**Bank Fraud**

**Detection Model**

**Report**

**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.

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**Algorithms**

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

1. K-Nearest Neighbors (KNN)



1. Random Forest Classifier

A screen shot of a computer program

Description automatically generated

1. XGBoost Classifier

A screen shot of a computer program

Description automatically generated

1. Ensemble Model (Voting Classifier)

A screenshot of a computer program

Description automatically generated

**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 (