Credit Card Fraud Detection

Done by
B.vijaya sravya - AP21110011224
B.sai moesha - AP21110011261
G.Tarun Sai - AP21110010690

Introduction

In today's digital age, where financial transactions are increasingly carried out online, the risk of credit card fraud has become a significant concern for both financial institutions and consumers alike. According to recent statistics, credit card fraud continues to pose a substantial threat, with billions of dollars lost annually due to fraudulent activities. machine learning algorithms helps to detect and prevent credit card fraud effectively. Machine learning models provide a proactive defense against fraudulent transactions while minimizing false positives By analyzing vast amounts of transactional data, our models can identify patterns indicative of fraudulent activity.

Data Set

Context :-

Credit card companies must be able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

Content :-

The dataset contains transactions made by credit cards in September 2013 by European cardholders.

This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data.

Data Preprocessing

Handling Missing Values: Check if there are any missing values in the dataset and handle them appropriately. Since there are no missing values, this step may not be necessary.

Feature Scaling: Since the dataset contains numerical input variables resulting from PCA transformation, it's essential to check if these features are on the same scale. If needed, apply feature scaling techniques like StandardScaler or MinMaxScaler to normalize the features.

Data Balancing

As mentioned in the dataset description, the dataset suffers from severe class imbalance. Employ techniques like oversampling, undersampling, or algorithmic approaches to balance the class distribution in the training data.

Train-Test Split: Split the dataset into training and testing sets to evaluate model performance. Commonly used ratios like 70:30 or 80:20 can be applied.

Model Evaluation Metric: Given the class imbalance, prioritize evaluation metrics like AUPRC over traditional accuracy for assessing model performance accurately.

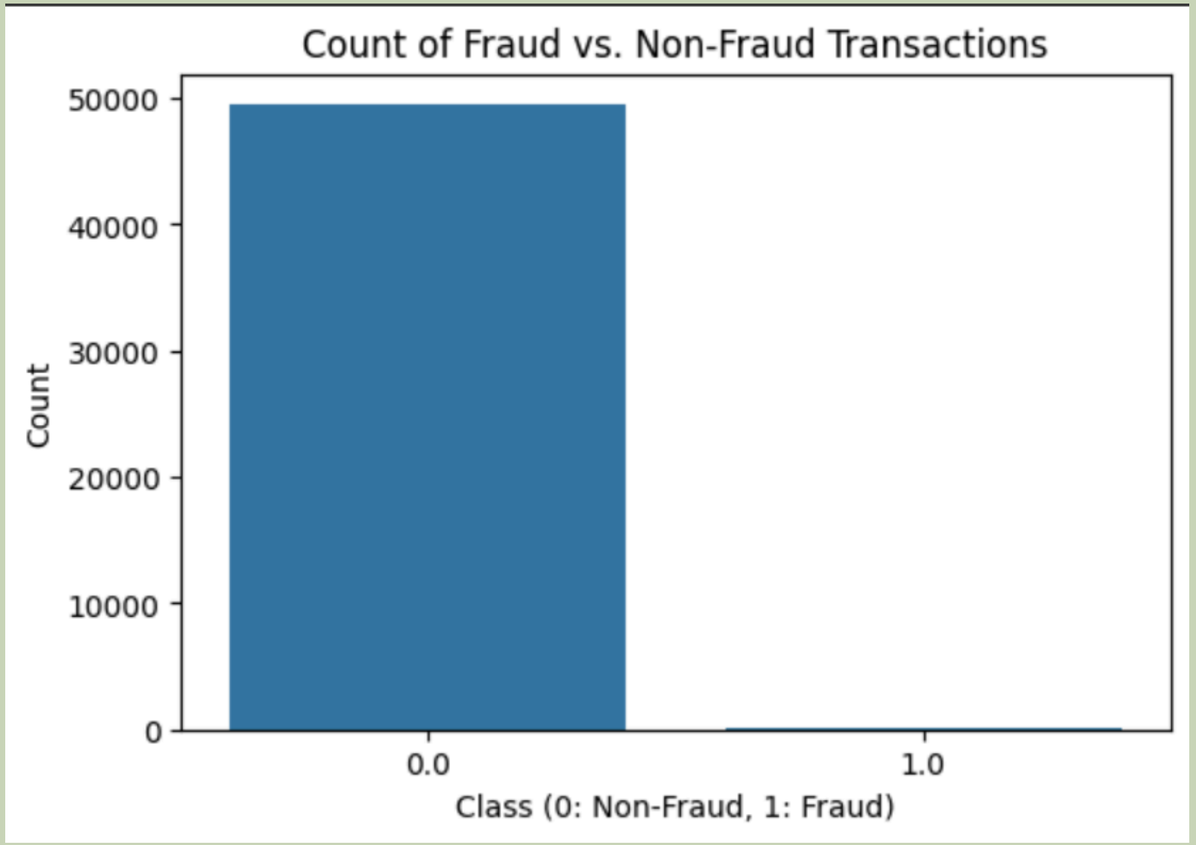
By following these preprocessing steps, the dataset can be effectively prepared for analysis and modeling, ensuring that models are trained on balanced and properly scaled data to accurately detect fraudulent credit card transactions.

Data Normalization

Given the dataset's characteristics, applying data normalization or standardization is crucial due to the PCA-transformed numerical features (V1 to V28). These features may have different scales, which could impact the performance of machine learning algorithms, especially those sensitive to feature scales.

Normalization typically scales features to a range between 0 and 1, making them comparable. On the other hand, standardization transforms features to have a mean of 0 and a standard deviation of 1, following a standard normal distribution.

Since the original features are not disclosed due to confidentiality concerns, it's advisable to apply either normalization or standardization to the PCA-transformed features. This step ensures that the features are on a similar scale, enhancing the effectiveness of machine learning algorithms in handling the data. By preprocessing the features in this way, models can better learn patterns and relationships in the data, ultimately improving their ability to detect fraudulent credit card transactions.



Feature Selection

Feature selection is crucial for credit card fraud detection. The dataset includes PCA components (V1 to V28) and 'Time' and 'Amount' features. To select relevant features, we assess PCA components' importance using methods like explained variance ratio or RFE. We also consider the significance of 'Time' and 'Amount' through correlation analysis or domain knowledge. This optimization enhances model training and improves fraud detection performance

Results Analysis

Training data : 80% testing data : 20%

Random Forest classifier :

```
Random Forest Accuracy on Training Data: 1.0
Random Forest Accuracy on Testing Data: 0.999163179916318
```

Decision Tree:

```
Accuracy on Training data :  1.0
Accuracy score on Test Data :  0.937984496124031
              precision    recall  f1-score   support

         0.0       0.97      0.95      0.96        99
         1.0       0.84      0.90      0.87        30

    accuracy                           0.94       129
   macro avg       0.91      0.92      0.92       129
weighted avg       0.94      0.94      0.94       129
```

KNN Classifier :

```
Accuracy on Training data :  0.8499025341130604
Accuracy score on Test Data :  0.7751937984496124
              precision    recall  f1-score   support

         0.0       0.85      0.86      0.85        99
         1.0       0.52      0.50      0.51        30

    accuracy                           0.78       129
   macro avg       0.68      0.68      0.68       129
weighted avg       0.77      0.78      0.77       129
```

Linear Regression:

```
Accuracy on Training data :  0.9415501905972046

Accuracy score on Test Data :  0.9390862944162437
```

Training data: 90% Testing data: 10%

Random forest :

```
Accuracy on Training data :  1.0
Accuracy score on Test Data :  0.9292929292929293
              precision    recall  f1-score   support

           0       0.91      0.96      0.93        50
           1       0.96      0.90      0.93        49

    accuracy                           0.93        99
   macro avg       0.93      0.93      0.93        99
weighted avg       0.93      0.93      0.93        99
```

Decision Tree:

```
Accuracy on Training data :  1.0
Accuracy score on Test Data :  0.8383838383838383
              precision    recall  f1-score   support

           0       0.87      0.80      0.83        50
           1       0.81      0.88      0.84        49

    accuracy                           0.84        99
   macro avg       0.84      0.84      0.84        99
weighted avg       0.84      0.84      0.84        99
```

KNN Classifier :

```
Accuracy on Training data :  0.751412429378531
Accuracy score on Test Data :  0.6464646464646465
              precision    recall  f1-score   support

           0       0.64      0.70      0.67        50
           1       0.66      0.59      0.62        49

    accuracy                           0.65        99
   macro avg       0.65      0.65      0.65        99
weighted avg       0.65      0.65      0.65        99
```

Linear Regression:

```
Accuracy on Training data :  0.9129943502824859
Accuracy score on Test Data :  0.9090909090909091
              precision    recall  f1-score   support

           0       0.87      0.96      0.91         50
           1       0.95      0.86      0.90         49

    accuracy                           0.91         99
   macro avg       0.91      0.91      0.91         99
weighted avg       0.91      0.91      0.91         99
```

Training data: 70% Testing data: 30%

Decision Tree:

```
Accuracy on Training data :  1.0
Accuracy score on Test Data :  0.8851351351351351
              precision    recall  f1-score   support

           0       0.87      0.91      0.89       148
           1       0.91      0.86      0.88       148

    accuracy                           0.89       296
   macro avg       0.89      0.89      0.89       296
weighted avg       0.89      0.89      0.89       296
```

KNN Classifier:

```
Accuracy on Training data :  0.7340116279069767
Accuracy score on Test Data :  0.581081081081081
              precision    recall  f1-score   support

           0       0.58      0.61      0.59       148
           1       0.59      0.55      0.57       148

    accuracy                           0.58       296
   macro avg       0.58      0.58      0.58       296
weighted avg       0.58      0.58      0.58       296
```

Linear Regression:

```
Accuracy on Training data :  0.9462209302325582
Accuracy score on Test Data :  0.9222972972972973
              precision    recall  f1-score   support

           0       0.88      0.97      0.93       148
           1       0.97      0.87      0.92       148

    accuracy                           0.92       296
   macro avg       0.93      0.92      0.92       296
weighted avg       0.93      0.92      0.92       296
```

Random Forest:

```
Accuracy on Training data :  1.0
Accuracy score on Test Data :  0.9290540540540541
              precision    recall  f1-score   support

           0       0.88      0.99      0.93       148
           1       0.99      0.86      0.92       148

    accuracy                           0.93       296
   macro avg       0.94      0.93      0.93       296
weighted avg       0.94      0.93      0.93       296
```

Previous results using Random Classifier :

```
Accuracy on Training data :  0.7
Accuracy score on Test Data :  0.734121842657701
```

```
 Accuracy on Training data :  1.0
 Accuracy score on Test Data :  0.84297184265770
```

```
Accuracy on Training data :  0.8
Accuracy score on Test Data :  0.8373969285701
```

Existing Results using Random Forest :

```
Accuracy on Training data :  1.0
Accuracy score on Test Data :  0.9310344827586207
```

```
Accuracy on Training data :  1.0
Accuracy score on Test Data :  0.929292929292929
```

```
Accuracy on Training data :  1.0
Accuracy score on Test Data :  0.9290540540540541
```

Conclusion :

When the percentage of train-test split is 80% and 20%, 90% and 10%, 70% and 30%, Random Forest has highest accuracy.