Model Development Phase

Date	1 July 2024
Team ID	739798
Project Title	Power Consumption Analysis for Households
Maximum Marks	4 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 classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

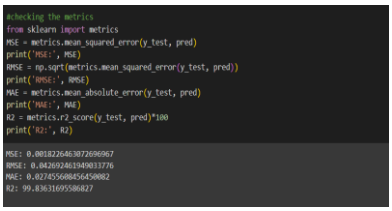
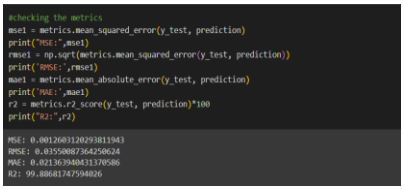
```
#importing the linear regression
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
#predicting the values
pred=lr.predict(X_test)
print("The predicted values are:",pred)
#checking the metrics
from sklearn import metrics
MSE = metrics.mean_squared_error(y_test, pred)
print('MSE:', MSE)
RMSE = np.sqrt(metrics.mean_squared_error(y_test, pred))
print('RMSE:', RMSE)
MAE = metrics.mean_absolute_error(y_test, pred)
print('MAE:', MAE)
R2 = metrics.r2_score(y_test, pred)*100
print('R2:', R2)
```

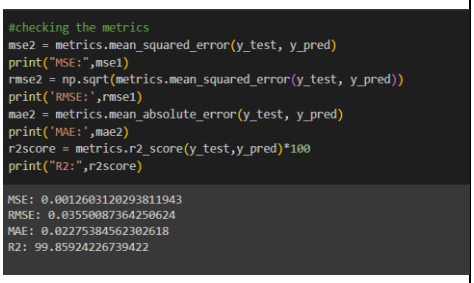
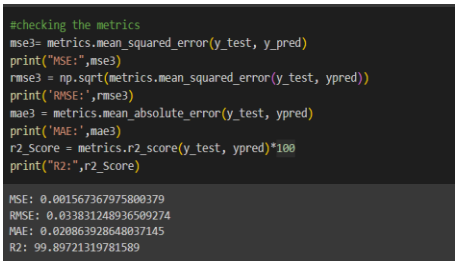
```
#importing the Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
rf.fit(X_train,y_train)
#predicting the values
prediction = rf.predict(X_test)
print("the predicted values:",prediction)
#checking the metrics
mse1 = metrics.mean_squared_error(y_test, prediction)
print("MSE:",mse1)
rmse1 = np.sqrt(metrics.mean_squared_error(y_test, prediction))
print('RMSE:',rmse1)
mae1 = metrics.mean_absolute_error(y_test, prediction)
print('MAE:',mae1)
r2 = metrics.r2_score(y_test, prediction)*100
print("R2:",r2)
```

```
#importing the Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_train,y_train)

#predicting the values
y_pred = regressor.predict(X_test)
print("The predicted values are:",y_pred)

#checking the metrics
mse2 = metrics.mean_squared_error(y_test, y_pred)
print("MSE:",mse1)
rmse2 = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print('RMSE:',rmse1)
mae2 = metrics.mean_absolute_error(y_test, y_pred)
print('MAE:',mae2)
r2score = metrics.r2_score(y_test,y_pred)*100
print("R2:",r2score)
```

```
#importing the XGBoost Regressor
import xgboost
from xgboost import XGBRegressor
xgbr= XGBRegressor()

xgbr.fit(X_train,y_train)
#predicting the values
ypred = xgbr.predict(X_test)
print("The predicted values are:",ypred)

#checking the metrics
mse3= metrics.mean_squared_error(y_test, y_pred)
print("MSE:",mse3)
rmse3 = np.sqrt(metrics.mean_squared_error(y_test, ypred))
print('RMSE:',rmse3)
mae3 = metrics.mean_absolute_error(y_test, ypred)
print('MAE:',mae3)
r2_Score = metrics.r2_score(y_test, ypred)*100
print("R2:",r2_Score)
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Linear Regression		99.83	-
Random Forest Regressor		99.88	-

Decision Tree Regressor		99.85	-
XGBoost Regressor		99.89	-