**Artificial Intelligence And Machine Learning.**

**(Long-Term Virtual Internship Program)**

**Project Documentation format**

**1. Introduction**

• **Project Title: Electric Motor Temperature Prediction using Machine Learning• Team Members:**

**Team Member-1:Challa Bhargav (Team leader)-developing**

**Team member -2: Chandraganti Veera Venkata Siva Prasad-developing & documentation**

**Team member-3 : Dantina Yugandhar Naresh-model training**

**Team member -4: Lawrence Galla-Testing**

**2. Project Overview**

• **Purpose:** The purpose of this project is to predict the temperature of an electric motor using machine learning techniques. The system analyzes parameters such as voltage, current, speed, torque, and ambient temperature to predict motor temperature accurately and prevent overheating.

**Features:**  Motor sensor data collection

 Data preprocessing (cleaning & scaling)

 Regression model training

 Temperature prediction

 Model performance evaluation (MAE, MSE, RMSE, R² Score)

 Simple user interface for input & prediction

 Preventive maintenance support.

**3. Architecture**

**Data Layer:**  
Sensor dataset containing motor parameters (Voltage, Current, Speed, Torque, Ambient Temperature).

• **Processing Layer:**

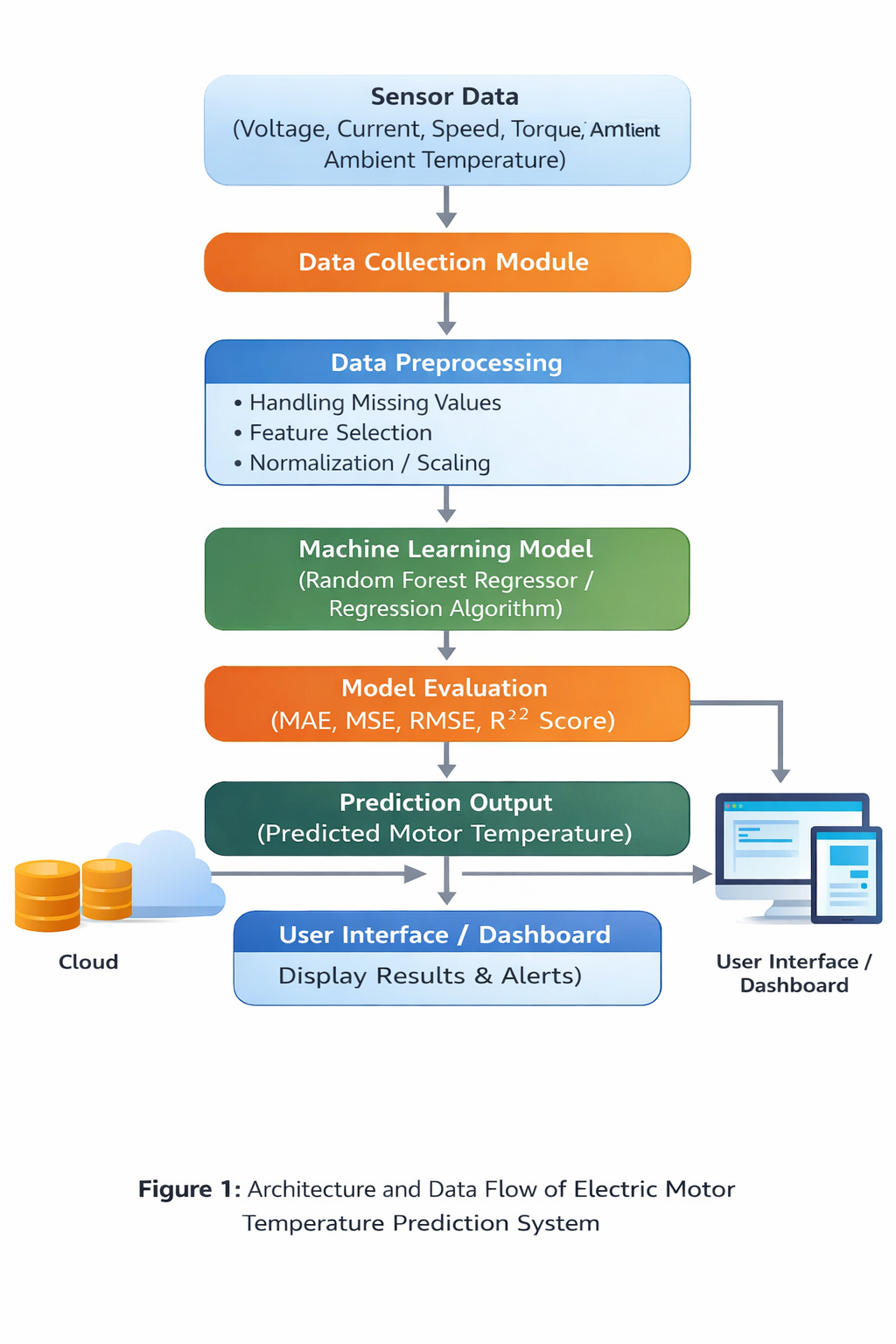
* Data cleaning
* Feature selection
* Normalization / Scaling

• **Model Layer:**

* Regression Algorithm (Random Forest Regressor)
* Model training & evaluation

• **Output Layer:**

* Predicted Motor Temperature
* Dashboard display.



**4. Setup Instructions**

• **Prerequisites:** Python 3.x

Jupyter Notebook / VS Code

Required Libraries:

Pandas

NumPy

Scikit-learn

Matplotlib

Seaborn

Flask / Streamlit (if deployed)

• Installation:

Step 1: Clone the project

git clone <repository-link>

Step 2: Navigate to project folder

cd motor-temperature-prediction

Step 3: Install dependencies

pip install -r requirements.txt

Step 4: Run the application

python app.py

**5. Folder Structure**

• **Dataset Folder:** Contains CSV file with motor sensor data.

**• Model Folder:** Contains trained model file (.pkl).

**• Application Folder:** Contains prediction script and deployment files.

**• Notebook Folder:**Contains Jupyter Notebook for training and evaluation.

**6. Running the Application**

To train the model:

python train\_model.py

• To start the prediction system:

python app.py

If using Streamlit:

streamlit run app.py

**7. API Documentation**

**Algorithm Used:** Random Forest Regressor

• **Input Features:**

* Voltage
* Current
* Speed
* Torque
* Ambient Temperature

• **Output:**

* Predicted Motor Temperature

• **Evaluation Metrics:**

* MAE
* MSE
* RMSE
* R² Score

**8. Authentication**

(Not applicable for basic ML model)

If deployed as web app:

* Optional login system for user access
* Secure API endpoints

**9. User Interface**

The system provides:

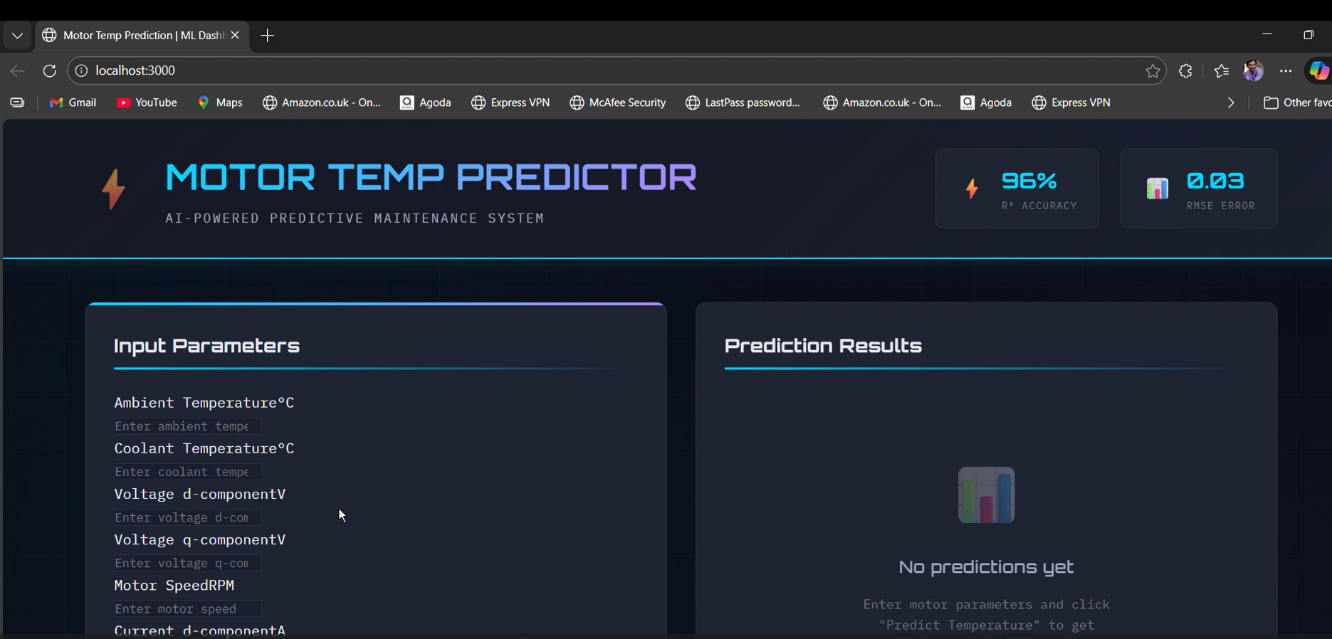
* Input form for motor parameters
* Predict button
* Display of predicted temperature
* Display of model evaluation metrics

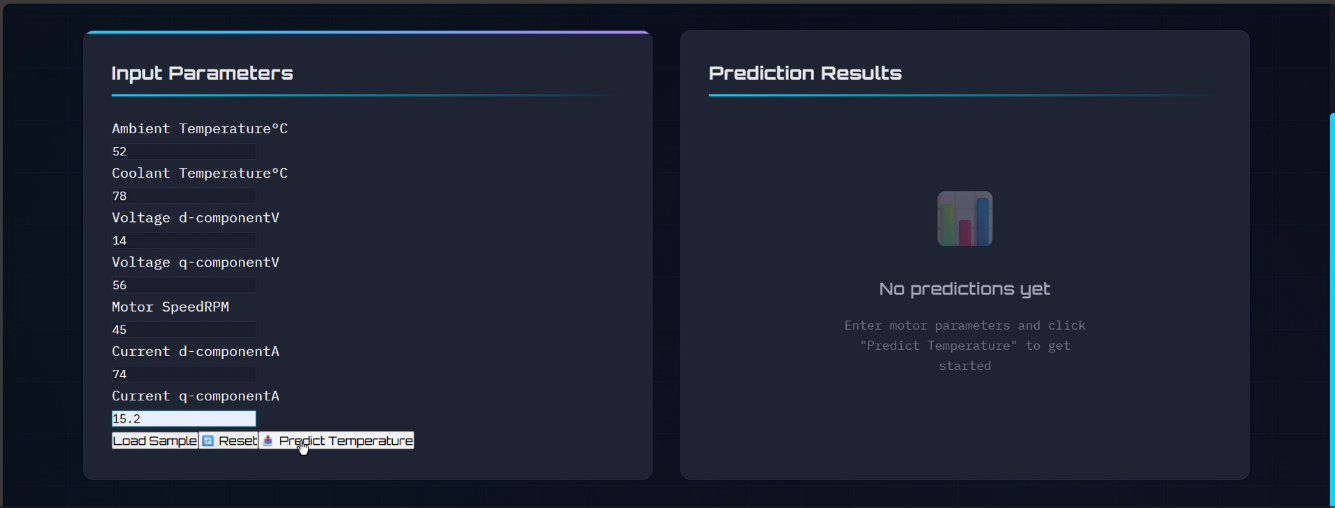
**10. Testing**

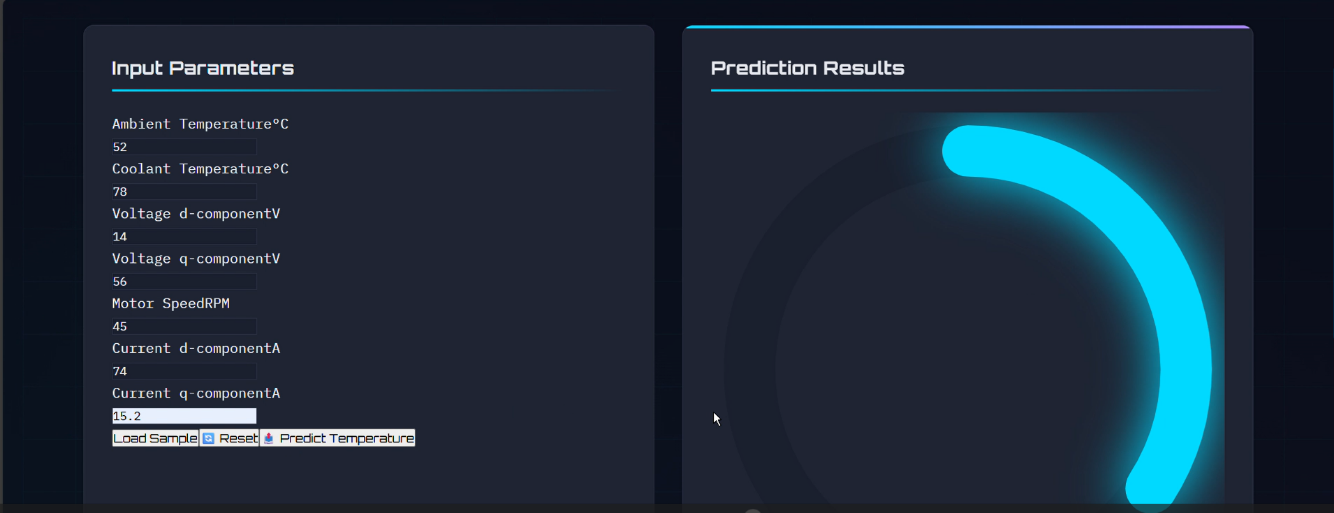
• Train-Test Split (80:20)  
• Cross Validation (K-Fold = 5)  
• Manual input testing  
• Invalid input testing

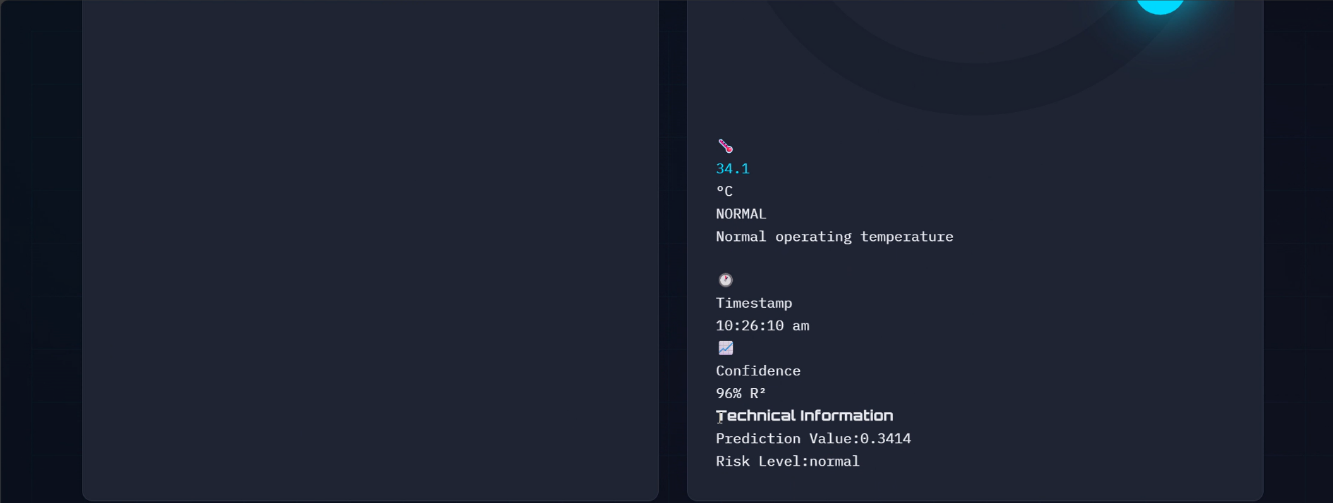
**11. Screenshots or Demo**

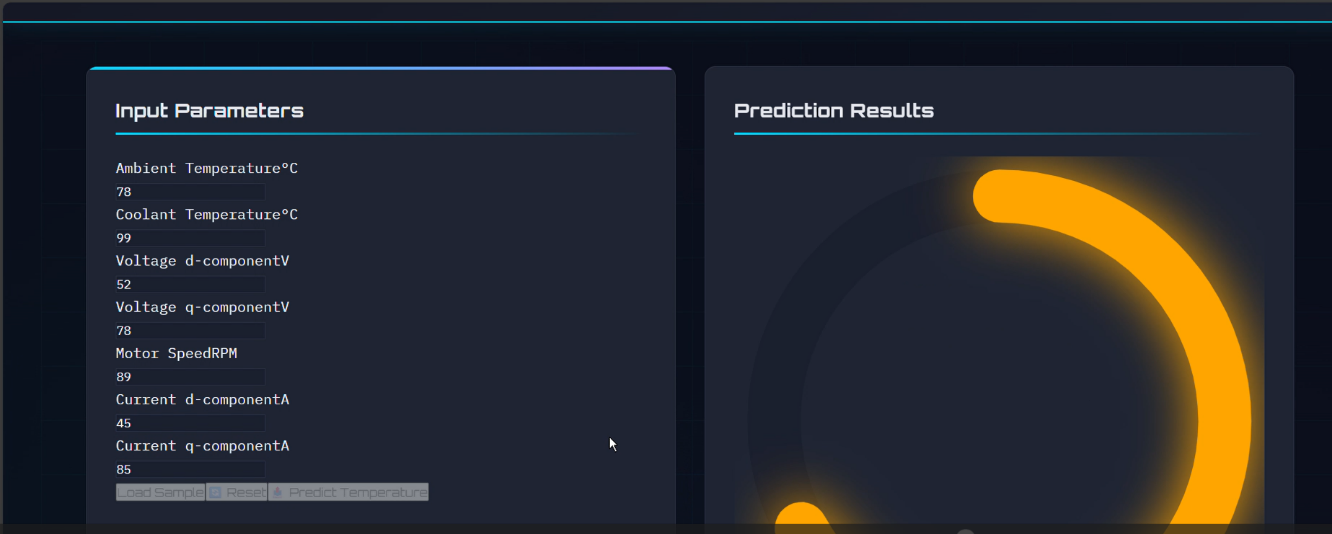
• screenshots and a link to a demo to showcase the application.

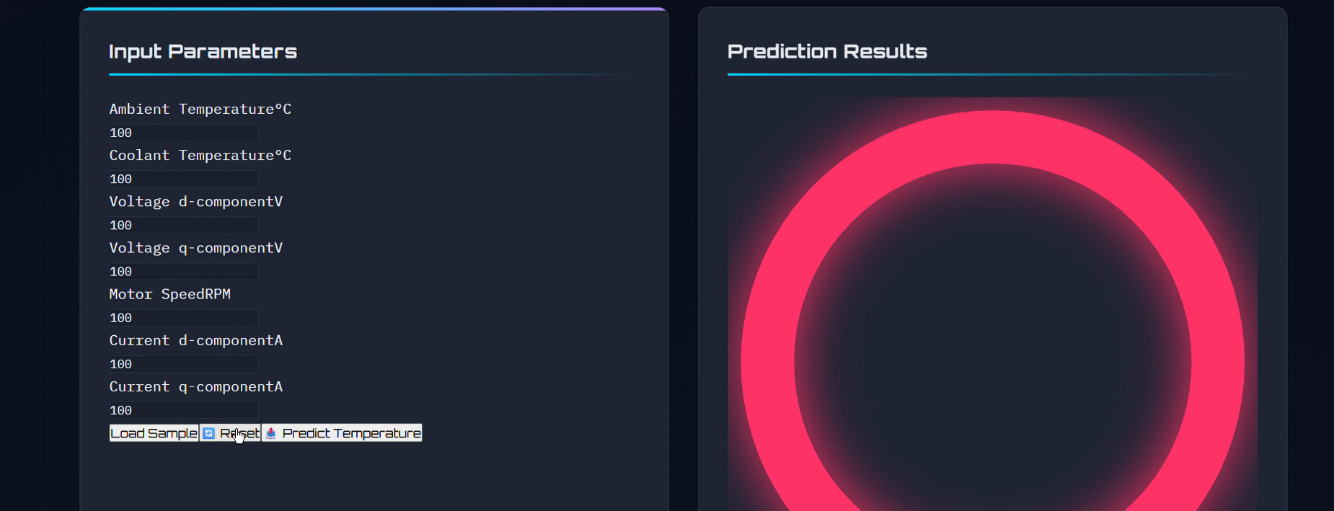


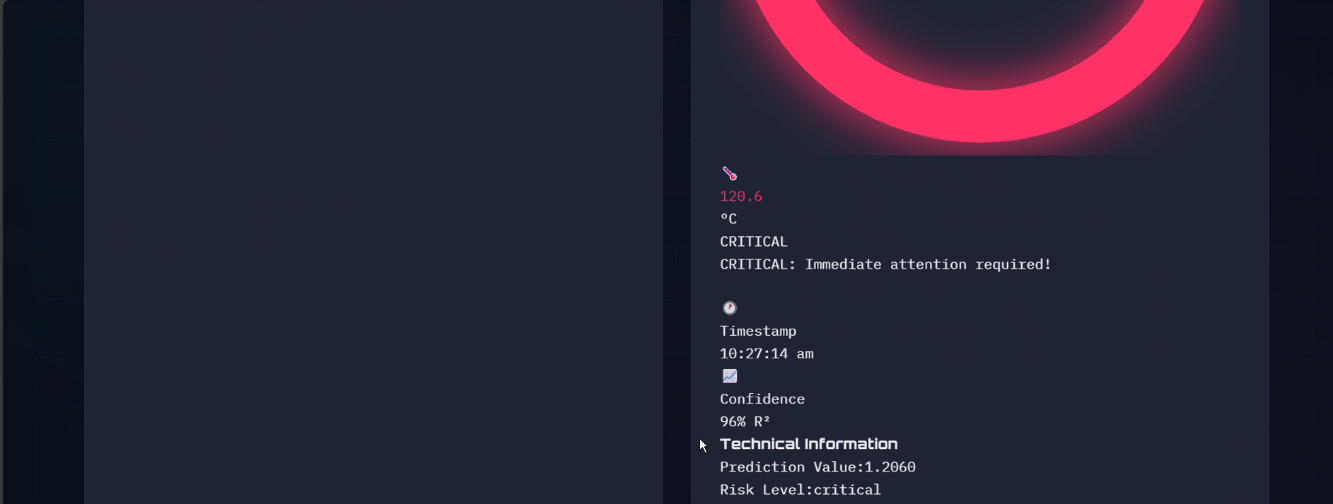












**🡪GIT HUB LINK:** [**https://github.com/Bhargav-2005/motor-temp-prediction**](https://github.com/Bhargav-2005/motor-temp-prediction)

**🡪DEMO VIDEO DRIVE LINK:** [**https://drive.google.com/file/d/1dhEmzQ1kw2qdn\_rTBEAHmKEh\_l\_UhthD/view?usp=drive\_link**](https://drive.google.com/file/d/1dhEmzQ1kw2qdn_rTBEAHmKEh_l_UhthD/view?usp=drive_link)

**12. Known Issues**

• Model performance depends on dataset quality  
• Slight delay during first model loading  
• Not integrated with real-time IoT sensors (offline dataset only)

**13. Future Enhancements**

• Integration with real-time IoT motor sensors  
• Cloud deployment  
• Mobile application interface  
• Deep Learning model for improved accuracy  
• Automatic alert system for overheating