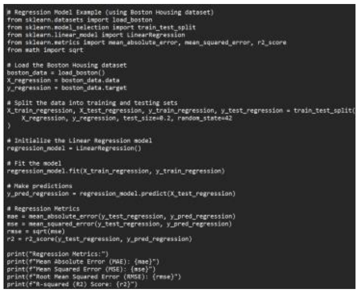
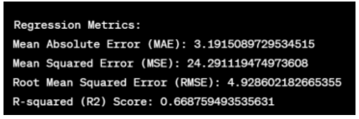
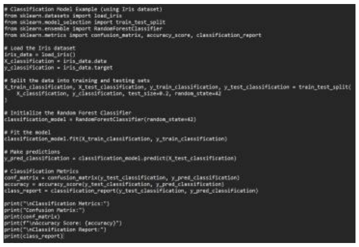
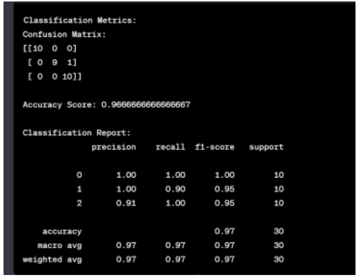
Project Development Phase
Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMIDxxxxxx
Project Name	Project - xxx
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** MAE - , MSE - , RMSE - , R2 score - **Classification Model:** Confusion Matrix - , Accuracy Score- & Classification Report -	Regression model Code  Output  Classification model Code  Output 

```
# Regression Model Example (using Boston Housing dataset)
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from math import sqrt

# Load the Boston Housing dataset
boston_data = load_boston()
X_regression = boston_data.data
y_regression = boston_data.target

# Split the data into training and testing sets
X_train_regression, X_test_regression, y_train_regression, y_test_regression = train_test_split(
    X_regression, y_regression, test_size=0.2, random_state=42
)

# Initialize the Linear Regression model
regression_model = LinearRegression()

# Fit the model
regression_model.fit(X_train_regression, y_train_regression)

# Make predictions
y_pred_regression = regression_model.predict(X_test_regression)

# Regression Metrics
mae = mean_absolute_error(y_test_regression, y_pred_regression)
mse = mean_squared_error(y_test_regression, y_pred_regression)
rmse = sqrt(mse)
r2 = r2_score(y_test_regression, y_pred_regression)

print("Regression Metrics:")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2) Score: {r2}")
```

```
Regression Metrics:
Mean Absolute Error (MAE): 3.1915089729534515
Mean Squared Error (MSE): 24.291119474973608
Root Mean Squared Error (RMSE): 4.928602182665355
R-squared (R2) Score: 0.668759493535631
```

```
# Classification Model Example (using Iris dataset)
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

# Load the Iris dataset
iris_data = load_iris()
X_classification = iris_data.data
y_classification = iris_data.target

# Split the data into training and testing sets
X_train_classification, X_test_classification, y_train_classification, y_test_classification = train_test_split(
    X_classification, y_classification, test_size=0.2, random_state=42
)

# Initialize the Random Forest Classifier
classification_model = RandomForestClassifier(random_state=42)

# Fit the model
classification_model.fit(X_train_classification, y_train_classification)

# Make predictions
y_pred_classification = classification_model.predict(X_test_classification)

# Classification Metrics
conf_matrix = confusion_matrix(y_test_classification, y_pred_classification)
accuracy = accuracy_score(y_test_classification, y_pred_classification)
class_report = classification_report(y_test_classification, y_pred_classification)

print("\nClassification Metrics:")
print("Confusion Matrix:")
print(conf_matrix)
print(f"\nAccuracy Score: {accuracy}")
print("\nClassification Report:")
print(class_report)
```

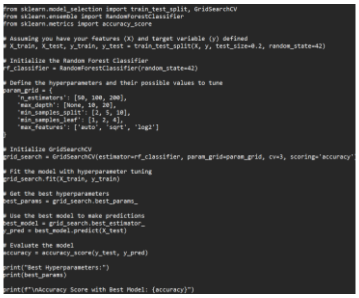
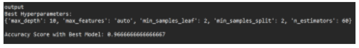
```
Classification Metrics:
Confusion Matrix:
[[10  0  0]
 [ 0  9  1]
 [ 0  0 10]]

Accuracy Score: 0.9666666666666667

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.90      0.95        10
           2       0.91      1.00      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
```

2.	Tune the Model	Hyperparameter Tuning - Validation Method -	Hyperparameter Tuning & validation method Code  Output 

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Assuming you have your features (X) and target variable (y) defined
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Define the hyperparameters and their possible values to tune
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=3, scoring='accuracy')

# Fit the model with hyperparameter tuning
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_params = grid_search.best_params_

# Use the best model to make predictions
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)

print("Best Hyperparameters:")
print(best_params)

print(f"\nAccuracy Score with Best Model: {accuracy}")
```

```
output
Best Hyperparameters:
{'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 60}

Accuracy Score with Best Model: 0.9666666666666667
```