

Model Development Phase Template

Date	July 2024
Team ID	Team-740101
Project Title	Power Consumption Analysis For Households
Maximum Marks	10 Marks

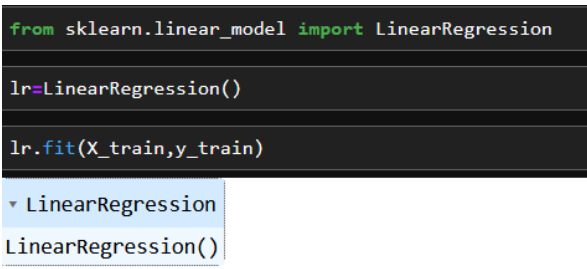
Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):

Paste the screenshot of the model training code

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics
Model 1	Linear regression is a valuable tool for analyzing household power consumption, employing a straightforward yet powerful approach to model the relationship between dependent variables (such as daily kWh usage) and various independent predictors (e.g., temperature, time of day, household size).	 <pre>from sklearn.linear_model import LinearRegression lr=LinearRegression() lr.fit(X_train,y_train)</pre>

<p>Model 2</p>	<p>Decision Tree Regressor is a powerful machine learning technique used to analyze power consumption in households by partitioning data into subsets based on different features such as temperature, time of day, and appliance usage patterns.</p>	<pre> from sklearn.tree import DecisionTreeRegressor from sklearn.datasets import make_regression from sklearn.model_selection import train_test_split from sklearn.metrics import mean_squared_error # Generate a random regression problem X, y = make_regression(n_samples=1000, n_features=10, noise=0.1, random_state=42) # Split the data into training and test sets X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # Create the decision tree regressor regressor = DecisionTreeRegressor(random_state=42) # Train the model regressor.fit(X_train, y_train) # Make predictions y_pred = regressor.predict(X_test) # Evaluate the model mse = mean_squared_error(y_test, y_pred) print(f'Mean Squared Error: {mse}') print(f'R Squares value:', metrics.r2_score(y_test, y_pred)) print(f'RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred))) print(f'MAE:', metrics.mean_absolute_error(y_test, y_pred)) Mean Squared Error: 6381.005171945824 R Squares value: 0.6217795975822628 RMSE: 79.88119410698994 MAE: 62.928834813527175 </pre>
<p>Model 3</p>	<p>Random Forest Regressor is a powerful machine learning model used in power consumption analysis for households. It operates by constructing multiple decision trees during training and outputs the average prediction of the individual trees for regression tasks..</p>	<pre> from sklearn.ensemble import RandomForestRegressor from sklearn.datasets import make_regression from sklearn.model_selection import train_test_split from sklearn.metrics import mean_squared_error # Generate a random regression problem X, y = make_regression(n_samples=1000, n_features=10, noise=0.1, random_state=42) # Split the data into training and test sets X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # Create the random forest regressor regressor = RandomForestRegressor(n_estimators=100, random_state=42) # Train the model regressor.fit(X_train, y_train) # Make predictions y_pred = regressor.predict(X_test) # Evaluate the model mse = mean_squared_error(y_test, y_pred) print(f'Mean Squared Error: {mse}') print(f'MAE:', metrics.mean_absolute_error(y_test, y_pred)) # Calculate RMSE (Root Mean Squared Error) print(f'RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred))) print(f'R Squares value:', metrics.r2_score(y_test, y_pred)) Mean Squared Error: 2686.5521627095645 MAE: 40.125547430828625 RMSE: 51.0544839501938 R Squares value: 0.8455022082982877 </pre>