Project Report

Electric Motor Temperature Prediction

Introduction:

Permanent Magnet Synchronous Motors (PMSMs) are integral to modern electric vehicles, but their performance and safety are critically impacted by temperature. The primary challenge lies in the difficulty and cost of real-time temperature monitoring within the rotor. Excessive heat can lead to the demagnetization of permanent magnets, which is an irreversible process, leading to a loss of motor efficiency and potential system failure.

This project aims to address this challenge by developing a machine learning model to accurately predict the permanent magnet (PM) rotor temperature. The model uses readily available sensor data from a test bench as input to predict the internal temperature, which is difficult to measure directly. This approach is crucial for implementing advanced thermal management systems, optimizing motor performance, and ensuring the longevity and safety of electric powertrains in the automotive industry.

Methodology:

➤ **Data Collection and Preprocessing:**
The project utilizes the "Electric Motor Temperature" dataset, a large-scale collection of sensor data from a PMSM on a test bench. The dataset, contained in measures_v2.csv, consists of over 1.3 million records, each sampled at 2 Hz. The features include a combination of electrical, mechanical, and thermal properties of the motor and its environment.

The key features are:

- **u_q and u_d**: Voltage d and q components.
- **i_d and i_q**: Current d and q components.
- **motor_speed**: The rotational speed of the motor.
- **torque**: The torque produced by the motor.
- **coolant**: The temperature of the coolant at the outflow.
- **ambient**: The ambient temperature near the stator.
- **stator_winding**, **stator_tooth**, **stator_yoke**: Temperatures of different components of the stator.
- **pm**: The permanent magnet temperature, which serves as the primary target variable for prediction.

The dataset also includes a profile_id, which identifies different measurement sessions. This column was removed during analysis and model building as it's a categorical identifier and not a predictive feature. To ensure effective model training, a MinMaxScaler was applied to normalize the features, bringing them all into a consistent range between 0 and 1.

➤ **Exploratory Data Analysis (EDA):**
Uni-variate analysis was conducted on all features using distribution and box plots. These visualizations were used to identify the statistical properties of each variable, such as their central tendency, spread, and the presence of any outliers or skewed distributions. This step helps in understanding the data and informing subsequent preprocessing and model selection decisions.

➤ **Model Training and Evaluation:**

This project frames the temperature prediction as a regression problem. The dataset was split into training and testing subsets to validate the models' performance on unseen data. Several machine learning regression algorithms were implemented to find the best-performing model for predicting the PM temperature.

The models trained and evaluated were:

- **Linear Regression**: A simple baseline model.
- **Decision Tree Regressor**: A tree-based model capable of capturing non-linear relationships.
- **Random Forest Regressor**: An ensemble method that uses multiple decision trees to improve accuracy and reduce overfitting.
- **Support Vector Regressor (SVR)**: A powerful regression model that finds the best fit line while considering a margin of error.

The performance of each model was evaluated using standard regression metrics, such as the R^2 (R-squared) score and the Root Mean Squared Error (RMSE). Cross-validation was also performed to ensure the models' robustness and generalize well to new data.

Deployment:

The app.py file demonstrates a simple deployment of the trained models using the Flask web framework. The application loads the model_dr.save and transform.save files, which are saved using joblib.

- **model_dr.save**: This is a saved Decision Tree Regressor model.
- **transform.save**: This is a saved MinMaxScaler object used to pre process the input data before it is fed into the model for prediction.

The application includes two main routes:

- /: Renders an HTML page (Manual_predict.html) that provides a user interface for manual temperature prediction.
- /y_predict: Handles POST requests from the user interface, takes the input features, uses the loaded MinMaxScaler to scale them, and then applies the model_dr to predict the rotor temperature. The prediction is then returned to the user.

Conclusion:

The project successfully demonstrates the application of machine learning regression algorithms to predict the rotor temperature of a PMSM using a dataset of sensor measurements. The comprehensive approach, from data preprocessing and exploratory analysis to training and evaluating multiple models, validates the feasibility of this method. The use of joblib and Flask in app.py showcases a practical approach to deploying a predictive model. Ultimately, the successful implementation of such models can help improve thermal management and overall performance in permanent magnet synchronous motors, which is highly beneficial for the electric vehicle industry.