Model Optimization and Tuning Phase

Date	2 July 2024
Team ID	739798
Project Title	Power Consumption Analysis for Households
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

Model	Tuned Hyperparameters	Optimal Values
Linear Regression	-	-
Random Forest Regressor	-	-
Decision Tree Regressor	-	-
XGBoost Regressor	-	-

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric
Linear Regression	`#checking the metrics` `from sklearn import metrics` `MSE = metrics.mean_squared_error(y_test, pred)` `print('The MSE value:', MSE)` `RMSE = np.sqrt(metrics.mean_squared_error(y_test, pred))` `print('The RMSE value:', RMSE)` `MAE = metrics.mean_absolute_error(y_test, pred)` `print('The MAE value:', MAE)` `R2 = metrics.r2_score(y_test, pred)*100` `print('The R2 Value:', R2)` `The MSE value: 0.0018226463072696967` `The RMSE value: 0.042692461949033776` `The MAE value: 0.027455608456450082` `The R2 Value: 99.83631695586827`
Random Forest Regressor	`#checking the metrics` `mse1 = metrics.mean_squared_error(y_test, prediction)` `print("The MSE value:",mse1)` `rmse1 = np.sqrt(metrics.mean_squared_error(y_test, prediction))` `print("The RMSE value:",rmse1)` `mae1 = metrics.mean_absolute_error(y_test, prediction)` `print("The MAE value:",mae1)` `r2 = metrics.r2_score(y_test, prediction)*100` `print("The R2:",r2)` `The MSE value: 0.0012603120293811943` `The RMSE value: 0.03550087364250624` `The MAE value: 0.021363940431370586` `The R2: 99.88681747594026`

Decision Tree Regressor	``` #checking the metrics mse2 = metrics.mean_squared_error(y_test, y_pred) print("The MSE value:",mse1) rmse2 = np.sqrt(metrics.mean_squared_error(y_test, y_pred)) print("The RMSE value",rmse1) mae2 = metrics.mean_absolute_error(y_test, y_pred) print("The MAE value:",mae2) r2score = metrics.r2_score(y_test,y_pred)*100 print("The R2 value:",r2score) ``` ``` The MSE value: 0.0012603120293811943 The RMSE value 0.03550087364250624 The MAE value: 0.02275384562302618 The R2 value: 99.85924226739422 ```
XGBoost Regressor	``` #checking the metrics mse3= metrics.mean_squared_error(y_test, y_pred) print("The MSE value:",mse3) rmse3 = np.sqrt(metrics.mean_squared_error(y_test, ypred)) print("The RMSE values:",rmse3) mae3 = metrics.mean_absolute_error(y_test, ypred) print("The MAE value:",mae3) r2_Score = metrics.r2_score(y_test, ypred)*100 print("The R2: value",r2_Score) ``` ``` The MSE value: 0.001567367975800379 The RMSE values: 0.033831248936509274 The MAE value: 0.020863928648037145 The R2: value 99.89721319781589 ```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
XGBoost Regressor	The Extreme Gradient Boost Regression is the final model chosen because of its best overall performance compared to the other models. It captures the variance in the data very well with minimal prediction error. XGBoost can capture complex non-linear relationships.