask application file, handling routes, model loading, image processing, and predictions.

train_model.py: Script for training and evaluating the machine learning model.

data_prep_and_viz.py: Script for data preprocessing and visualization.

7. Running the Application

1. Open terminal

2. Navigate to project folder

cd Electric-Motor-Temperature-Prediction

3. Train the model

python train_model.py

4. Run the web application

python app.py

5. Open browser and go to:

http://127.0.0.1:5000/

Now the prediction system will appear.

Local Setup (e.g., in a terminal after cloning):

1. Ensure all prerequisites are met and dependencies are installed.
2. Start the Flask backend: python app.py
3. The application will typically run on http://127.0.0.1:5000.

Google Colab Setup (as shown in the demo video i.e., GITHUB repositary):

1. Mount Google Drive.
2. Unzip your project into /content/Electric motor temperature prediction Project.
3. Change the current working directory to /content/Electric motor temperature prediction Project.
4. Install required libraries within the Colab notebook.
5. Run app.py in the background (e.g., !nohup python app.py > app.log 2>&1 &).
6. Start ngrok to expose the Flask port (e.g., !ngrok http 5000). Access the application via the provided ngrok public URL.

8. API Documentation

The application provides a web interface rather than a public API for direct consumption. However, the backend exposes the following key routes handled by

app.py:

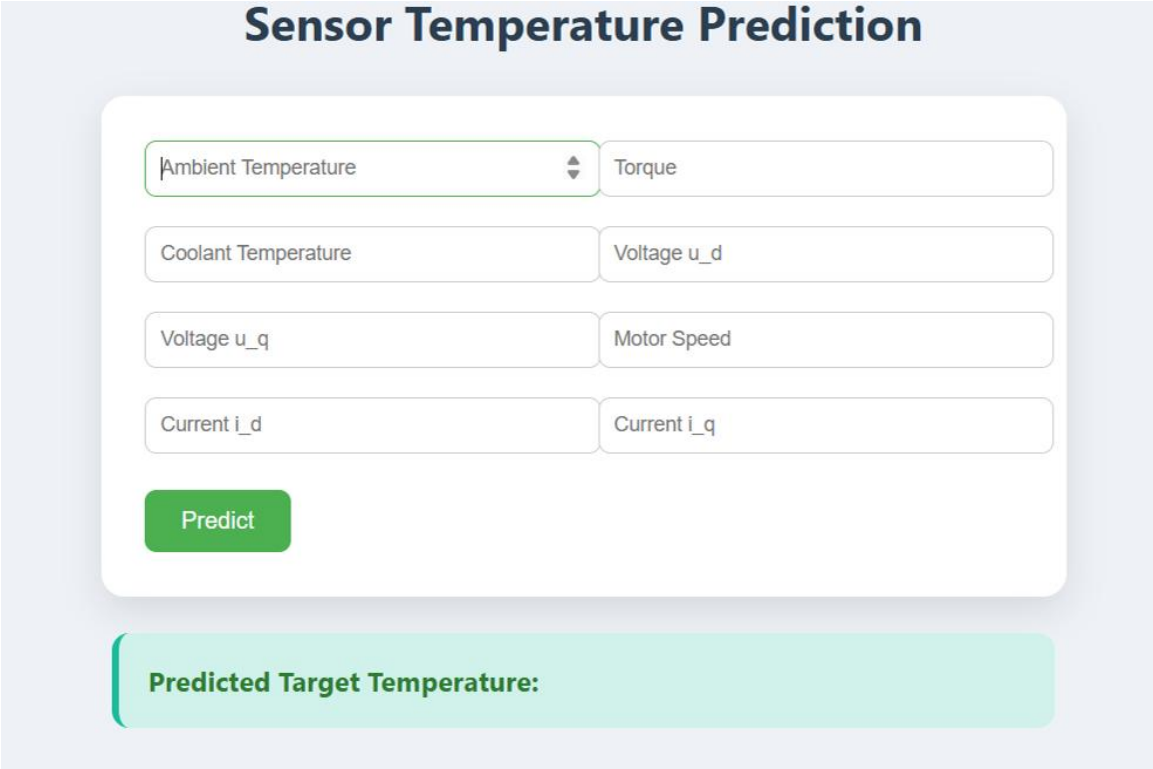
/ (GET): Serves the main index.html page, which is the entry point for the application.

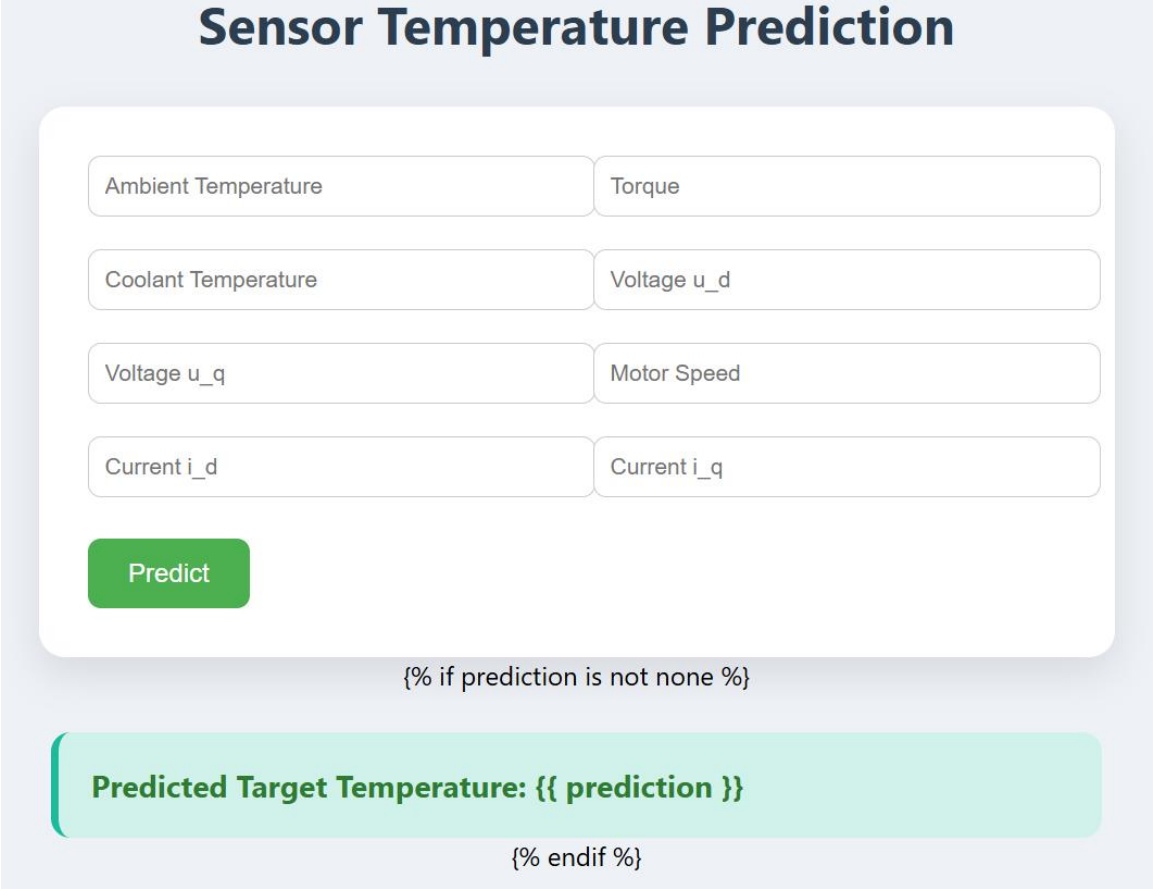
/predict (POST): Accepts an image file (typically via a form submission).

9.User Interface (Screenshots Section)

The UI is a simple web page allowing users to upload an image file. After processing, it displays the prediction result clearly.

THE WEB PAGE ALLOWING USERS TO UPLOAD AN IMAGE FILE:





PREDICTION RESULTS:

Sensor Temperature Prediction

Ambient Temperature	Torque
Coolant Temperature	Voltage u_d
Voltage u_q	Motor Speed
Current i_d	Current i_q

Predict

Predicted Target Temperature: Motor Temp: 73.57 °C

LINK **Strategy**: Testing primarily involves verifying the accuracy of the machine learning model (during training in train_model.py) and ensuring the web application correctly handles image uploads and displays predictions.

Tools: Standard Python unit testing (e.g., unittest or pytest) could be used for backend logic, and manual testing for the web interface.

11.DEMO VIDEO :

12. KNOWN ISSUES:

1. Dataset Limitation

The prediction model is trained using a limited dataset. If the motor operates under conditions not present in the training data, the prediction accuracy may decrease.

2. Sensor and Input Errors

The system relies on input parameters such as voltage, current, speed, and torque. Any noise, missing values, or incorrect sensor readings can affect the predicted temperature.

3. Lack of Real-Time Processing

The current implementation mainly works with stored data. Continuous real-time monitoring and prediction are not fully implemented.

4. Environmental Factors Ignored

Important external conditions such as ambient temperature, humidity, and cooling efficiency are not included in the model, even though they influence motor heating.

5. Limited Generalization

The trained model is suitable for a specific motor configuration. It may not provide accurate results for motors with different ratings or manufacturers.

13 .FUTURE ENHANCEMENTS:

1. IoT-Based Real-Time Monitoring

The system can be connected with IoT sensors (Arduino/Raspberry Pi) to collect live motor parameters and perform continuous temperature prediction.

2. Advanced Learning Algorithms

Deep learning models such as ANN or LSTM can be implemented to capture complex relationships between motor parameters and temperature.

3. User Interface and Dashboard

A web or mobile application can be developed to display temperature graphs, motor status, and historical performance data.

4. Alert and Notification System

Automatic alerts (SMS/Email) can be generated when the motor temperature exceeds safe limits, helping to prevent overheating and failure.

5. Predictive Maintenance

The project can be extended to detect motor faults like bearing wear, insulation failure, and overload conditions before actual breakdown occurs.