Bank Fraud Detection Model Report

Made by : Mandar Kelkar

Introduction

Fraudulent activities in financial systems can result in significant losses. In response, developing effective fraud detection models becomes crucial. This report outlines the development and evaluation of a machine learning model for Bank Fraud Detection using a dataset named "bank_data.csv".

Data Description

The dataset contains various features related to transactions and customer information. It includes columns such as:

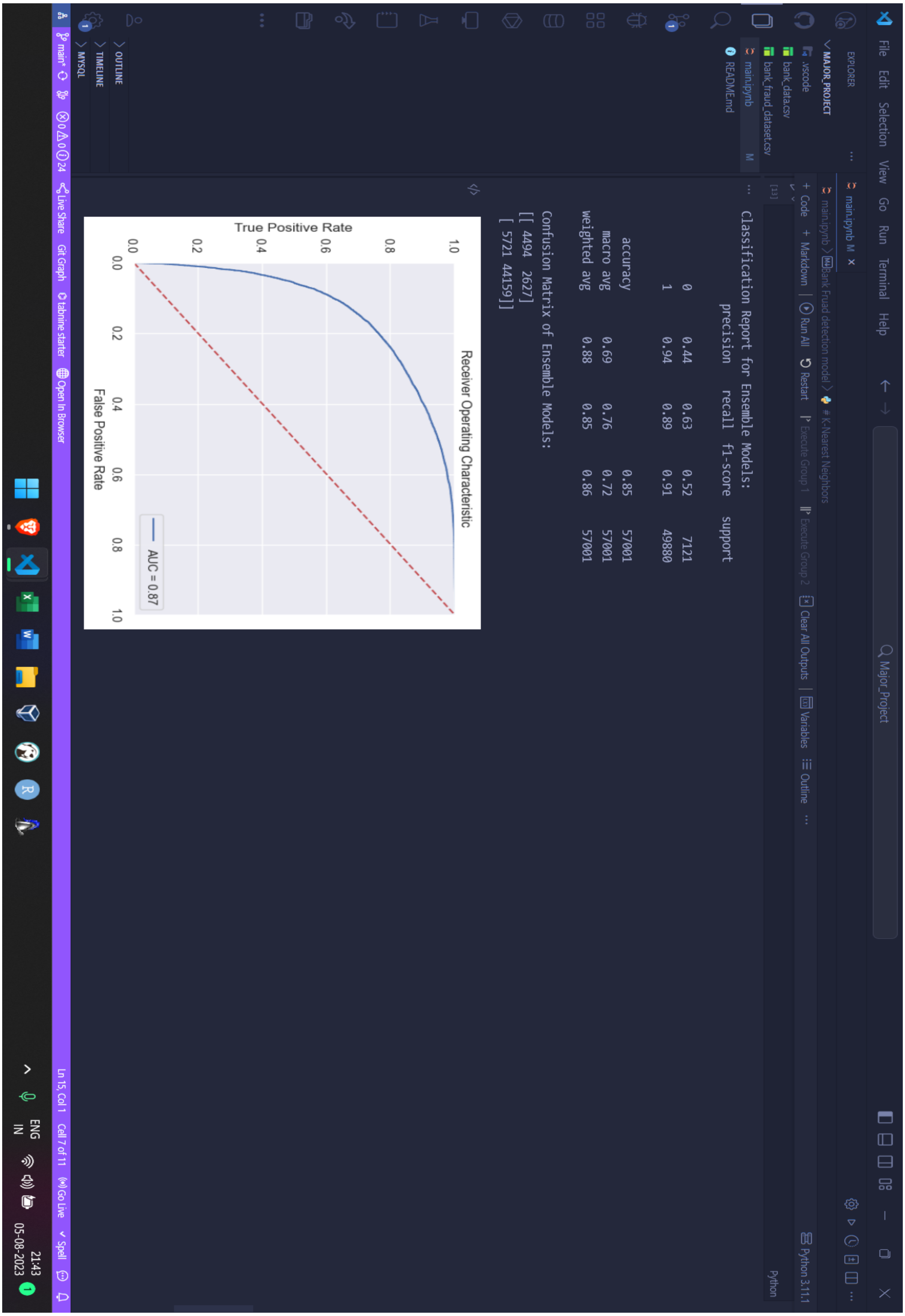
- `aon`: Age on network.
- `daily_decr30`: Average daily amount spent in the last 30 days.
- `daily_decr90`: Average daily amount spent in the last 90 days.
- `rental30`: Average rent amount in the last 30 days.
- `rental90`: Average rent amount in the last 90 days.
- ...

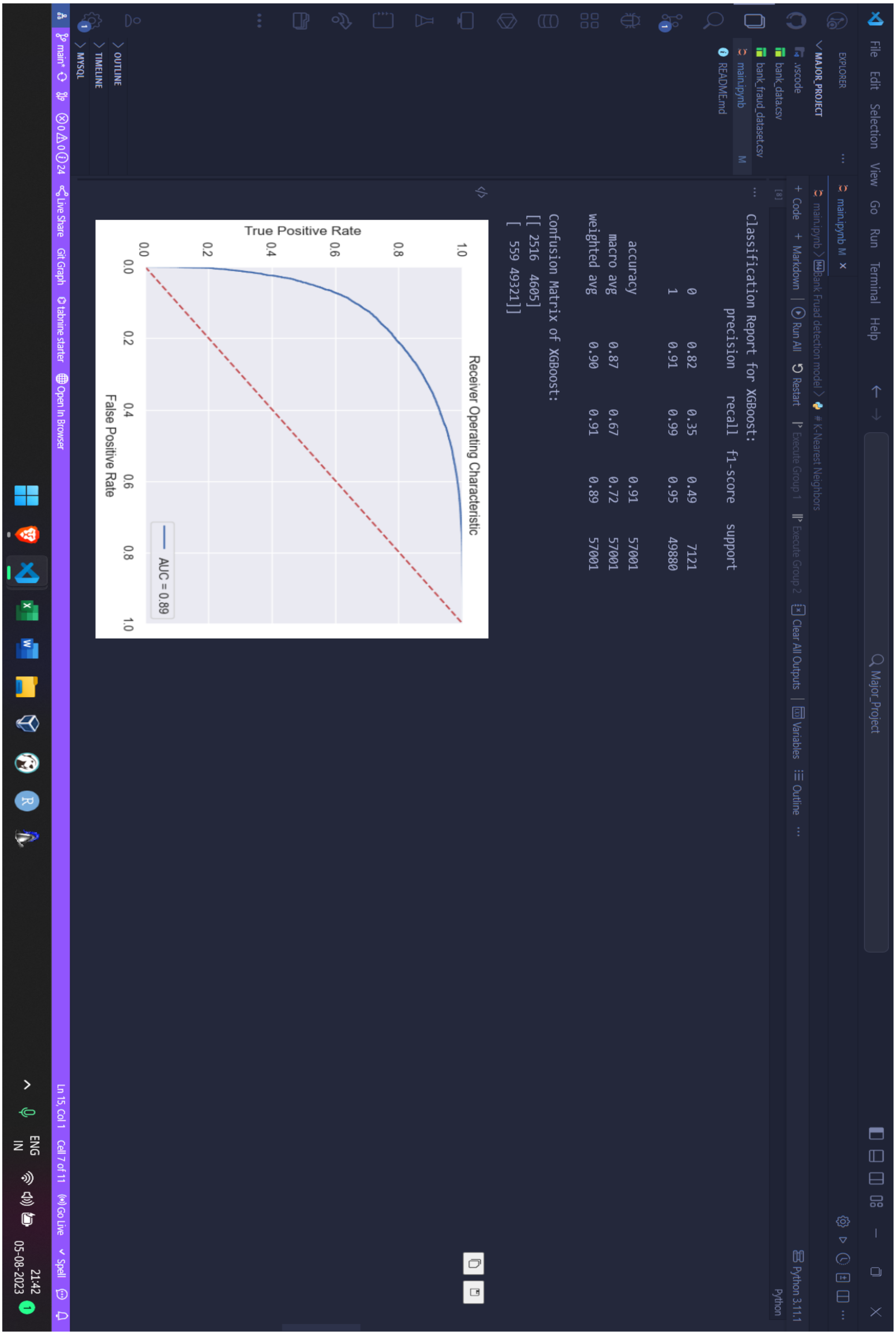
Approach

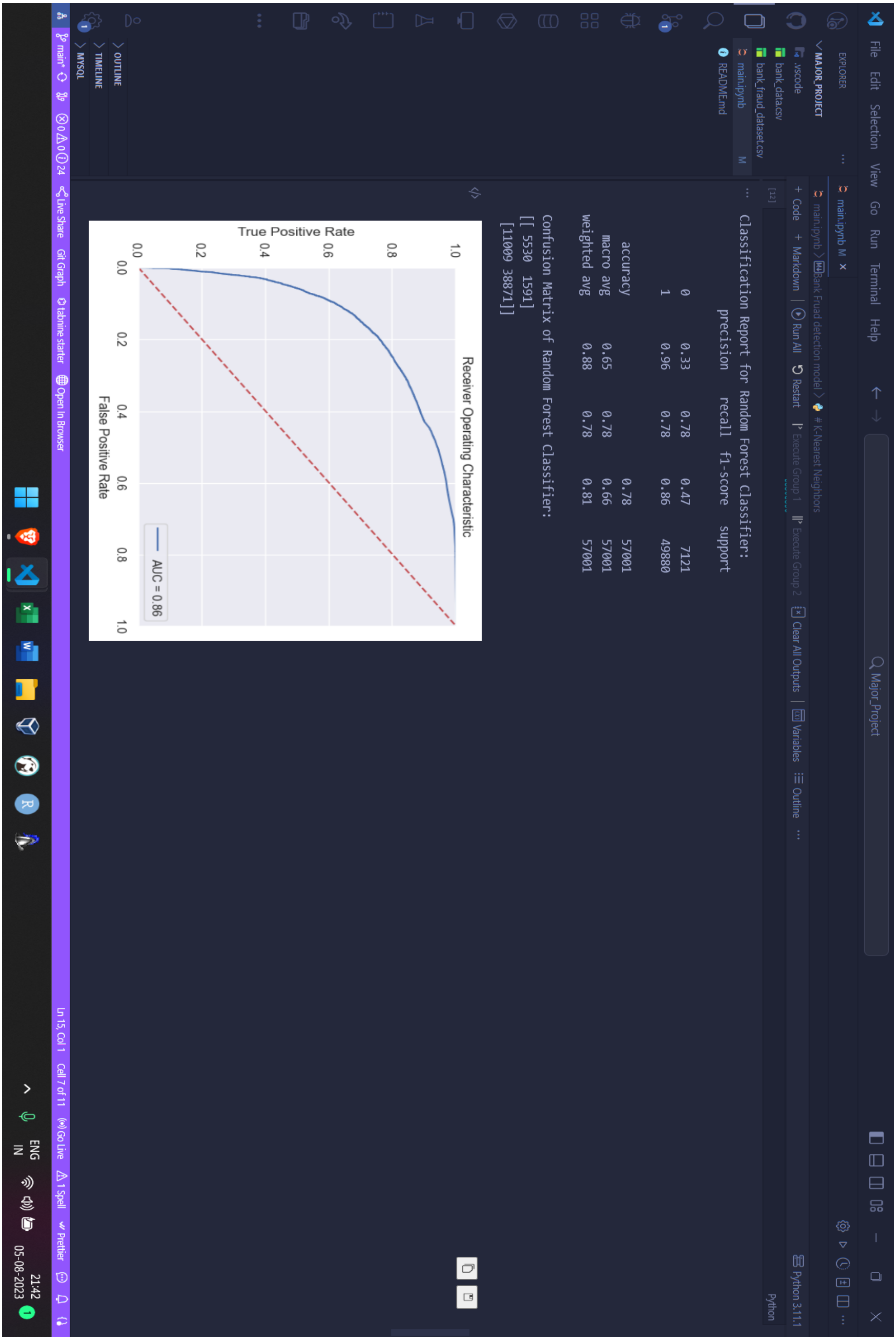
The primary approach for this project involved thorough data visualization and evaluation of performance metrics like precision, recall, and F1-score across different models.

Visualization

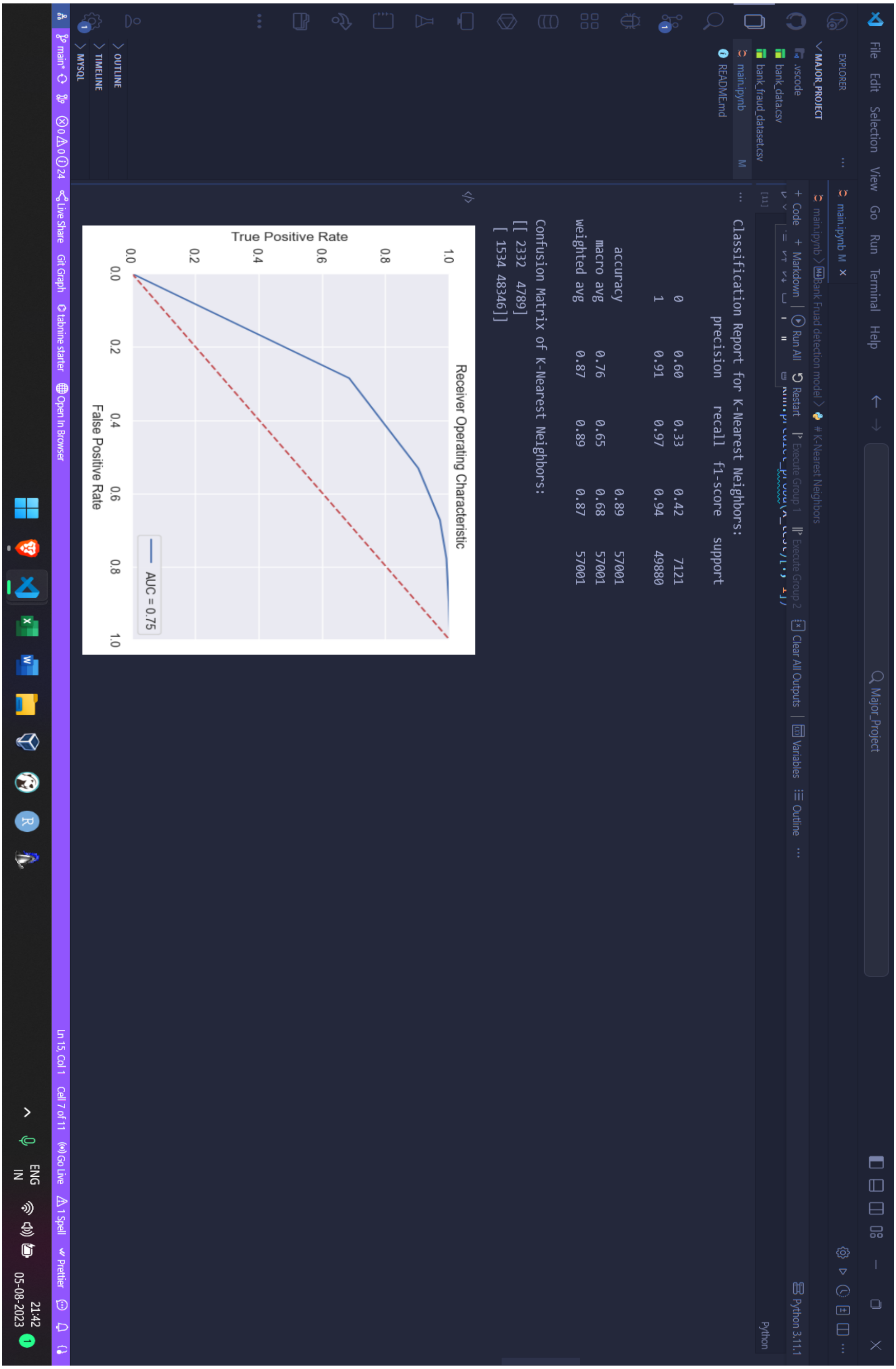
Visualization played a crucial role in understanding the data and model performance. Various visualizations were generated to explore relationships and evaluate model performance. The code for these visualizations is available in the provided codebase.


main.ipynb M
main.ipynb > Bank Fraud detection model > # K-Nearest Neighbors
EXPLORER
MAJOR_PROJECT
.vscode
bank_data.csv
bank_fraud_dataset.csv
main.ipynb
README.md
OUTLINE
TIMELINE
MYSQL
Classification Report for Ensemble Models:
precision recall f1-score support
0 0.44 0.63 0.52 7121
1 0.94 0.89 0.91 49880
accuracy 0.85 57001
macro avg 0.69 0.76 0.72 57001
weighted avg 0.88 0.85 0.86 57001
Confusion Matrix of Ensemble Models:
[[4494 2627]
[5721 44159]]
Receiver Operating Characteristic
True Positive Rate
False Positive Rate
AUC = 0.87
Python 3.11.1
21:43
05-08-2023


Classification Report for XGBoost:
precision recall f1-score support
0 0.82 0.35 0.49 7121
1 0.91 0.99 0.95 49880
accuracy 0.91 57001
macro avg 0.87 0.67 0.72 57001
weighted avg 0.90 0.91 0.89 57001
Confusion Matrix of XGBoost:
[[2516 4605]
[559 49321]]
Receiver Operating Characteristic
True Positive Rate
False Positive Rate
AUC = 0.89



Classification Report for Random Forest Classifier:
precision recall f1-score support
0 0.33 0.78 0.47 7121
1 0.96 0.78 0.86 49880
accuracy 0.78 57001
macro avg 0.65 0.78 0.66 57001
weighted avg 0.88 0.78 0.81 57001
Confusion Matrix of Random Forest Classifier:
[[5530 1591]
[11009 38871]]
Receiver Operating Characteristic
True Positive Rate
False Positive Rate
AUC = 0.86


Classification Report for K-Nearest Neighbors:
precision recall f1-score support
0 0.60 0.33 0.42 7121
1 0.91 0.97 0.94 49880
accuracy 0.89 57001
macro avg 0.76 0.65 0.68 57001
weighted avg 0.87 0.89 0.87 57001
Confusion Matrix of K-Nearest Neighbors:
[[2332 4789]
[1534 48346]]
Receiver Operating Characteristic
True Positive Rate
False Positive Rate
AUC = 0.75

Algorithms

Several classification algorithms were applied to the dataset. The following machine learning algorithms were used:

1. K-Nearest Neighbors (KNN)

```
# K-Nearest Neighbors
knn = KNeighborsClassifier(n_neighbors=5, p=1)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

print("Classification Report for K-Nearest Neighbors:")
print(classification_report(y_test, y_pred))
print("Confusion Matrix of K-Nearest Neighbors:")
print(confusion_matrix(y_test, y_pred))
plot_roc_auc(y_test, knn.predict_proba(X_test)[:, 1])
```

```
Classification Report for K-Nearest Neighbors:
              precision    recall  f1-score   support

           0       0.60      0.33      0.42      7121
           1       0.91      0.97      0.94     49880

    accuracy                           0.89     57001
   macro avg       0.76      0.65      0.68     57001
weighted avg       0.87      0.89      0.87     57001

Confusion Matrix of K-Nearest Neighbors:
[[ 2332  4789]
 [ 1534 48346]]
```

2. Random Forest Classifier

```
# Random Forest Classifier
rf_clf = RandomForestClassifier(n_estimators=100, max_depth=8, random_state=42, class_weight="balanced")
rf_clf.fit(X_train, y_train)
y_pred = rf_clf.predict(X_test)

print("Classification Report for Random Forest Classifier:")
print(classification_report(y_test, y_pred))
print("Confusion Matrix of Random Forest Classifier:")
print(confusion_matrix(y_test, y_pred))
plot_roc_auc(y_test, rf_clf.predict_proba(X_test)[:, 1])
```

```
Classification Report for Random Forest Classifier:
              precision    recall  f1-score   support

           0       0.33      0.78      0.47      7121
           1       0.96      0.78      0.86     49880

    accuracy                           0.78     57001
   macro avg       0.65      0.78      0.66     57001
weighted avg       0.88      0.78      0.81     57001

Confusion Matrix of Random Forest Classifier:
[[ 5530  1591]
 [11009 38871]]
```

3. XGBoost Classifier

```
# XGBoost
xgb_clf = xgb.XGBClassifier(max_depth=6, learning_rate=0.05, n_estimators=400, random_state=42, verbosity=1)
xgb_clf.fit(X_train, y_train)
y_pred = xgb_clf.predict(X_test)

print("Classification Report for XGBoost:")
print(classification_report(y_test, y_pred))
print("Confusion Matrix of XGBoost:")
print(confusion_matrix(y_test, y_pred))
plot_roc_auc(y_test, xgb_clf.predict_proba(X_test)[:, 1])
```

```
Classification Report for XGBoost:
              precision    recall  f1-score   support

           0       0.82      0.35      0.49      7121
           1       0.91      0.99      0.95     49880

    accuracy                           0.91     57001
   macro avg       0.87      0.67      0.72     57001
weighted avg       0.90      0.91      0.89     57001

Confusion Matrix of XGBoost:
[[ 2516  4605]
 [  559 49321]]
```

4. Ensemble Model (Voting Classifier)

```
# Ensemble
estimators = [("KNN", knn), ("RF", rf_clf), ("XGB", xgb_clf)]
ensemble = VotingClassifier(estimators=estimators, voting="soft", weights=[1, 4, 1])
ensemble.fit(X_train, y_train)
y_pred = ensemble.predict(X_test)

print("Classification Report for Ensemble Models:")
print(classification_report(y_test, y_pred))
print("Confusion Matrix of Ensemble Models:")
print(confusion_matrix(y_test, y_pred))
plot_roc_auc(y_test, ensemble.predict_proba(X_test)[:, 1])
```

```
Classification Report for Ensemble Models:
              precision    recall  f1-score   support

           0       0.44      0.63      0.52      7121
           1       0.94      0.89      0.91     49880

    accuracy                           0.85     57001
   macro avg       0.69      0.76      0.72     57001
weighted avg       0.88      0.85      0.86     57001

Confusion Matrix of Ensemble Models:
[[ 4494  2627]
 [ 5721 44159]]
```

Code:

All required modules import pandas as pd import numpy as np import seaborn as sns import matplotlib.pyplot as plt from sklearn.model_selection import train_test_split from sklearn.metrics import confusion_matrix, classification_report from sklearn.metrics import roc_curve, auc from sklearn.preprocessing import LabelEncoder import xgboost as xgb from sklearn.neighbors import KNeighborsClassifier from sklearn.ensemble import RandomForestClassifier from sklearn.ensemble import VotingClassifier # Set seaborn style for better visualizations sns.set() # Read and plot the dataset data = pd.read_csv("bank_data.csv") data.head(5) # Preprocessing data_reduced = data.drop(['msisdn', 'pdate'], axis=1) label_encoder = LabelEncoder() for col in data_reduced.columns: if data_reduced[col].dtype == 'object': data_reduced[col] = label_encoder.fit_transform(data_reduced[col]) X = data_reduced.drop(['label'], axis=1) y = data['label'] # Split the data into training and testing sets X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, shuffle=True, stratify=y) # Function for plotting ROC-AUC curve def plot_roc_auc(y_test, preds): fpr, tpr, threshold = roc_curve(y_test, preds) roc_auc = auc(fpr, tpr) plt.title('Receiver Operating Characteristic') plt.plot(fpr, tpr, 'b', label='AUC = %0.2f' % roc_auc) plt.legend(loc='lower right') plt.plot([0, 1], [0, 1], 'r--') plt.xlim([0, 1]) plt.ylim([0, 1]) plt.ylabel('True Positive Rate') plt.xlabel('False Positive Rate') plt.show() # K-Nearest Neighbors knn = KNeighborsClassifier(n_neighbors=5, p=1) knn.fit(X_train, y_train) y_pred = knn.predict(X_test) print("Classification Report for K-Nearest Neighbors:") print(classification_report(y_test, y_pred)) print("Confusion Matrix of K-Nearest Neighbors:") print(confusion_matrix(y_test, y_pred)) plot_roc_auc(y_test, knn.predict_proba(X_test)[:, 1]) # Random Forest Classifier rf_clf = RandomForestClassifier(n_estimators=100, max_depth=8, random_state=42, class_weight="balanced") rf_clf.fit(X_train, y_train) y_pred = rf_clf.predict(X_test) print("Classification Report for Random Forest Classifier:") print(classification_report(y_test, y_pred)) print("Confusion Matrix of Random Forest Classifier:") print(confusion_matrix(y_test, y_pred)) plot_roc_auc(y_test, rf_clf.predict_proba(X_test)[:, 1]) # XGBoost xgb_clf = xgb.XGBClassifier(max_depth=6, learning_rate=0.05, n_estimators=400,

random_state=42, verbosity=1) xgb_clf.fit(X_train, y_train) y_pred = xgb_clf.predict(X_test) print("Classification Report for XGBoost:") print(classification_report(y_test, y_pred)) print("Confusion Matrix of XGBoost:") print(confusion_matrix(y_test, y_pred)) plot_roc_auc(y_test, xgb_clf.predict_proba(X_test)[:, 1]) # Ensemble estimators = [("KNN", knn), ("RF", rf_clf), ("XGB", xgb_clf)] ensemble = VotingClassifier(estimators=estimators, voting="soft", weights=[1, 4, 1]) ensemble.fit(X_train, y_train) y_pred = ensemble.predict(X_test) print("Classification Report for Ensemble Models:") print(classification_report(y_test, y_pred)) print("Confusion Matrix of Ensemble Models:") print(confusion_matrix(y_test, y_pred)) plot_roc_auc(y_test, ensemble.predict_proba(X_test)[:, 1])

Evaluation

Model performance was assessed using key metrics such as F1-score, accuracy, precision, and recall. These metrics were utilized to compare and contrast the performance of different models.

Result and Discussion

The evaluation results highlighted distinct strengths and weaknesses of each model. XGBoost exhibited the highest accuracy and F1-score, indicating its robustness in identifying fraudulent transactions. The ensemble model provided a balanced approach, showcasing good precision and recall trade-off.

Output:

Classification Report for Ensemble Models:

	precision	recall	f1-score	support
0	0.44	0.63	0.52	7121
1	0.94	0.89	0.91	49880
accuracy			0.85	57001
macro avg	0.69	0.76	0.72	57001
weighted avg	0.88	0.85	0.86	57001

Confusion Matrix of Ensemble Models:

[[4494 2627]
[5721 44159]]

Classification Report for XGBoost:

	precision	recall	f1-score	support
0	0.82	0.35	0.49	7121
1	0.91	0.99	0.95	49880
accuracy			0.91	57001
macro avg	0.87	0.67	0.72	57001
weighted avg	0.90	0.91	0.89	57001

Confusion Matrix of XGBoost:
[[2516 4605]
[559 49321]]

Classification Report for Random Forest Classifier:

	precision	recall	f1-score	support
0	0.33	0.78	0.47	7121
1	0.96	0.78	0.86	49880
accuracy			0.78	57001
macro avg	0.65	0.78	0.66	57001
weighted avg	0.88	0.78	0.81	57001

Confusion Matrix of Random Forest Classifier:
[[5530 1591]
[11009 38871]]

Classification Report for Random Forest Classifier:

	precision	recall	f1-score	support
0	0.33	0.78	0.47	7121
1	0.96	0.78	0.86	49880
accuracy			0.78	57001
macro avg	0.65	0.78	0.66	57001
weighted avg	0.88	0.78	0.81	57001

Confusion Matrix of Random Forest Classifier:
[[5530 1591]
[11009 38871]]

Classification Report for K-Nearest Neighbors:

	precision	recall	f1-score	support
0	0.60	0.33	0.42	7121
1	0.91	0.97	0.94	49880
accuracy			0.89	57001
macro avg	0.76	0.65	0.68	57001
weighted avg	0.87	0.89	0.87	57001

Confusion Matrix of K-Nearest Neighbors:
[[2332 4789]
[1534 48346]]

Conclusion

In conclusion, the XGBoost model demonstrated superior performance in identifying fraudulent activities within financial transactions. The ensemble model also emerged as a viable option. The insights gained from this project contribute to enhancing fraud detection strategies in financial systems.

Future Work

Future work involves refining the models further, exploring advanced techniques, and utilizing additional features to improve detection accuracy. The models' performance could be further evaluated using more extensive and diverse datasets.

References

1. The dataset "bank_data.csv" (provide data source reference)

2. Additional resources, papers, and websites consulted during the project (