Electric Motor Temperature Prediction

Project Overview

This project aims to predict the **temperature of a permanent magnet (pm) motor** based on various operational parameters. By analyzing features such as voltages, currents, and coolant temperature, the goal is to develop a regression model that can accurately forecast the motor's internal temperature. This is crucial for condition monitoring, preventing overheating, and extending the lifespan of electric motors.

Technical Highlights

- **Dataset**: The project uses a dataset related to electric motor temperature. The specific dataset is titled measures_v2.csv and is likely from a Kaggle source related to electric motor temperature prediction, though the specific link is missing from the provided code block.
- **Size**: 20,000 entries, 13 columns.
- **Key Features**:
  - **Motor Electrical Signals**: u_q, u_d, i_d, i_q.
  - **Temperatures**: coolant, stator_winding, stator_tooth, stator_yoke, ambient.
  - **Performance Metrics**: motor_speed, torque.
- **Approach**:
  - **Data Cleaning**: The dataset appears to be clean, with no missing values or duplicates in the sample used.
  - **Exploratory Data Analysis**: Histograms, boxplots, and a heatmap were used for visualization to understand data distributions and correlations. The heatmap reveals several strong correlations between the features.
  - **Regression Task**: The target variable is pm, which likely represents the permanent magnet temperature.
  - **Models Used**:
    - A suite of regression models were trained, including Ridge Regression, XGBoost, Random Forest, AdaBoost, Gradient Boosting, Bagging, Decision Tree, SVR, and K-Nearest Neighbors (KNN).
- **Best R² Score**:
  - **0.99992** with Random Forest Regressor.
  - **0.99984** with KNN Regressor.
  - **0.99983** with XGBoost Regressor.

    - The extremely high R^2 scores across multiple models indicate that the input features are highly predictive of the motor's temperature.

Purpose and Applications

- Enable **predictive maintenance** by forecasting motor temperatures to prevent overheating and component failure.
- Optimize the operation of electric motors in industrial settings to improve efficiency and reduce energy consumption.
- Support the design of more effective cooling systems for motors.
- Provide a foundational model for developing real-time condition monitoring and fault diagnosis systems.

Installation

Clone the repository and download the dataset.

Install the necessary libraries:

```
pip install pandas numpy seaborn matplotlib scikit-learn xgboost
```

Collaboration

We welcome contributions to improve the project. You can help by:

- Performing a deeper analysis of the relationships between the input features and the target variable to understand the underlying physics.
- Investigating the impact of the highly correlated features on the models.
- Exploring more advanced time-series forecasting techniques, as temperature prediction is often a time-dependent problem.
- Adding explainability (e.g., SHAP or LIME) to understand which operational parameters are the most significant drivers of motor temperature.

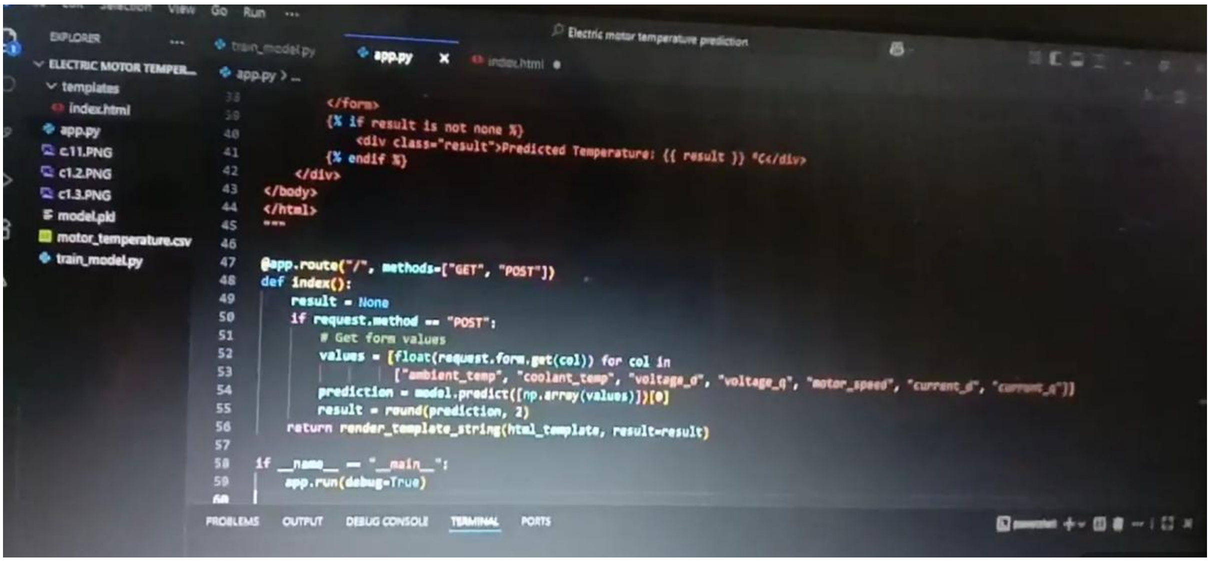
GITHUB LINK:

https://github.com/manyam-vamsi/Electric-motor-temperature-prediction

Video Link:

https://drive.google.com/file/d/1mWtbZcGJHYywfbDpy9sVU8WzfnpwCbFK/view?usp=sharing

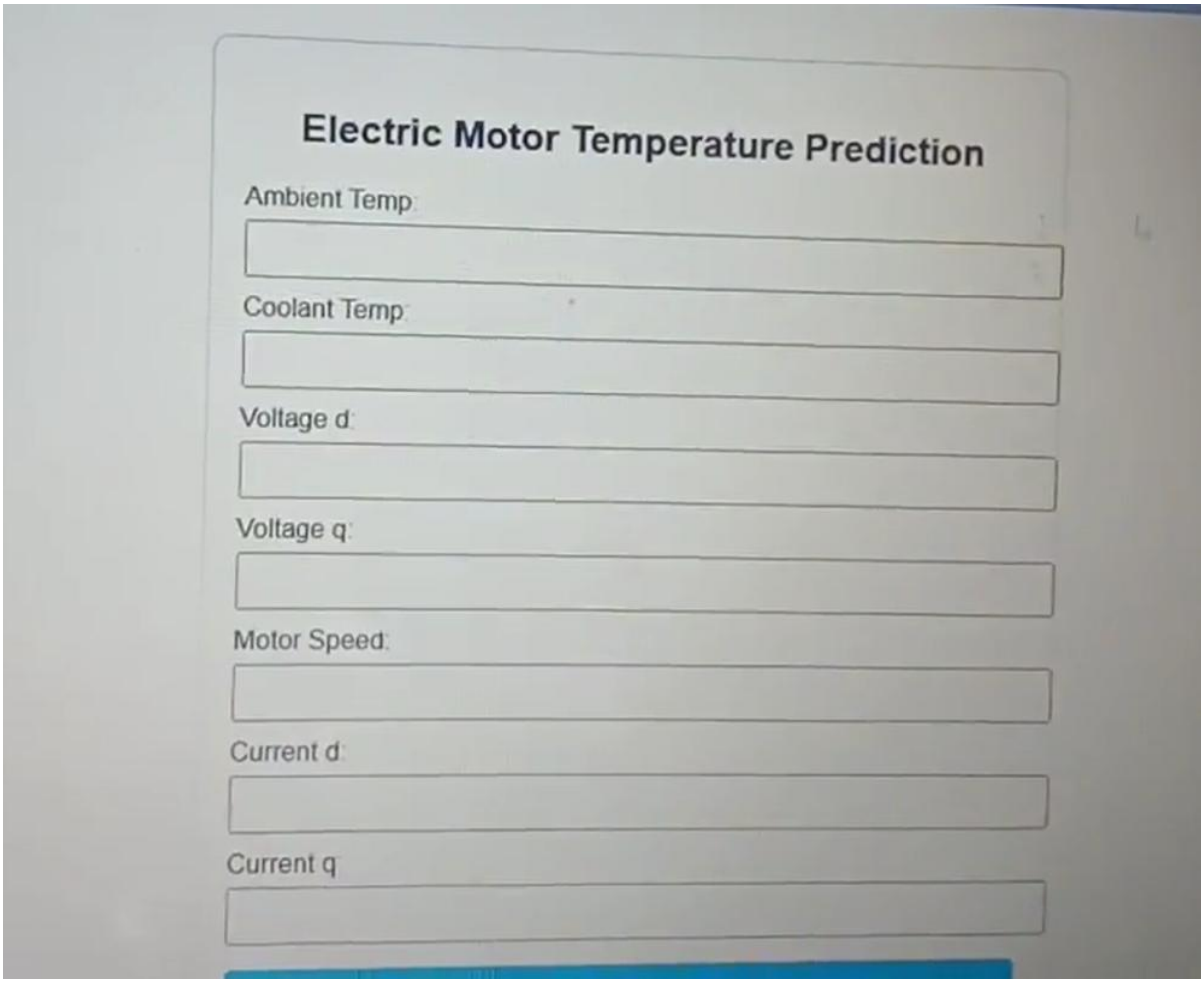
PHOTOS



```
            </form>
            {% if result is not none %}
                <div class="result">Predicted Temperature: {{ result }} °C</div>
            {% endif %}
        </div>
    </body>
    </html>
    """

    @app.route("/", methods=["GET", "POST"])
    def index():
        result = None
        if request.method == "POST":
            # Get form values
            values = [float(request.form.get(col)) for col in
                      ["ambient_temp", "coolant_temp", "voltage_d", "voltage_q", "motor_speed", "current_d", "current_q"]]
            prediction = model.predict([np.array(values)])[0]
            result = round(prediction, 2)
        return render_template_string(html_template, result=result)

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



Electric Motor Temperature Prediction

Ambient Temp:

Coolant Temp:

Voltage d:

Voltage q:

Motor Speed:

Current d:

Current q: