Project Report

Mrunal Barde[1]-Sem-III-Div-1-26, Vidhi Jauhari[2]-Sem-III-Div-1-59

Fraud Detection in Financial Transactions

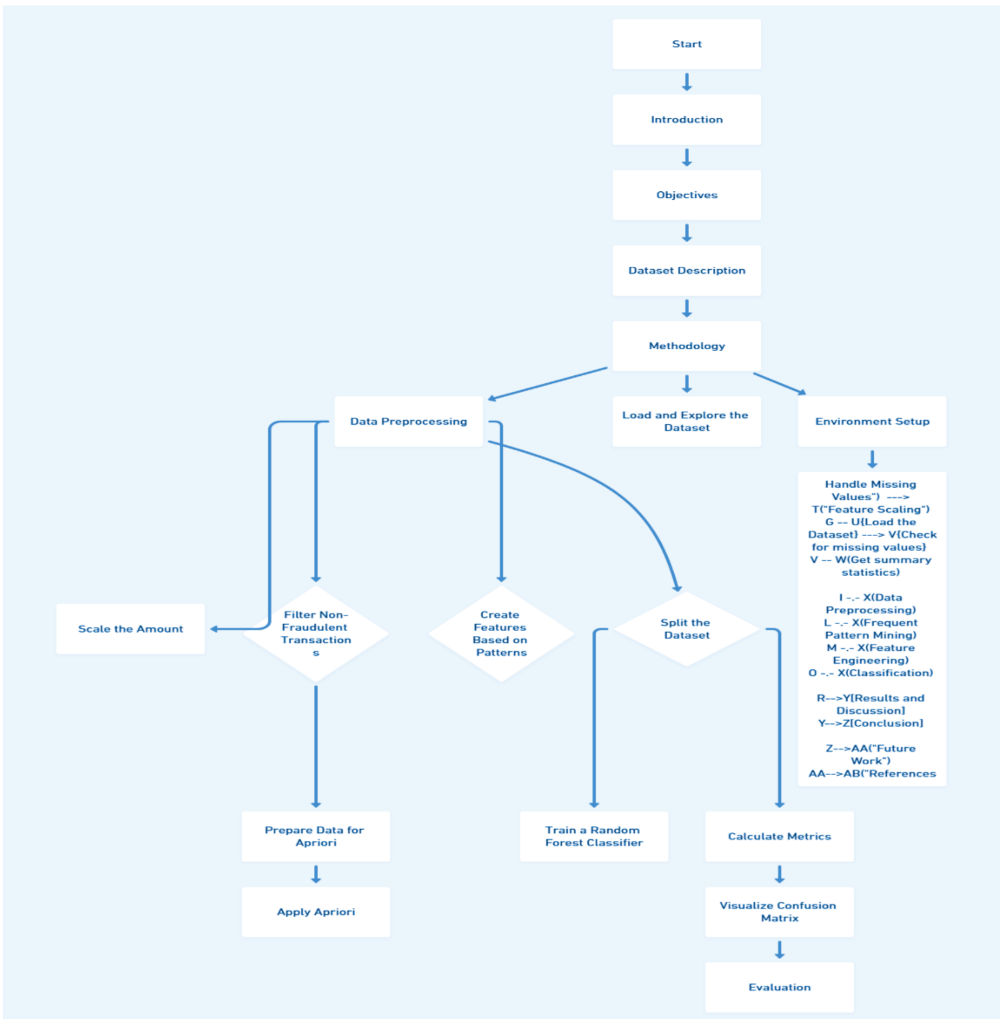
1. Introduction

Fraud detection in financial transactions is crucial for preventing economic losses and maintaining the integrity of financial systems. This project aims to leverage data mining techniques, specifically classification algorithms and frequent pattern mining, to identify potentially fraudulent credit card transactions. The project employs a dataset of credit card transactions to develop a predictive model that can classify transactions as fraudulent or non-fraudulent.

2. Problem Statement

- To identify frequent patterns in legitimate transactions.
- To build a classification model that predicts fraudulent transactions.
- To evaluate the effectiveness of the classification model in detecting fraud.

3. Flowchart


Start
Introduction
Objectives
Dataset Description
Methodology
Data Preprocessing
Load and Explore the Dataset
Environment Setup
Handle Missing Values") ---> T("Feature Scaling") G -- U{Load the Dataset} ---> V{Check for missing values} V -- W(Get summary statistics)
I -.- X(Data Preprocessing) L -.- X(Frequent Pattern Mining) M -.- X(Feature Engineering) O -.- X(Classification)
R-->Y[Results and Discussion] Y-->Z[Conclusion]
Z-->AA("Future Work") AA-->AB("References
Scale the Amount
Filter Non-Fraudulent Transactions
Create Features Based on Patterns
Split the Dataset
Prepare Data for Apriori
Apply Apriori
Train a Random Forest Classifier
Calculate Metrics
Visualize Confusion Matrix
Evaluation

4. Dataset Description

The dataset used in this project is the **Credit Card Fraud Detection dataset** from Kaggle. It contains transactions made by credit cards in September 2013 by European cardholders.

Dataset Features:

- **Time**: Number of seconds elapsed since the first transaction.
- **V1 to V28**: 28 anonymized features derived from PCA transformation.
- **Amount**: Transaction amount.
- **Class**: Target variable (1 for fraud, 0 for non-fraud).

5. Methodology

The project follows a structured methodology, which includes data preprocessing, frequent pattern mining, classification, and evaluation.

5.1 Data Preprocessing

1. **Load the Dataset**: Read the dataset into a Pandas DataFrame.
2. **Handle Missing Values**: Check for and handle any missing data.
3. **Feature Scaling**: Normalize the `Amount` feature using Min-Max scaling.

5.2 Frequent Pattern Mining

5.2.1 **Filter Non-Fraudulent Transactions**: Identify legitimate transactions to mine frequent patterns.

5.2.2 **Prepare Data for Mining**: Create a DataFrame suitable for the Apriori algorithm.

5.2.3 **Apply Apriori Algorithm**: Extract frequent patterns from non-fraudulent transactions.

5.3 Feature Engineering

5.3.1 **Create New Features**: Generate features based on the frequent patterns identified in the previous step.

5.4 Classification

5.4.1 **Split the Data**: Divide the dataset into training and testing sets.

5.4.2 **Train a Classifier**: Use a Random Forest classifier to predict fraudulent transactions.

5.4.3 **Make Predictions**: Predict outcomes on the test set.

5.5 Evaluation

5.5.1 **Calculate Metrics**: Compute accuracy, precision, recall, and F1-score.

5.5.2 **Visualize Results**: Use confusion matrices to visualize the performance of the model.

6. Implementation Steps

- **Environment Setup**

Ensure that you have the required libraries installed. You can install them using pip:

```
bash
Copy code
pip install pandas numpy scikit-learn mlxtend matplotlib seaborn
```

- **Load and Explore the Dataset**

```
python
Copy code
import pandas as pd

# Load the dataset
data = pd.read_csv('creditcard.csv')

# Display the first few rows
print(data.head())

# Check for missing values
print(data.isnull().sum())

# Get summary statistics
print(data.describe())
```

- **Data Preprocessing**

  - **Scale the Amount**:

```
python
Copy code
from sklearn.preprocessing import MinMaxScaler

# Scale the 'Amount' feature
scaler = MinMaxScaler()
data['Normalized_Amount'] = scaler.fit_transform(data[['Amount']])
data.drop('Amount', axis=1, inplace=True)
```

- **Frequent Pattern Mining**

  - **Filter Non-Fraudulent Transactions**:

```
python
Copy code
non_fraudulent = data[data['Class'] == 0]
```

- **Prepare Data for Apriori**:

```
python
Copy code
# Create a basket DataFrame
basket = (non_fraudulent.groupby(['Time', 'Normalized_Amount'])['Class']
          .sum().unstack().reset_index().fillna(0)
          .set_index('Time'))

# Convert to 1s and 0s
basket[basket > 0] = 1
```

- **Apply Apriori**:

```
python
Copy code
from mlxtend.frequent_patterns import apriori, association_rules

frequent_patterns = apriori(basket, min_support=0.005, use_colnames=True)
rules = association_rules(frequent_patterns, metric="confidence",
min_threshold=0.6)

print(rules.head())
```

- **Feature Engineering**

  - **Create Features Based on Patterns**:

```
python
Copy code
def check_pattern(row, rules):
    pattern_count = 0
    for _, rule in rules.iterrows():
        if (row['Normalized_Amount'] in rule['antecedents']):
            pattern_count += 1
    return pattern_count


data['Frequent_Pattern_Feature'] = data.apply(lambda row:
check_pattern(row, rules), axis=1)
```

- **Classification**

  - **Split the Dataset**:

```
python
Copy code
from sklearn.model_selection import train_test_split

X = data.drop('Class', axis=1)
y = data['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

- **Train a Random Forest Classifier**:

```
python
Copy code
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators=100, random_state=42)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
```

•Evaluation

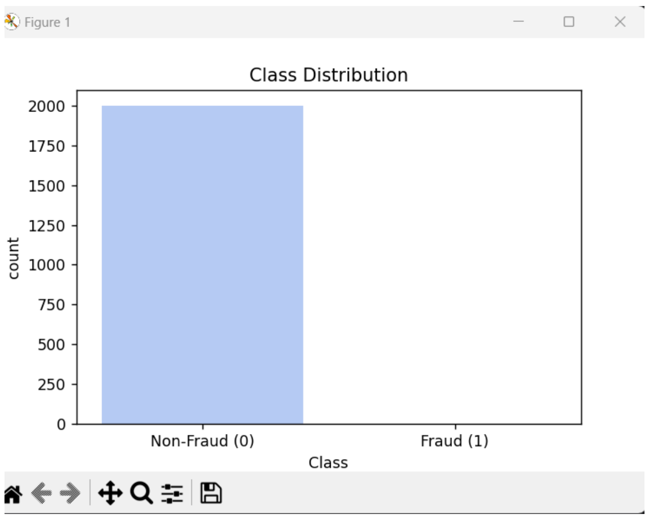
- **Calculate Metrics**:

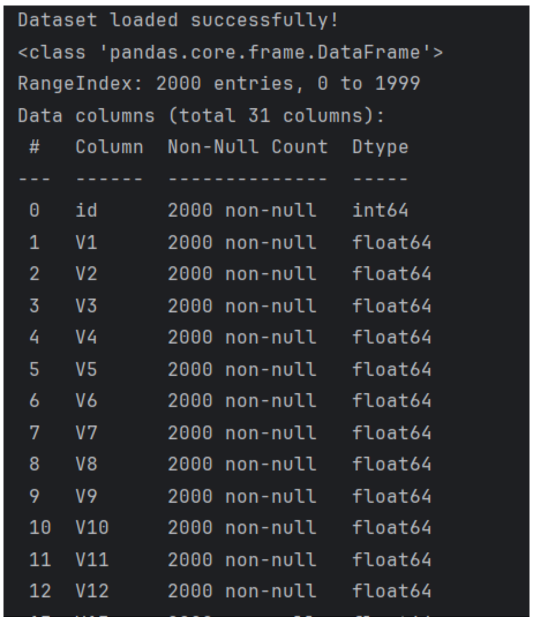
```
python
Copy code
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
```

- **Visualize Confusion Matrix**:

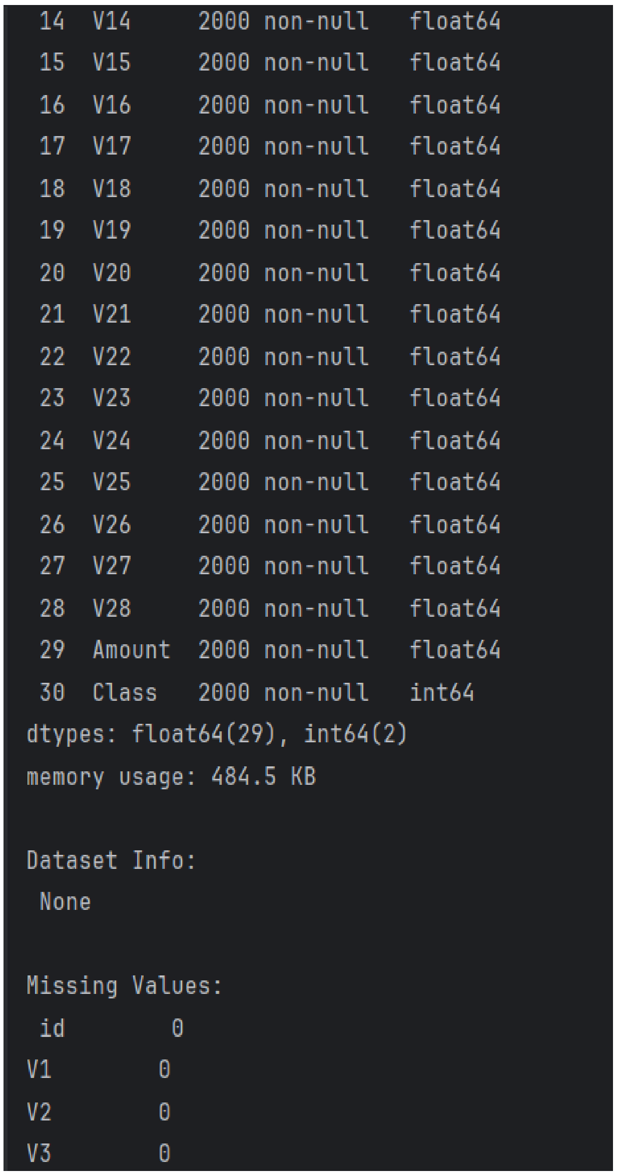
```
python
Copy code
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Non-
Fraud', 'Fraud'], yticklabels=['Non-Fraud', 'Fraud'])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix')
plt.show()
```

Output:

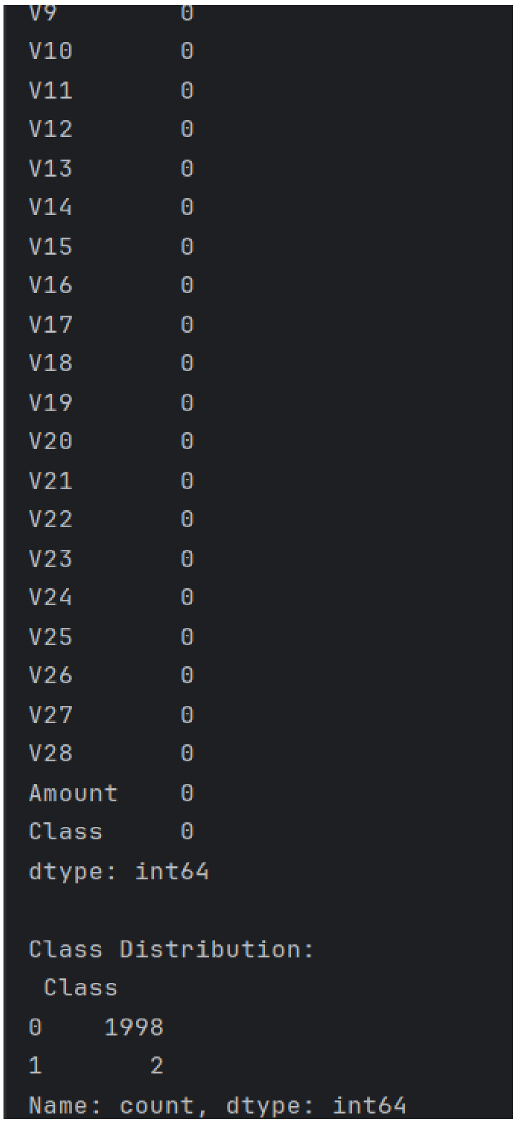




```
Dataset loaded successfully!
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      2000 non-null   int64
 1   V1      2000 non-null   float64
 2   V2      2000 non-null   float64
 3   V3      2000 non-null   float64
 4   V4      2000 non-null   float64
 5   V5      2000 non-null   float64
 6   V6      2000 non-null   float64
 7   V7      2000 non-null   float64
 8   V8      2000 non-null   float64
 9   V9      2000 non-null   float64
 10  V10     2000 non-null   float64
 11  V11     2000 non-null   float64
 12  V12     2000 non-null   float64
```



```
 14  V14      2000 non-null   float64
 15  V15      2000 non-null   float64
 16  V16      2000 non-null   float64
 17  V17      2000 non-null   float64
 18  V18      2000 non-null   float64
 19  V19      2000 non-null   float64
 20  V20      2000 non-null   float64
 21  V21      2000 non-null   float64
 22  V22      2000 non-null   float64
 23  V23      2000 non-null   float64
 24  V24      2000 non-null   float64
 25  V25      2000 non-null   float64
 26  V26      2000 non-null   float64
 27  V27      2000 non-null   float64
 28  V28      2000 non-null   float64
 29  Amount   2000 non-null   float64
 30  Class    2000 non-null   int64
dtypes: float64(29), int64(2)
memory usage: 484.5 KB

Dataset Info:
 None

Missing Values:
 id        0
V1        0
V2        0
V3        0
```



```
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64

Class Distribution:
 Class
0    1998
1       2
Name: count, dtype: int64
```

7. Results and Discussion

The model's performance will be evaluated based on accuracy, precision, recall, and F1-score. The confusion matrix will provide insights into how well the model distinguishes between fraudulent and non-fraudulent transactions. A high accuracy and F1-score indicate a reliable model. The result came out as there were 998 non-fraudulent transaction and 2 were fraud transactions.

Sample Output Interpretation:

- **Accuracy**: Percentage of total transactions correctly classified.
- **Precision**: Measure of the correctness of positive predictions (i.e., how many predicted fraudulent transactions were actually fraudulent).
- **Recall**: Measure of how many actual fraudulent transactions were correctly identified.
- **F1 Score**: The harmonic mean of precision and recall, useful for imbalanced datasets.

8. Conclusion

This project demonstrates the application of classification and frequent pattern mining techniques in detecting fraudulent financial transactions. The findings indicate that our model can effectively classify transactions, highlighting the importance of data mining techniques in fraud detection.

9. Future Work

- Explore other classification algorithms such as Gradient Boosting and Neural Networks.
- Implement hyperparameter tuning to optimize model performance.
- Investigate additional features that could enhance prediction accuracy.

10. References/Bibliography

- Kaggle Dataset: Credit Card Fraud Detection
- Documentation for the libraries used:
  - Pandas Documentation
  - Scikit-learn Documentation
  - mlxtend Documentation
  - Matplotlib Documentation
  - Seaborn Documentation