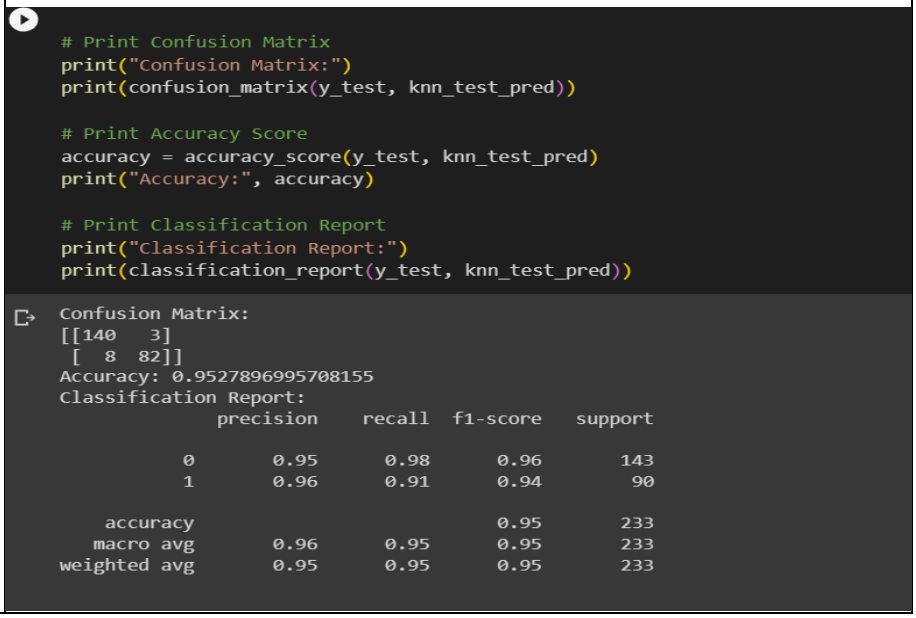
Project Development Phase
Model Performance Test

Date	19 May 2023
Team ID	NM2023TMID01937
Project Name	Project – Audit AI: A Machine Learning for Detecting Fraud in Audit Data

Model Performance Testing:

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

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

```
[139] mae = mean_absolute_error(y_test, knn_test_pred)

    # Calculate MSE
    mse = mean_squared_error(y_test, knn_test_pred)

    # Calculate RMSE
    rmse = mean_squared_error(y_test, knn_test_pred, squared=False)

    # Calculate R2 score
    r2 = r2_score(y_test, knn_test_pred)

    # Print MAE, MSE, RMSE, R2 score
    print("MAE:", mae)
    print("MSE:", mse)
    print("RMSE:", rmse)
    print("R2 score:", r2)
```

```
MAE: 0.04721030042918455
MSE: 0.04721030042918455
RMSE: 0.21727931431497235
R2 score: 0.8008547008547009
```

```
# Print Confusion Matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, knn_test_pred))

# Print Accuracy Score
accuracy = accuracy_score(y_test, knn_test_pred)
print("Accuracy:", accuracy)

# Print Classification Report
print("Classification Report:")
print(classification_report(y_test, knn_test_pred))
```

```
Confusion Matrix:
[[140   3]
 [  8  82]]
Accuracy: 0.9527896995708155
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       143
           1       0.96      0.91      0.94        90

    accuracy                           0.95       233
   macro avg       0.96      0.95      0.95       233
weighted avg       0.95      0.95      0.95       233
```

2.	Tune the Model	Hyperparameter Tuning	

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.impute import SimpleImputer
from sklearn.model_selection import GridSearchCV

# Assuming your DataFrame is named 'df'
# Extract the features (x_train) and the target variable (y_train)
x_train = df.drop('Risk', axis=1)
y_train = df['Risk']
# Select only the desired 14 features from x_train
selected_features = ['Sector_score', 'LOCATION_ID', 'PARA_A', 'Score_A', 'Risk_A',
                     'PARA_B', 'Score_B', 'Risk_B', 'TOTAL', 'numbers',
                     'Money_Value', 'Score_MV', 'District_Loss', 'History']
x_train_selected = x_train[selected_features]
# Handle missing values by replacing them with the mean of each column
imputer = SimpleImputer(strategy='mean')
x_train_selected_imputed = imputer.fit_transform(x_train_selected)
# Create the KNeighborsClassifier
knn = KNeighborsClassifier()
# Define the hyperparameter grid for tuning
param_grid = {'n_neighbors': [3, 5, 7], 'weights': ['uniform', 'distance']}
# Perform GridSearchCV with 5-fold cross-validation
grid_search = GridSearchCV(knn, param_grid, cv=5)
grid_search.fit(x_train_selected_imputed, y_train)
# Print the best hyperparameters found
print("Best Hyperparameters:", grid_search.best_params_)
# Print the best model's score
print("Best Model Score:", grid_search.best_score_)
```

```
Best Hyperparameters: {'n_neighbors': 3, 'weights': 'distance'}
Best Model Score: 0.9213647642679901
```

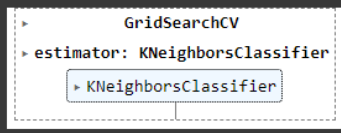
Validation Method -

```
[145] param_grid = {'n_neighbors': [3, 5, 7, 9],
                    'weights': ['uniform', 'distance'],
                    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']}
```

```
[146] from sklearn.model_selection import GridSearchCV
```

```
[147] knn = GridSearchCV (knn, param_grid, cv=5, n_jobs=-1)
```

```
knn.fit(x_train_selected_imputed, y_train)
```




```
[149] # Print the best hyperparameters and corresponding mean cross-validated score
      print("Best hyperparameters: ", knn.best_params_)
```

```
Best hyperparameters:  {'algorithm': 'auto', 'n_neighbors': 9, 'weights': 'distance'}
```

```
[150] print("Best mean cross-validated score: {:.2f}".format(knn.best_score_))
```

```
Best mean cross-validated score: 0.92
```