CREDIT CARD FRAUD DETECTION

Submitted by

Joes Arockiam X – 2133020

Sanjay V V – 2133038

Submitted to:
Dr. Yamuna Devi N
Ms. Aruna A G
Ms. Rathika J

Department of Computing

COIMBATORE INSTITUTE OFTECHNOLOGY
(Government Aided Autonomous Institution)
Coimbatore-641014

Aim:

This project aims to develop an advanced credit card fraud detection system using predictive analysis techniques. By leveraging data analytics algorithms, the proposed solution aims to enhance security measures and mitigate financial losses associated with credit card fraud. The project will focus on building a robust predictive model that can effectively identify fraudulent transactions in real-time and minimizing false positives.

Abstract:

Credit card fraud is a major problem for banks and credit card companies. In 2022, the global cost of credit card fraud is estimated to be $28.6 billion. Fraudsters use a variety of methods to steal credit cardinformation, including phishing, skimming, and card cloning. One way to combat credit card fraud is to use machine learning techniques to identify fraudulent transactions. Time series analysis and clustering are two machine learning techniques that can be used for fraud detection.

Principal Component Analysis:

- PCA can be a useful tool for credit card fraud detection by reducing the dimensionality of largedataset and preserving the important pattern or relationships in the data
- This also helps improve the accuracy of the Machine learning model to detect fraudulenttransaction

Time Series:

- Time series analysis is a statistical technique that is used to study the behavior of data overtime. It can be used to identify patterns in data that can be used to predict future behavior.

- Time series analysis can be used to identify patterns in fraudulent transactions. For example, fraudsters often use stolen credit cards to make small, high-frequency transactions. By identifying these patterns, it is possible to flag suspicious transactions for further investigation.

Clustering:

- Clustering is a machine learning technique that is used to group data points together based on their similarity. In the context of credit card fraud, clustering can be used to group transactions together based on their risk of being fraudulent.

- Transactions that are made from different countries, at different times of day, and for different amounts of money are more likely to be fraudulent. By clustering transactions together based on these factors, it is possible to identify fraudulent transactions more easily.

Discriminant Analysis:

- Discriminant analysis, also known as linear discriminant analysis (LDA), is a statistical technique used for classifying or categorizing objects into different groups based on their features or attributes. It aims to find a linear combination of features that maximally separates the classes or groups.
- Discriminant analysis is a statistical technique that can be used for credit card fraud detection. It is a supervised learning method that aims to find a linear combination of features that maximally separates different classes or groups

CONCEPT:

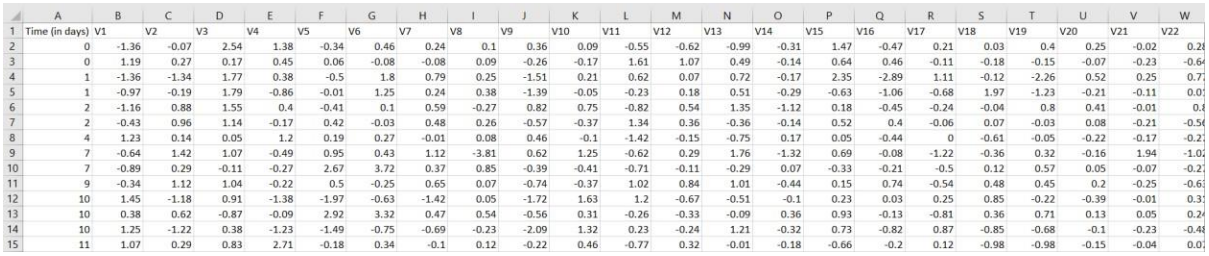
The project idea revolves around developing an advanced credit card fraud detection system using predictive analysis techniques. Credit card fraud poses a significant challenge to financial institutions and individuals, leading to substantial financial losses and security risks. The objective of this project is to leverage data analytics algorithms to build a robust predictive model that can effectively identify fraudulent transactions in real-time and minimize false positives.

The concept involves analysing historical credit card transaction data and extracting meaningful patterns, trends, and anomalies associated with fraudulent activities. By utilizing predictive analysis techniques, such as machine learning algorithms, the project aims to train a model that can learn from past fraudulent transactions and accurately predict the likelihood of a transaction being fraudulent.

The ultimate goal of the project is to develop a highly accurate and efficient credit card fraud detection system that can effectively identify and prevent fraudulent transactions while minimizing false positives. This would contribute to enhancing the security measures in the financial industry, protecting individuals and organizations from financial losses, and improving overall trust in credit card transactions.

DATASET:

The dataset is downloaded from the Kaggle.com – "Credit Card Fraudulent"



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Time (in days)	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22
2	0	-1.36	-0.07	2.54	1.38	-0.34	0.46	0.24	0.1	0.36	0.09	-0.55	-0.62	-0.99	-0.31	1.47	-0.47	0.21	0.03	0.4	0.25	-0.02	0.2
3	0	1.19	0.27	0.17	0.45	0.06	-0.08	-0.08	0.09	-0.26	-0.17	1.61	1.07	0.49	-0.14	0.64	0.46	-0.11	-0.18	-0.15	-0.07	-0.23	-0.6
4	1	-1.36	-1.34	1.77	0.38	-0.5	1.8	0.79	0.25	-1.51	0.21	0.62	0.07	0.72	-0.17	2.35	-2.89	1.11	-0.12	-2.26	0.52	0.25	0.7
5	1	-0.97	-0.19	1.79	-0.86	-0.01	1.25	0.24	0.38	-1.39	-0.05	-0.23	0.18	0.51	-0.29	-0.63	-1.06	-0.68	1.97	-1.23	-0.21	-0.11	0.0
6	2	-1.16	0.88	1.55	0.4	-0.41	0.1	0.59	-0.27	0.82	0.75	-0.82	0.54	1.35	-1.12	0.18	-0.45	-0.24	-0.04	0.8	0.41	-0.01	0.8
7	2	-0.43	0.96	1.14	-0.17	0.42	-0.03	0.48	0.26	-0.57	-0.37	1.34	0.36	-0.36	-0.14	0.52	0.4	-0.06	0.07	-0.03	0.08	-0.21	-0.5
8	4	1.23	0.14	0.05	1.2	0.19	0.27	-0.01	0.08	0.46	-0.1	-1.42	-0.15	-0.75	0.17	0.05	-0.44	0	-0.61	-0.05	-0.22	-0.17	-0.2
9	7	-0.64	1.42	1.07	-0.49	0.95	0.43	1.12	-3.81	0.62	1.25	-0.62	0.29	1.76	-1.32	0.69	-0.08	-1.22	-0.36	0.32	-0.16	1.94	-1.0
10	7	-0.89	0.29	-0.11	-0.27	2.67	3.72	0.37	0.85	-0.39	-0.41	-0.71	-0.11	-0.29	0.07	-0.33	-0.21	-0.5	0.12	0.57	0.05	-0.07	-0.2
11	9	-0.34	1.12	1.04	-0.22	0.5	-0.25	0.65	0.07	-0.74	-0.37	1.02	0.84	1.01	-0.44	0.15	0.74	-0.54	0.48	0.45	0.2	-0.25	-0.6
12	10	1.45	-1.18	0.91	-1.38	-1.97	-0.63	-1.42	0.05	-1.72	1.63	1.2	-0.67	-0.51	-0.1	0.23	0.03	0.25	0.85	-0.22	-0.39	-0.01	0.3
13	10	0.38	0.62	-0.87	-0.09	2.92	3.32	0.47	0.54	-0.56	0.31	-0.26	-0.33	-0.09	0.36	0.93	-0.13	-0.81	0.36	0.71	0.13	0.05	0.2
14	10	1.25	-1.22	0.38	-1.23	-1.49	-0.75	-0.69	-0.23	-2.09	1.32	0.23	-0.24	1.21	-0.32	0.73	-0.82	0.87	-0.85	-0.68	-0.1	-0.23	-0.4
15	11	1.07	0.29	0.83	2.71	-0.18	0.34	-0.1	0.12	-0.22	0.46	-0.77	0.32	-0.01	-0.18	-0.66	-0.2	0.12	-0.98	-0.98	-0.15	-0.04	0.0

PACKAGES USED:

- Numpy
- Pandas
- Statsmodels
- Sklearn
- Matplotlib.py

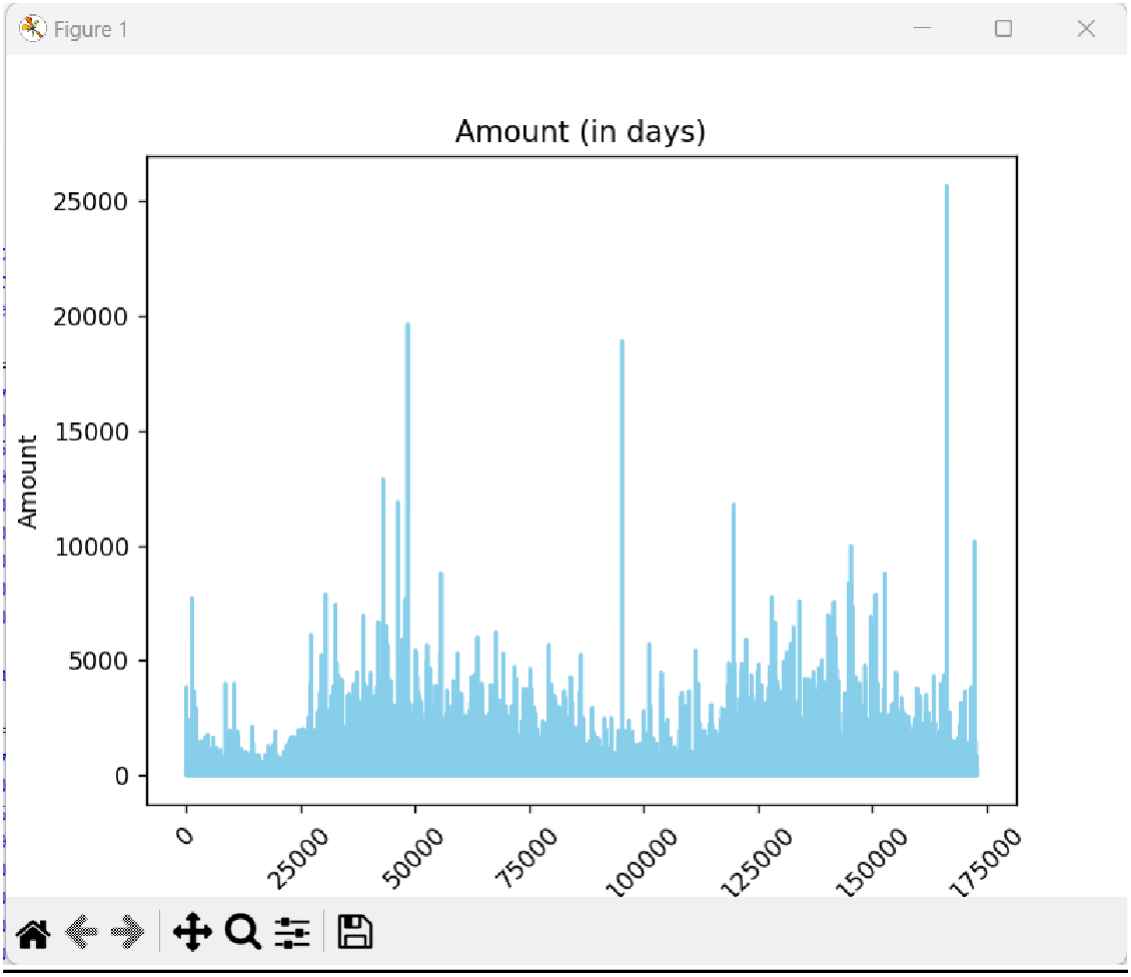
Code:

#Pre-Processing:

```
df=pd.read_csv("S:\\Users\\Sanjay\\PA\\Project\\dataset1.csv")
df = df.dropna()
print(df.describe())
plt.fill_between(df['Time (in days)'], df['Amount'], color='skyblue', alpha=0.4)
plt.plot(df['Time (in days)'], df['Amount'], color='blue')
plt.xlabel('Time (in days)')
plt.ylabel('Amount')
plt.title('Amount (in days)')
plt.xticks(rotation=45)
plt.show()
```

OUTPUT:

```
       Time (in days)             V1  ...         Amount          class
count   284807.000000  284807.000000  ...  284807.000000  284807.000000
mean     94813.859575       0.000008  ...      88.357227       0.001727
std      47488.145955       1.958696  ...     250.119146       0.041527
min          0.000000     -56.410000  ...       0.000000       0.000000
25%      54201.500000      -0.920000  ...       5.600000       0.000000
50%      84692.000000       0.020000  ...      22.000000       0.000000
75%     139320.500000       1.320000  ...      77.200000       0.000000
max     172792.000000       2.450000  ...   25691.200000       1.000000

[8 rows x 31 columns]
```



INFERENCE:

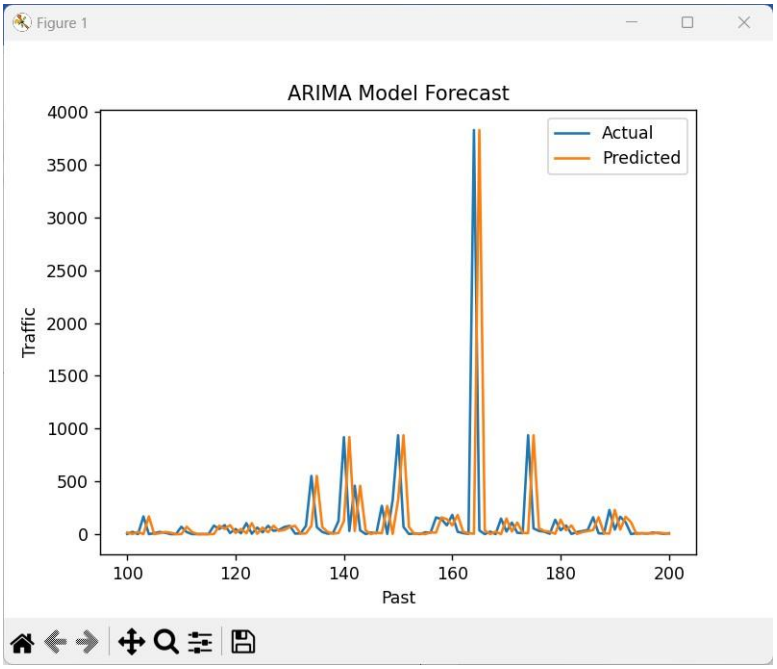
The data has been loaded and pre-processed before starting the analysis. The pre-processing is to prepare the dataset for further analysis. Firstly, the dataset is checked for missing values and if there exists any, it is dropped from the dataset. Then the exploratory analysis is undertaken to oversee the dataset before starting with further analysis techniques. The summary statistics of the dataset is displayed and a plot to represent the amount in days to see how the amount varies from days to days.

#ARIMA:

```
import numpy as np
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error
import matplotlib.pyplot as plt
data = pd.read_csv("dataset1.csv")
endog = data['Amount']
p, d, q = 0, 1, 0
# Create and fit the ARIMA model
arima_model = ARIMA(endog, order=(p, d, q))
arima_model_fit = arima_model.fit()
predictions = arima_model_fit.predict(start=100, end=200)
print(predictions)
test_data = endog[100:201] # Assuming you have a separate test dataset
print(test_data)
mae = np.mean(np.abs(predictions - test_data))
print("Mean Absolute Error (MAE): ", mae)
mape = np.mean(np.abs((test_data - predictions) / test_data))
print("Mean Absolute Percentage Error (MAPE):",mape,"%")
# Plot the actual values
plt.plot(test_data.index, test_data.values, label='Actual')
plt.plot(predictions.index, predictions.values, label='Predicted')
plt.xlabel('Past')
plt.ylabel('Traffic')
plt.title('ARIMA Model Forecast')
plt.legend()
plt.show()
```

OUTPUT:

```
100     16.0
101      2.7
102     22.4
103      0.8
104    168.6
       ...
196      8.8
197      6.0
198     15.9
199     10.0
200      3.9
Name: predicted_mean, Length: 101, dtype: float64
100      2.7
101     22.4
102      0.8
103    168.6
104      0.8
       ...
196      6.0
197     15.9
198     10.0
199      3.9
200     10.0
Name: Amount, Length: 101, dtype: float64
Mean Absolute Error (MAE):  193.50495049504495
Mean Absolute Percentage Error (MAPE): 8.92977825623859 %
```



INFERENCE:

The MAE value of 193.504 indicates, on average, the difference between the actual and predicted values of the amount variable is around 193.504 units. The MAPE value of 8.93% indicates that, on average, the model's predictions have an absolute percentage error of approximately 8.93%. Overall, the ARIMA model seems to perform well in predicting the variable. The relatively low MAE and MAPE values indicate that the model's predictions are relatively close to the actual values, suggesting that it can provide useful insights and forecasts for the credit fraudulent.

 # PCA

```
import seaborn as sns
columns = [ i for i in df.columns if i not in ['Time (in days)' , 'Amount' , 'class']]
print(columns)
X = df[columns]
y=df['class']
# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
pca = PCA()
pca.fit(X_scaled)
X_pca = pca.fit_transform(X_scaled)
explained_variance = pca.explained_variance_ratio_
for i in range(len(columns)):
    print("Column ", columns[i],":-")
    print("Explained Variance Ratio:", explained_variance[i])
    print()
num_components = X_pca.shape[1]
column_names = [f"PC{i+1}" for i in range(num_components)]
X_pca_df = pd.DataFrame(X_pca, columns=column_names)
# correlation heatmap
plt.figure(figsize = (10,20))
sns.heatmap(X_pca_df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap of Principal Components')
plt.show()
```

OUTPUT:

```
[8 rows x 31 columns]
['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V2
8']
Column  V1 :-
Explained Variance Ratio: 0.035720322961395275

Column  V2 :-
Explained Variance Ratio: 0.03571792787779109

Column  V3 :-
Explained Variance Ratio: 0.03571708710658551

Column  V4 :-
Explained Variance Ratio: 0.035716769899551803

Column  V5 :-
Explained Variance Ratio: 0.035716497555535735

Column  V6 :-
Explained Variance Ratio: 0.03571593716853748

Column  V7 :-
Explained Variance Ratio: 0.03571572228796649

Column  V8 :-
Explained Variance Ratio: 0.035715452891401406

Column  V9 :-
Explained Variance Ratio: 0.035715158530269006

Column  V10 :-
Explained Variance Ratio: 0.035714977727174566

Column  V11 :-
Explained Variance Ratio: 0.03571490305587478

Column  V12 :-
Explained Variance Ratio: 0.03571482950271838

Column  V13 :-
Explained Variance Ratio: 0.03571450863873285

Column  V14 :-
Explained Variance Ratio: 0.03571436976542608
```

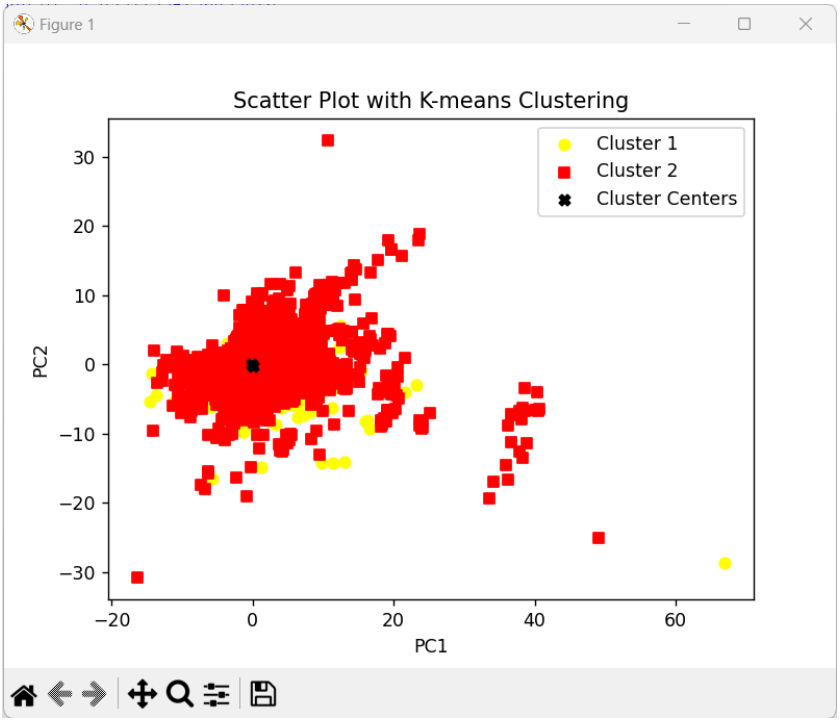
Inference:

The dataset contains 28 columns (V1 to V28) representing helps observe that each principal component explains a relatively small portion of the total variance, with values ranging between approximately 0.0357 and 0.0357. This suggests that the dataset may not exhibit strong patterns or significant variations captured by individual principal components. Which makes all the 28 rows as **PCAs.**

Clustering:

```
from sklearn.cluster import KMeans

k =  2  # Number of clusters

kmeans = KMeans(n_clusters=k, n_init=10)

kmeans.fit(X_pca)

colors = ['yellow', 'red']

markers = ['o', 's']

for i in range(k):

    plt.scatter(X_pca[kmeans.labels_ == i, 0], X_pca[kmeans.labels_ == i, 1], color=colors[i], marker=markers[i], label=f'Cluster {i+1}')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], color='black', marker='X', label='Cluster Centers')

plt.legend()

plt.title('Scatter Plot with K-means Clustering')

# Show the plot

plt.show()
```

OUTPUT:



INFERENCE:

K-means clustering with k=2 clusters is applied to the reduced PCA data. The scatter plot shows the clustered data points, where each cluster is represented by a different colour and marker shape. The cluster centres are also plotted. Each cluster represents a distinct segment of dataset with similar characteristics. By analysing the scatter plot, we can observe how the data points are grouped into clusters and their distribution across the principal components. Clusters that are closer together indicate similar patterns or behaviours among the corresponding credit card reliability.

Discriminant Analysis:

```
classifier = SVC()
classifier.fit(X_train, y_train)
# Use the trained classifier to make predictions on the test data
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("Classification:\nAccuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)
```

OUTPUT:

```
Accuracy: 0.9993153330290369
Precision: 0.9682539682539683
Recall: 0.6224489795918368
F1-Score: 0.7577639751552796
```

INFERENCE:

The Model shows that the SVC model applied to the **Discriminant Analysis** achieved high accuracy and precision rates. However, the recall rate is relatively lower, suggesting that there is room for improvement in capturing a higher proportion of actual fraudulent transactions.

Conclusion:

In credit card fraud detection, multiple analyses were performed to assess different aspects of the data and classification models. The ARIMA model provided useful insights and forecasts for the amount variable, with predictions relatively close to actual values. The PCA analysis revealed that the dataset's principal components explained a small portion of the total variance, suggesting limited patterns or significant variations captured by individual components.The clustering analysis using K-means identified distinct segments of the dataset with similar characteristics, indicating different credit card reliability patterns. Lastly, the discriminant analysis with the SVC model demonstrated high accuracy and precision rates, but the recall rate was relatively lower, indicating room for improvement in capturing a higher proportion of actual fraudulent transactions.

Overall, these analyses collectively contribute to credit card fraud detection. The ARIMA model can help forecast future transaction amounts, the PCA analysis provides insight into the dataset's overall variance structure, the clustering analysis identifies different segments of credit card reliability, and the discriminant analysis with the SVC model contributes to detecting fraudulent transactions accurately. However, to enhance the fraud detection system further, efforts should be made to improve the recall rate, ensuring a higher capture of actual fraudulent transactions.