CREDIT CARD FRAUD DETECTION

TEAM MEMBER

911721104060 : M.MITHUN KUMAR

PHASE 2 SUBMISSION DOCUMENT

Project:credit card fraud detection

Content for project phase 2:

Data source:

Dataset link:(https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud**)**

Data Collection:

1.Source Selection:

Obtain transaction data from credit card companies or financial institutions.

Ensure the data includes features like transaction amount, merchant information, time, and customer details.

2.Data Privacy and Security:

Ensure compliance with data protection regulations (such as GDPR).

Anonymize sensitive information to protect customer privacy.

Data Processing:

1.Data Cleaning:

Handle missing values, duplicate entries, and outliers in the dataset.

Remove or correct erroneous data points to enhance data quality.

2.Data Transformation:

Convert categorical variables into numerical representations using techniques like one-hot encoding.

Normalize numerical features to bring them to a similar scale, enhancing model performance.

3.Data Splitting:

Split the dataset into training and testing sets (e.g., 70% for training, 30% for testing) to evaluate the model effectively.

Feature Engineering:

1.Time-Based Features:

Extract features like hour of the day, day of the week, or whether it's a holiday, as fraudulent patterns might vary with time.

2.Transaction Frequency:

Create features based on the frequency of transactions within specific time intervals for each user.

3.Amount-Based Features:

Derive statistical features such as mean, standard deviation, and maximum transaction amount for each user.

4.Merchant-based Features:

Analyze patterns related to specific merchants, creating features like the average transaction amount per merchant.

Model Selection:

1.Algorithm Selection:

Choose suitable algorithms for fraud detection, such as Random Forest, Gradient Boosting, or Deep Learning models like Neural Networks.

2.Ensemble Methods:

Consider ensemble methods like Random Forest and XGBoost to combine predictions from multiple models, enhancing accuracy.

Model Training:

1.Parameter Tuning:

Use techniques like Grid Search or Random Search to optimize hyperparameters, enhancing the model's performance.

2.Cross-Validation:

Implement cross-validation techniques like k-fold cross-validation to ensure the model's reliability and generalizability.

Evaluation:

1.Metrics Selection:

Use appropriate evaluation metrics such as accuracy, precision, recall, F1-score, and Area Under the ROC Curve (AUC-ROC) to measure the model's effectiveness in fraud detection.

2.Imbalanced Data Handling:

Address class imbalance by using techniques like oversampling, undersampling, or Synthetic Minority Over-sampling Technique (SMOTE) to ensure the model doesn't get biased towards the majority class.

3.Threshold Selection:

Adjust the probability threshold for classification to balance false positives and false negatives, depending on the specific requirements and costs associated with fraud detection errors.

4.Real-time Monitoring:

Implement a system for real-time monitoring of transactions, where the model can be retrained periodically with new data to adapt to evolving fraud patterns.

PROGRAM:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import gridspec
data = pd.read_csv("creditcard.csv")
data.head()
print(data.shape)
print(data.describe())
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]
outlierFraction = len(fraud)/float(len(valid))
print(outlierFraction)
print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))
print("Amount details of the fraudulent transaction")
print(fraud.Amount.describe())
print("details of valid transaction")
print(valid.Amount.describe())
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
X = data.drop(['Class'], axis = 1)
Y = data["Class"]
```

```
print(X.shape)
print(Y.shape)
xData = X.values
yData = Y.values
from sklearn.model_selection import train_test_split
xTrain, xTest, yTrain, yTest = train_test_split(
    xData, yData, test_size = 0.2, random_state = 42)
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
rfc.fit(xTrain, yTrain)
yPred = rfc.predict(xTest)
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix
n_outliers = len(fraud)
n_errors = (yPred != yTest).sum()
print("The model used is Random Forest classifier")
acc = accuracy_score(yTest, yPred)
print("The accuracy is {}".format(acc))
prec = precision_score(yTest, yPred)
print("The precision is {}".format(prec))
rec = recall_score(yTest, yPred)
print("The recall is {}".format(rec))
f1 = f1_score(yTest, yPred)
print("The F1-Score is {}".format(f1))
```

```
MCC = matthews_corrcoef(yTest, yPred)

print("The Matthews correlation coefficient is{}".format(MCC))

LABELS = ['Normal', 'Fraud']

conf_matrix = confusion_matrix(yTest, yPred)

plt.figure(figsize =(12, 12))

sns.heatmap(conf_matrix, xticklabels = LABELS,

        yticklabels = LABELS, annot = True, fmt ="d");

plt.title("Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()
```

output:

```
(284807, 31)

            Time          V1          V2          V3          V4  ...         V26         V27         V28
Amount         Class

count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  ...
2.848070e+05  2.848070e+05  2.848070e+05  284807.000000  284807.000000

mean    94813.859575  1.168375e-15  3.416908e-16 -1.379537e-15  2.074095e-15  ...
1.683437e-15 -3.660091e-16 -1.227390e-16      88.349619       0.001727

std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00  ...
4.822270e-01  4.036325e-01  3.300833e-01     250.120109       0.041527

min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00  ... -
2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000       0.000000

25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01  ... -
3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000       0.000000

50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02  ... -
5.213911e-02  1.342146e-03  1.124383e-02      22.000000       0.000000

75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01  ...
2.409522e-01  9.104512e-02  7.827995e-02      77.165000       0.000000
```

max 172792.000000 2.454930e+00 2.205773e+01 9.382558e+00 1.687534e+01 ... 3.517346e+00 3.161220e+01 3.384781e+01 25691.160000 1.000000

[8 rows x 31 columns]

0.0017304750013189597

Fraud Cases: 492

Valid Transactions: 284315

Amount details of the fraudulent transaction

count 492.000000

mean 122.211321

std 256.683288

min 0.000000

25% 1.000000

50% 9.250000

75% 105.890000

max 2125.870000

Name: Amount, dtype: float64

details of valid transaction

count 284315.000000

mean 88.291022

std 250.105092

min 0.000000

25% 5.650000

50% 22.000000

75% 77.050000

max 25691.160000

Name: Amount, dtype: float64

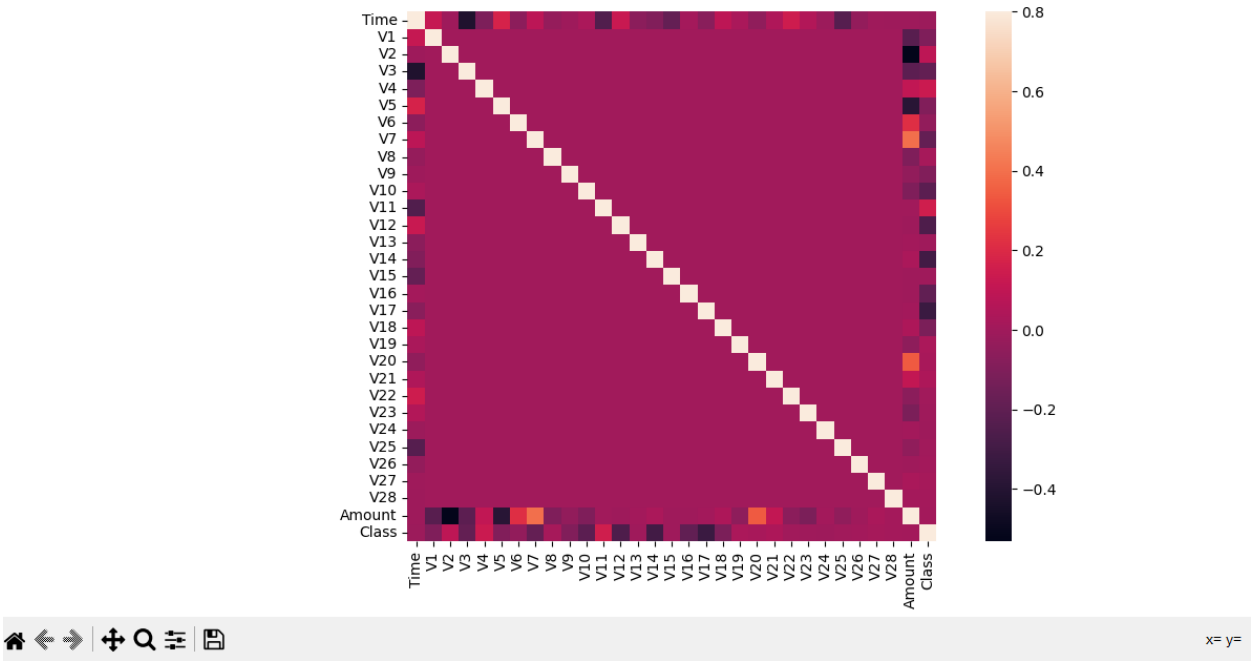
Figure 1
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
x= y=

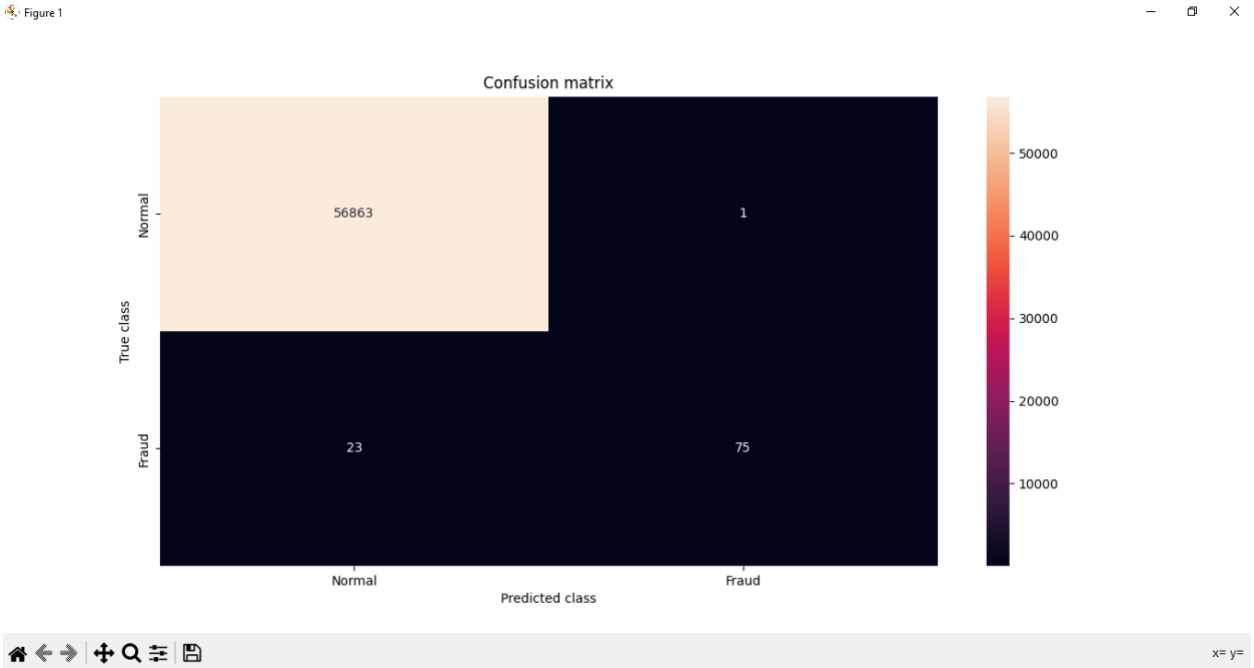
Figure 1
Confusion matrix
Normal
Fraud
True class
56863
1
23
75
Normal
Fraud
Predicted class
50000
40000
30000
20000
10000
x= y=

Conclusion:

In conclusion, credit card fraud detection using machine learning is a multidimensional process. It starts with meticulous data collection, followed by preprocessing to ensure data quality and integrity. Feature engineering plays a vital role in capturing subtle fraud patterns, while model selection and training determine the system's efficacy.

Regular evaluation using appropriate metrics is essential to measure the model's performance accurately. A dynamic approach, involving real-time monitoring and periodic retraining, is crucial to staying ahead of emerging fraud tactics. By integrating these steps cohesively, organizations can build robust fraud detection systems, safeguarding financial transactions and ensuring a secure environment for both businesses and consumers.