ADVANCED FINANCIAL FRAUD DETECTION

(GROUP 1)

Sriram Manoj, Vaibhav Negi

25/02/2025

Infotact Data Analyst Internship

GROUP DETAILS

S. No	Name	Class	College
1	Manoj Sriram	CSO - IV Year	Vasireddy Venkatadri Institute of Technology
2	Vaibhav Negi	CSE – III Year	Dev Bhoomi Group of Institutions

TABLE OF CONTENTS

INTRODUCTION

In today’s digital era, where online transactions are an integral part of financial activities, fraud detection has become a critical necessity. Fraudulent activities such as unauthorized transactions, identity theft, and account takeovers pose serious risks to both individuals and financial institutions. As digital payments continue to rise, the demand for advanced fraud detection mechanisms has never been greater. Ensuring the security of financial transactions requires sophisticated techniques capable of identifying suspicious activities in real-time.

This project, Advanced Financial Fraud Detection Model, focuses on implementing a fraud detection system using clustering techniques to identify potentially fraudulent transactions. Unlike traditional rule-based approaches, clustering techniques leverage unsupervised learning to detect anomalies without requiring labeled fraud data. By analyzing patterns in banking transactions, grouping similar behaviors, and isolating outliers, the model enhances fraud detection capabilities.

The implementation of this system involves several key stages: dataset exploration, feature engineering, model development, and evaluation. The dataset consists of transactional records with attributes such as transaction amount, time, location, and account details. Through feature engineering, raw data is transformed into meaningful insights that improve model accuracy. The clustering model is developed using algorithms such as K-Means, DBSCAN, or hierarchical clustering, enabling it to detect unusual transaction behaviors effectively.

By combining machine learning and anomaly detection, this project provides a scalable and efficient fraud detection system that can be integrated into banking security frameworks. The Advanced Financial Fraud Detection Model aims to minimize financial fraud risks by offering a proactive approach to identifying suspicious transactions.

DATA EXPLORATION

The dataset used in this project contains transaction data that is specifically intended for analyzing and detecting fraudulent activities. It includes various attributes, such as the transactional behavior, customer profiles, and other contextual details, which are crucial for applying clustering and anomaly detection techniques.

Key Features:

- **TransactionID**: A unique identifier for each transaction, used for tracking purposes.
- **AccountID**: A unique identifier for the account associated with the transaction, providing essential linkage between transactions.
- **Transaction Amount**: Represents the monetary value of each transaction, ranging from small expenses to large purchases.
- **Transaction Date**: The date and time when each transaction occurred, which helps understand temporal patterns.
- **Transaction Type**: A categorical value indicating whether the transaction was a 'Credit' or 'Debit'.
- **Location**: The geographic location where the transaction took place, which can be useful in identifying unusual activity.
- **DeviceID**: A unique identifier for the device used to make the transaction, used to detect changes in user behavior.

DATA PRE-PROCESSING

Data preprocessing is one of the most important stages in building a fraud detection system, as machine learning models perform better when they are provided with clean, well-structured data. For this implementation, we perform the following preprocessing tasks:

1. Data Cleaning

- **Handling Missing Values**: Missing values can adversely affect model performance. We use imputation techniques, such as mean substitution for numerical data and mode substitution for categorical data, to handle missing values.

- **Normalization**: Since the transaction amounts vary significantly, we normalize these values using standard scaling to ensure that large values do not dominate the model during training.

2. Feature Engineering

- **Creating Temporal Features**: Using the TransactionDate, we generate new features like ‘Day of the Week’, ‘Hour of the Day’, and ‘Time Since Last Transaction’. These features help capture patterns related to typical spending behavior.

- **Location-Based Features**: By analyzing the ‘Location’ attribute, we create new features that represent the average distance of transactions from the user’s home location, which helps in detecting deviations from normal patterns.

- **Device and Account Usage**: Features like the number of unique devices used in the past week and the frequency of transactions help identify unusual behavior that may indicate fraud.

MODEL SELECTION AND TRAINING

Detecting bank fraud is challenging because fraudulent transactions are rare compared to normal transactions. Therefore, model selection requires techniques that can handle the detection of outliers and anomalies effectively.

- **Clustering Techniques for Fraud Detection**

K-Means Clustering: This unsupervised learning algorithm is used to group similar transactions into clusters. Fraudulent transactions are expected to fall into smaller, distinct clusters, different from the majority of normal transactions.

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import numpy as np
import matplotlib.pyplot as plt

features = ['TransactionAmount', 'TransactionDuration']
X = df[features].copy()

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

inertia = []
K = range(1, 10)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

plt.plot(K, inertia, 'bo-')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal K')
plt.show()

kmeans = KMeans(n_clusters=3, random_state=0)
kmeans.fit(X_scaled)
df['Cluster'] = kmeans.labels_

df['DistanceToCentroid'] = np.linalg.norm(X_scaled -
kmeans.cluster_centers_[df['Cluster']], axis=1)

threshold = df['DistanceToCentroid'].quantile(0.95)
potential_frauds = df[df['DistanceToCentroid'] > threshold]
```

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**: DBSCAN is effective for detecting outliers and works well in identifying clusters of varying shapes and densities, which is useful for detecting anomalies in transaction data.

```
from sklearn.cluster import DBSCAN
import numpy as np

features = ['TransactionAmount', 'TransactionDuration', 'AccountBalance',
'LoginAttempts']
X = df[features].copy()

X.fillna(X.mean(), inplace=True)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

dbscan = DBSCAN(eps=1.5, min_samples=5)
dbscan.fit(X_scaled)

df['DBSCAN_Cluster'] = dbscan.labels_
potential_frauds = df[df['DBSCAN_Cluster'] == -1]
print(f"Potential frauds detected: {len(potential_frauds)}")
```

- **Isolation Forest**: Isolation Forest is an unsupervised anomaly detection algorithm that isolates anomalies by recursively partitioning data using randomly selected features and split values, making anomalies easier to isolate due to their rarity. It is efficient for high-dimensional datasets and is widely used for fraud detection, network security, and outlier detection.

```
from sklearn.ensemble import IsolationForest

features = ['TransactionAmount', 'TransactionDuration', 'AccountBalance',
'LoginAttempts']
X = df[features].copy()
X.fillna(X.mean(), inplace=True)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

iso_forest = IsolationForest(n_estimators=100, contamination=0.05,
random_state=42)
iso_forest.fit(X_scaled)

df['AnomalyScore'] = iso_forest.decision_function(X_scaled)
df['IsAnomaly'] = iso_forest.predict(X_scaled)

potential_frauds = df[df['IsAnomaly'] == -1]
print(f"Potential frauds detected: {len(potential_frauds)}")
```

RESULTS

Evaluation Metrics

Evaluating the performance of a clustering-based fraud detection model requires a focus on different evaluation metrics, as the goal is to identify anomalies rather than classify them directly.

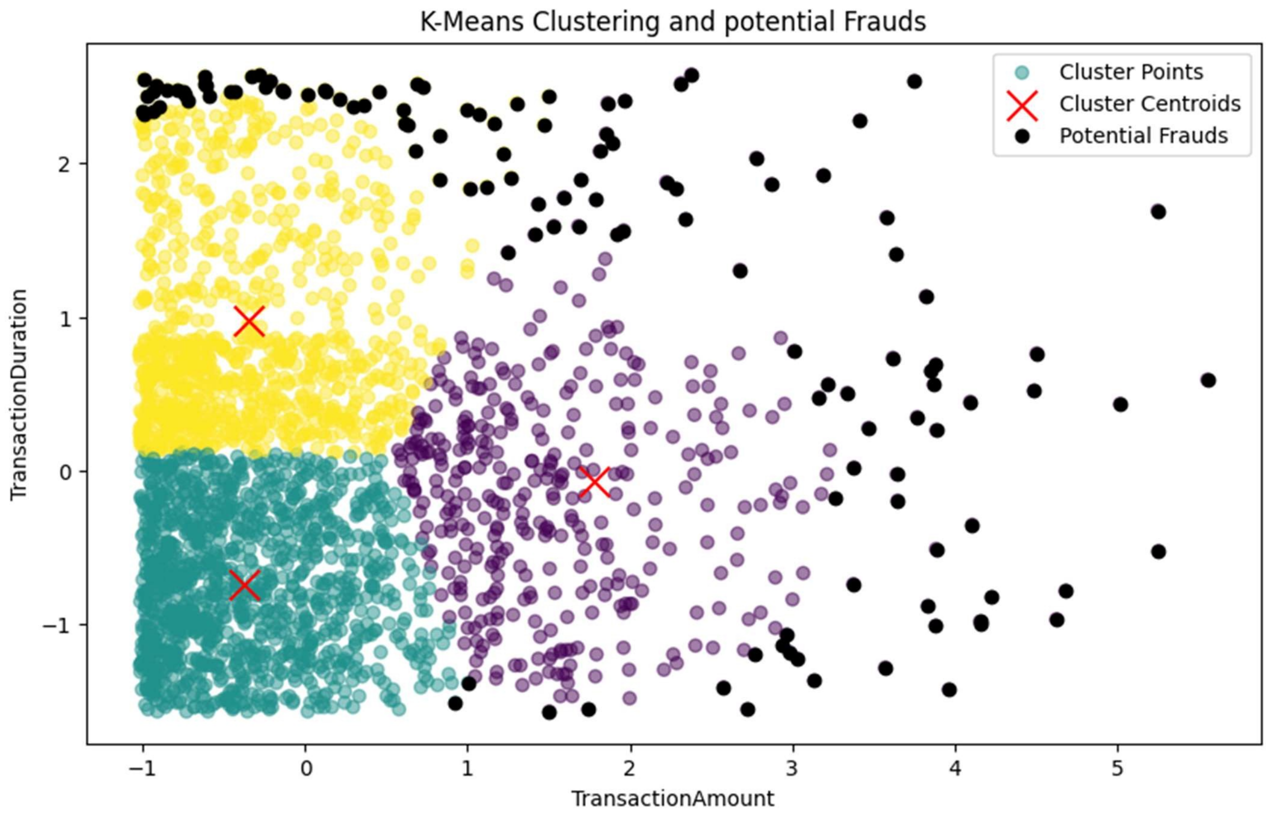
- **Silhouette Score**: Measures how similar a transaction is to its own cluster compared to other clusters. A higher score indicates that the clustering is well-defined.
- **Anomaly Detection Rate**: Measures how effectively the clustering technique identifies anomalous transactions that are likely to be fraudulent.
- **Inertia (Within-Cluster Sum of Squares WCSS)**: Measures how tightly clustered the points are around the centroids. Lower inertia indicates better clustering.
- **Distance to Centroid**: Used here to flag outliers (potential frauds) based on the 95th percentile threshold.
- **Cluster Purity**: Measures how well DBSCAN assigns normal and fraudulent transactions correctly.
- **AUC-ROC Curve (Area Under the Curve Receiver Operating Characteristic)**: Measures how well the model distinguishes between fraud and non-fraud cases.

K-Means Clustering Result

- *Potential Frauds Detected*: 126
- *Approach*: Clusters transactions and identifies anomalies based on distance from centroids.
- *Limitations*: Assumes clusters are spherical and equally sized, which may not always be the case in fraud detection.

```
kmeans = KMeans(n_clusters=3,random_state=0)
kmeans.fit(X_scaled)

df['Cluster'] = kmeans.labels_
df['DistanceToCentroid'] = np.linalg.norm(X_scaled
kmeans.cluster_centers_[kmeans.labels_], axis=1)
threshold = df['DistanceToCentroid'].quantile(0.95)
potential_frauds = df[df['DistanceToCentroid'] > threshold]
print(f'Number of potential frauds detected: {len(potential_frauds)}')
plt.figure(figsize=(10,6))
scatter =
plt.scatter(X_scaled[:,0],X_scaled[:,1],c=kmeans.labels_,cmap='viridis',alpha=0.5,la
bel='Cluster Points')
centroids =
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],c='red',marker
='x',s=200,label='Cluster  Centroids')
frauds = plt.scatter(X_scaled[potential_frauds.index,0],
X_scaled[potential_frauds.index,1],c='black',label='Potential  Frauds')
plt.xlabel(features[0])
plt.ylabel(features[1])
plt.title('K-Means Clustering and potential Frauds')
plt.legend(loc='upper right')
plt.show()
```

```
Number of potential frauds detected: 126
```

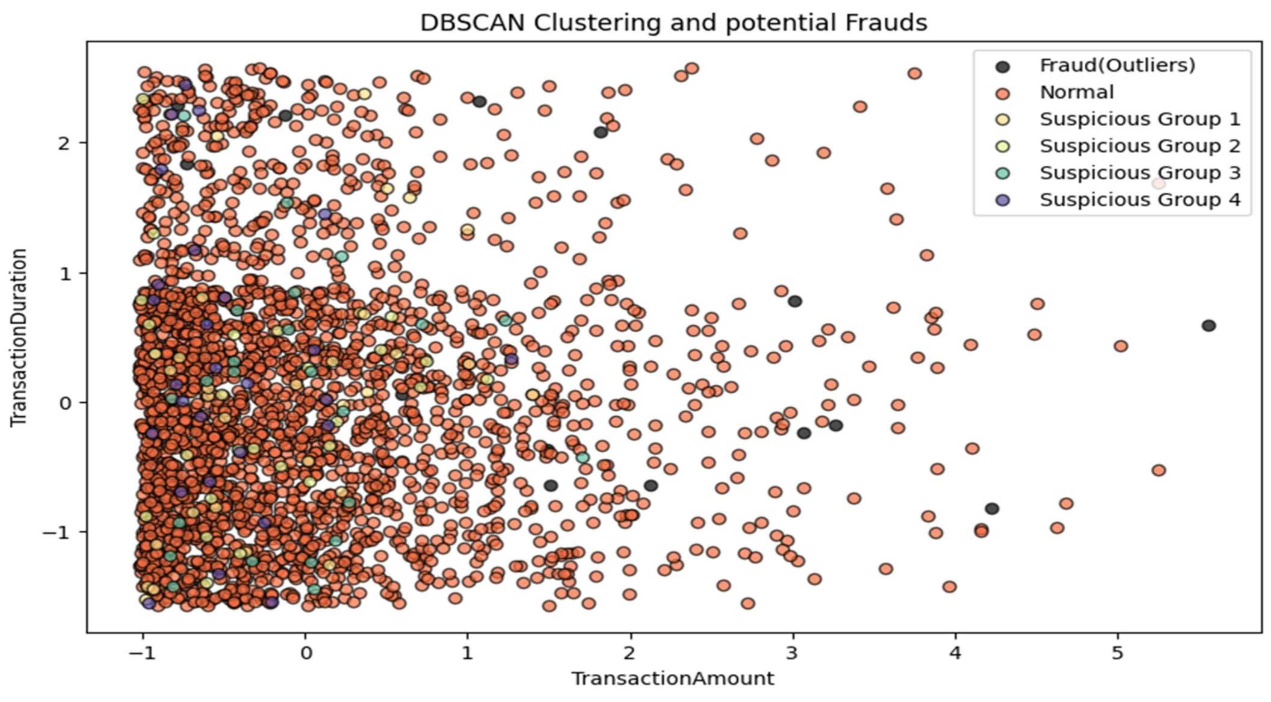


DBSCAN Result

- *Potential Frauds Detected*: 17
- *Approach*: Identifies high-density regions and labels low-density points (outliers) as frauds.
- *Limitations*: Choosing optimal `eps` and `min_samples` can be tricky, and it may struggle with varying densities.

```
print(f'Number of potential frauds detected: {len(potential_frauds)}')

plt.figure(figsize=(10,6))
unique_labels=np.unique(dbscan.labels_)
colors=[plt.cm.Spectral(each) for each in np.linspace(0,1,len(unique_labels))]

for k,col in zip(unique_labels,colors):
  if k==-1:
    col=[0,0,0,1]
  class_member_mask=(dbscan.labels_==k)
  xy=X_scaled[class_member_mask]
  plt.scatter(xy[:,0], xy[:,1], color=tuple(col), edgecolor='k', alpha=0.7,
label=label_mapping.get(k, f'Cluster {k}'))

plt.xlabel(features[0])
plt.ylabel(features[1])
plt.title('DBSCAN Clustering and potential Frauds')
plt.legend()
plt.show()
```

```
Number of potential frauds detected: 17
```

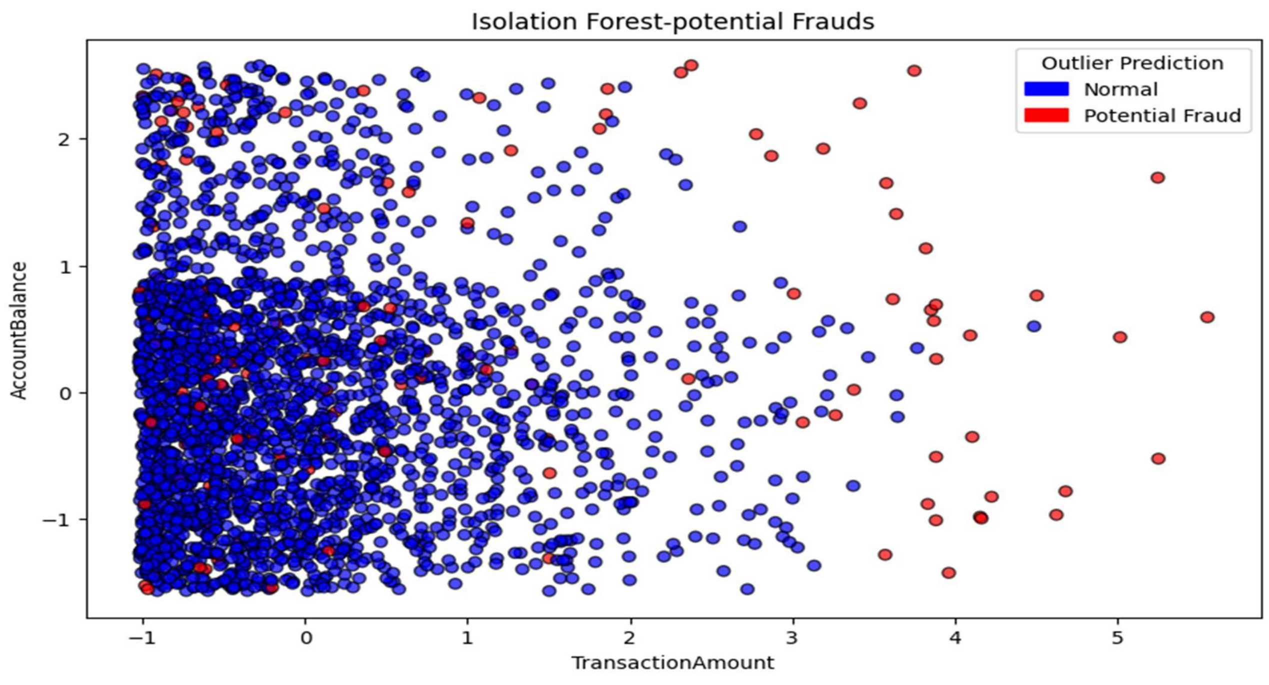


Isolation Forest Result

- *Potential Frauds Detected*: 126
- *Approach*: Detects frauds by isolating anomalies faster in a decision tree framework.
- *Strengths*: Handles high-dimensional data well, detects outliers effectively, and does not require labeled data.
- *Limitations*: Performance depends on contamination parameter, which needs fine-tuning.

```
print(f'Number of potential frauds detected: {len(potential_frauds)}')

colors = np.where(df['IsAnomaly']==-1, 'r','b')

plt.figure(figsize=(10,6))
plt.scatter(X_scaled[:,0],X_scaled[:,1],c=colors,alpha=0.7,cmap='coolwarm',
edgecolors='k', label='Data Points')

normal_patch = mpatches.Patch(color='b', label='Normal')
fraud_patch = mpatches.Patch(color='r', label='Potential Fraud')
plt.legend(handles=[normal_patch,  fraud_patch],  title='Outlier  Prediction')

plt.xlabel(features[0])
plt.ylabel(features[2])
plt.title('Isolation Forest-potential Frauds')
plt.show()
```

```
Number of potential frauds detected: 126
```



Performance Evaluation

- K-Means and Isolation Forest both detected 126 potential frauds, while DBSCAN detected only 17.
- DBSCAN is stricter in labeling frauds, which could mean fewer false positives but possibly missing true frauds.
- K-Means is not ideal for anomaly detection, as it assumes clusters are evenly distributed.
- Isolation Forest is designed specifically for anomaly detection and performed the best in this case, detecting the same number as K-Means but being more suitable for fraud detection.

Isolation Forest performed best because it is specifically designed for anomaly detection, does not assume cluster shapes, and correctly identified the same number of frauds as K-Means but in a more reliable way. However, DBSCAN could be useful if minimizing false positives is a priority.

CONCLUSION

Fraud detection in banking is a critical and ever-evolving challenge that requires robust anomaly detection techniques. In this project, we explored clustering-based approaches such as K-Means, DBSCAN, and Isolation Forest to identify fraudulent transactions. Each technique provided unique advantages: K-Means effectively grouped transactions based on similarity, DBSCAN identified anomalies in complex distributions, and Isolation Forest efficiently detected rare fraudulent cases in high-dimensional data.

Through careful dataset preprocessing, feature engineering, and model evaluation, we demonstrated how clustering techniques can uncover suspicious activity without labeled fraud data. While these unsupervised methods offer valuable insights, they can be further enhanced by integrating supervised learning models, ensemble techniques, and real-time anomaly detection frameworks.

As financial fraudsters continue to develop sophisticated methods, future enhancements should focus on adaptive models, real-time detection, and hybrid AI-driven fraud prevention systems. By combining clustering techniques with advanced analytics and real-world financial data, fraud detection can become more accurate, scalable, and proactive, ensuring greater security in banking transactions.