ELECTRIC MOTOR TEMPERATURE PREDICTION USING MACHINE LEARNING

Arepalli Sudeepthi– **Model Development & Deployment**

Godavarthi Sasi Kumar – **Data Preprocessing & Analysis**

Chitturi Poojitha– **Frontend Development & Integratio**

Kollu Bindu Priya – **Testing & Documentation**

Project Overview

The purpose of this project is to develop a machine learning-based system that predicts the rotor temperature of an electric motor using sensor data. The system aims to enable predictive maintenance and prevent overheating-related failures in industrial environments.

• Features:

- Data preprocessing and feature scaling
- Multiple regression model comparison
- Decision Tree model selection
- Flask-based deployment
- Manual prediction module
- Sensor-based prediction module (future enhancement)
- Error handling and validation

Architecture

Frontend

The frontend is developed using:

- HTML
- CSS

It provides:

- Home page
- Manual input form
- Sensor module interface
- Prediction result display

The user enters motor parameters through the form, and the prediction result is displayed dynamically.

Backend

The backend is implemented using Flask (Python framework).

Responsibilities of the backend:

- Load trained model (model.save)
- Load scaler (transform.save)

- Accept user input via POST request
- Scale input using MinMaxScaler
- Predict rotor temperature using Decision Tree model
- Return prediction result to the user interface

Database

This project does not use a database. Instead:

- CSV file (measures_v2.csv) is used as dataset
- Pickle files are used for model storage

Future **implementation may include cloud database integration for real-time data storage.**

Setup Instructions

• Prerequisites

- Python 3.9
- pip
- VS Code (recommended)
- Required libraries:
  - numpy
  - pandas
  - scikit-learn
  - flask
  - matplotlib
  - seaborn

Installation

Step 1: Clone Repository

git clone https://github.com/Johnpaul-225/Electric-Motor-Temperature-Prediction.git

cd Electric-Motor-Temperature-Prediction

Step 2: Install Dependencies

pip install numpy pandas scikit-learn flask matplotlib seaborn

Step 3: Train Model

Open:

Notebook/Rotor_Temperature_Detection.ipynb

Run all cells to generate:

- model.save

- transform.save

Step 4: Copy Model Files

Copy:

model.save

transform.save

Paste into:

Flask/

Folder Structure

Electric-Motor-Temperature-Prediction/

```
│
├── Dataset/
│   └── measures_v2.csv
│
├── Notebook/
│   └── Rotor_Temperature_Detection.ipynb
│
├── Flask/
│   ├── app.py
│   ├── model.save
│   ├── transform.save
│   ├── templates/
│   │   ├── home.html
│   │   ├── manual_predict.html
│   │   └── sensor_predict.html
│   └── static/
│       └── style.css
│
└── README.md
```

Running the Application

Backend (Flask)

cd Flask

python app.py

Open browser:

http://127.0.0.1:5000

API Documentation

Home Route

GET /
Returns the home page.

Manual Page

GET /manual
Displays manual prediction form.

Predict Endpoint

POST /predict

Parameters:

- ambient
- coolant
- u_d
- u_q
- motor_speed
- i_d
- i_q
- stator_yoke
- stator_winding

Response:
Predicted rotor temperature displayed on UI.

Authentication

Authentication is not implemented in this academic project.

Future enhancements may include:

- Login system
- JWT-based authentication
- Role-based access control
- ---

User Interface

The user interface includes:

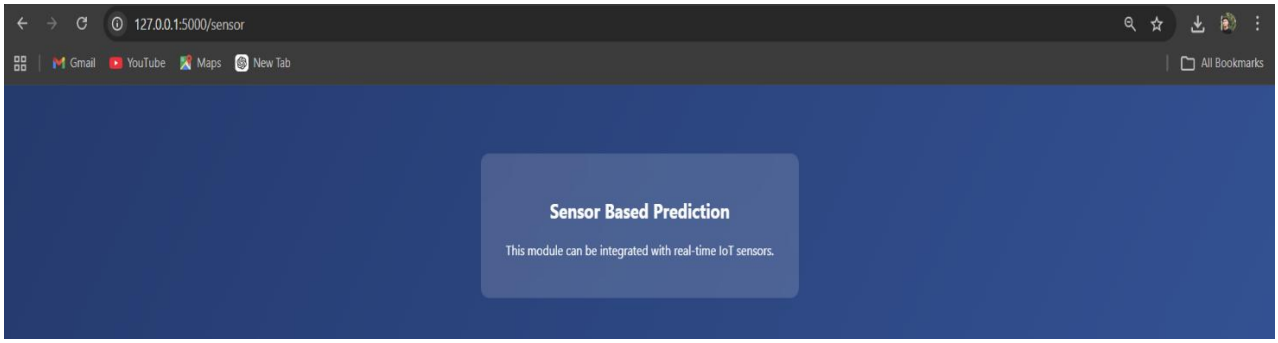
- Home page
- Manual input form
- Sensor module page
- Prediction output display

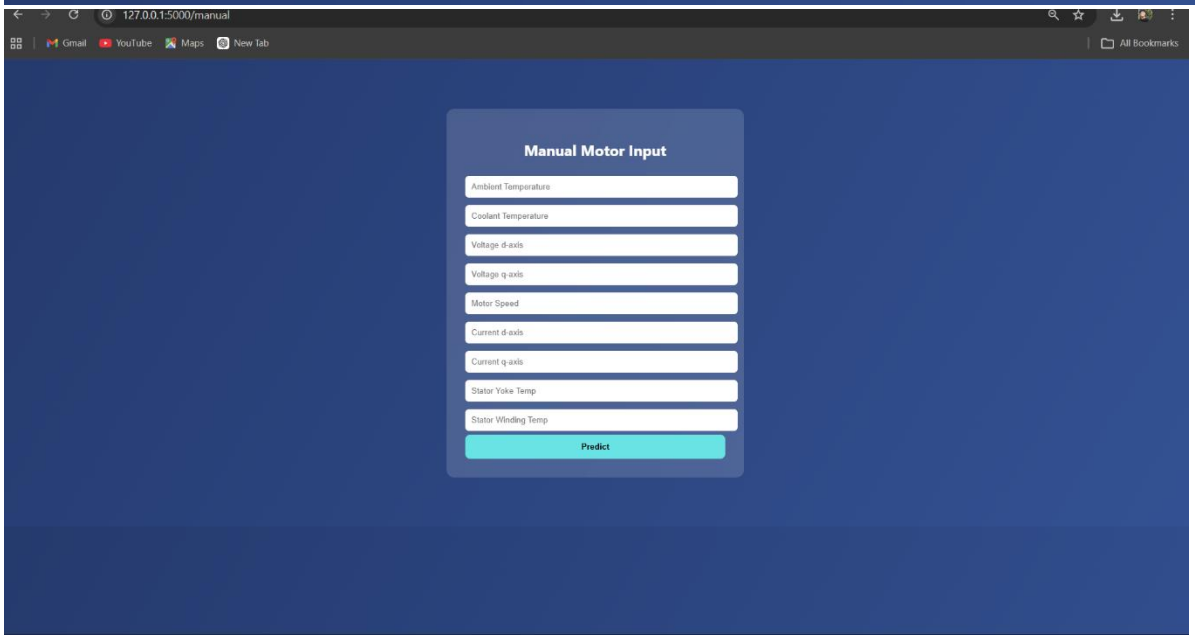
Testing

Testing included:

- Functional testing
- Input validation testing
- Model accuracy testing
- User Acceptance Testing (UAT)

All test cases passed successfully.





Known Issues

- Dataset not uploaded to GitHub due to file size limitation
- Sensor module currently conceptual

No major functional defects remain.

Future Enhancements

- Real-time IoT sensor integration
- Cloud deployment
- Mobile dashboard
- Deep learning model implementation
- Automated alert system