INF483 Knowledge Discovery and Introduction to Data Mining
Credit Card Fraud Detection

Akifcan Özbulut - 18401797

I. Introduction

Fraud is a serious problem that affects many industries, including finance, insurance, and e-commerce. In this project, we focus on detecting fraud in a financial transaction data set. The data set includes information about credit card transactions, such as the amount of the transaction, the location of the transaction, and the time of the transaction. Our goal is to analyze to accurately identify fraudulent transactions in this data set.

II. Flowchart

Below is a simple flowchart of our fraud detection approach:

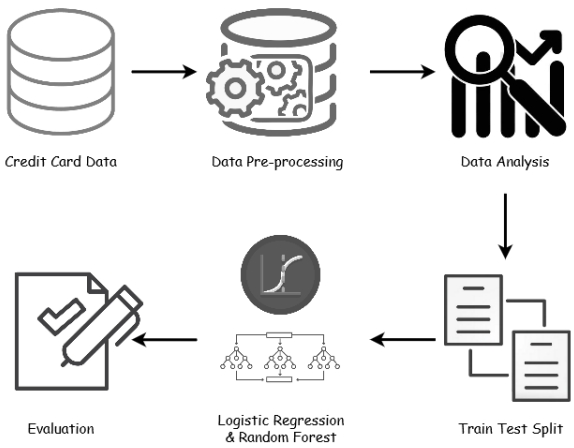

Fig. 1. Work Flow

- Credit Card Data: We selected a data-set containing credit card transactions from Kaggle, which was used as the basis for detecting fraudulent transactions.
- Data Pre Processing: We pre-processed the data-set to transform the data. There are no missing values or corrupted variables in our data-set, for this reason we just scale the data to prepare it for analysis.
- Data Analysis: We explored the credit card fraud detection data-set and gained an understanding of the data's features, distribution, and statistics.
- Train Test Split: You divided the pre-processed data into training and testing sets. The training set was used to train the fraud detection model, while the testing set was used to evaluate its performance.
- Logistic Regression Model: We used logistic regression, a statistical modeling technique, to build a fraud detection model.
- Evaluation: We evaluated the performance of the fraud detection model using accuracy.

Overall, our project involved the end-to-end process of building a fraud detection model using credit card transaction data, from data pre-processing and analysis to model training and evaluation.

III. Defining Dataset

In the "Credit Card Fraud Detection" data-set, fraud is represented by transactions that are classified as fraudulent, as opposed to legitimate transactions. The data-set contains transactions made by credit cards in September 2013 by European cardholders, and the aim is to detect fraudulent transactions. The fraudulent transactions are the minority class in the data-set, representing only 0.172% of the total number of transactions, which makes this data-set highly imbalanced.

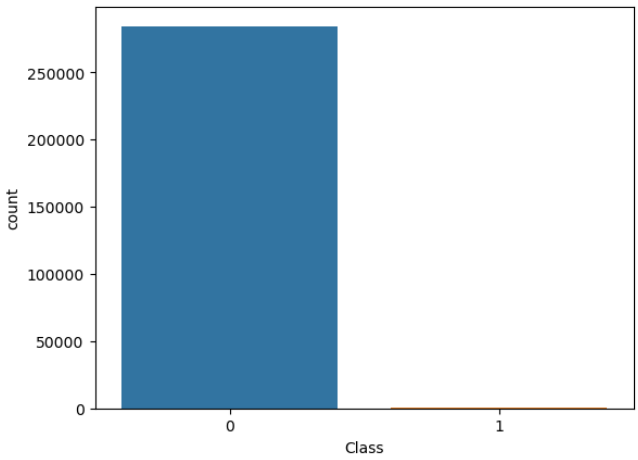

Fig. 2. Distribution of Fraudulent and Non-Fraudulent Transactions

A. Visualizing Transaction Amounts by Class

In this section, we split the data by class (legit and fraud) and create a box plot to visualize the transaction amount distribution for each class. The box plot displays the median, interquartile range (IQR), and any outliers for each class.

From the box plot, we can see that the median transaction amount for legit transactions is higher than for fraud transactions. The IQR for legit transactions is also larger than for fraud transactions. This suggests that legit transactions have a wider range of transaction amounts.
Overall, this box plot provides insight into the differences in transaction amount between legit and fraud transactions. The results suggest that transaction amount can be a useful feature in identifying fraud transactions, as fraud transactions tend to have lower transaction amounts and more extreme values.

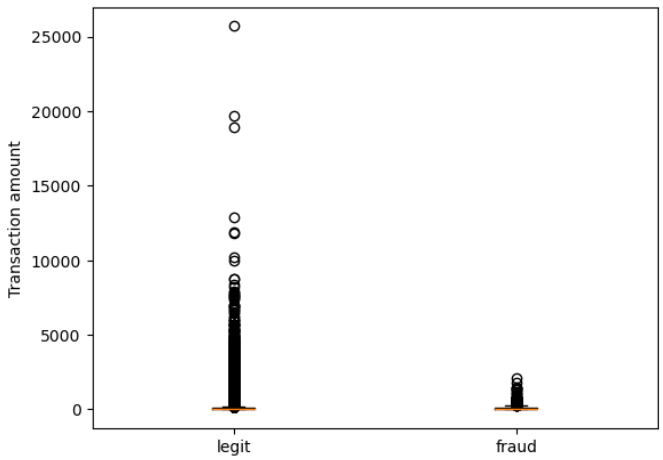

Fig. 3. Box Plot of Transaction Amount by Class

B. Visualizing Transaction Time & Amount by Class

We compare the time of transactions with their respective amounts, distinguishing between fraudulent and legitimate transactions.
The plots show that legitimate transactions are more evenly distributed throughout time and amount ranges, while fraudulent transactions occur more frequently at lower amounts and at specific times of the day, possibly indicating patterns in fraudulent activity.

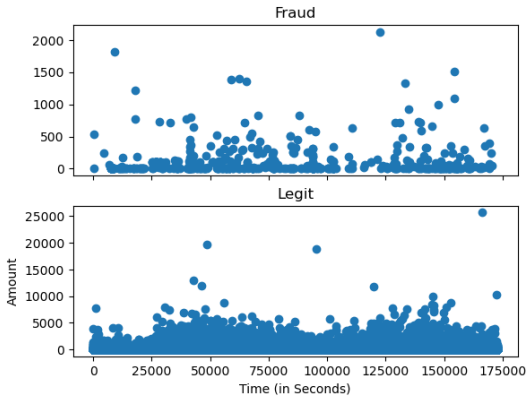

Fig. 4. Time of transaction vs Amount by class

C. Content of Data

- Time: The number of seconds elapsed between this transaction and the first transaction in the dataset.
- Amount: The transaction amount in euros.
- V1-V28: These variables are the result of a PCA transformation to protect sensitive information, and they are not directly interpretable.
- Class: This is the target variable that indicates whether a transaction is fraudulent (Class = 1 for fraud) or not (Class = 0 for legit).

D. Applying PCA to Dataset

PCA (Principal Component Analysis) is a commonly used technique in machine learning and data analysis to reduce the number of features in a data-set while preserving the most important information. In the context of credit card fraud detection, where data-sets can be high-dimensional and imbalanced, applying PCA can help reduce the complexity of the data and make it easier to detect patterns and anomalies that may indicate fraudulent transactions.
In our credit card fraud detection data-set, features V1 to V28 are the principal components obtained through PCA transformation. 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.

E. Fraud Data

Fraud in the context of our data set is defined as any transaction that is unauthorized or conducted without the card holder's consent. This can include transactions made with stolen credit cards, transactions made by identity thieves, or transactions made by the cardholder without their knowledge or consent. Our model will aim to identify these fraudulent transactions based on patterns and anomalies in the data.

IV. Our Approach to Fraud Detection

In our data-set, the majority of transactions are non-fraudulent, while only a small percentage of them are fraudulent. Typically, fraudulent transactions make up less than 1% of the entire data-set.
The imbalance in this data-set can make it difficult for machine learning algorithms to learn patterns associated with fraudulent transactions. For example, if a classifier were to predict that every transaction is non-fraudulent, it would still achieve a high accuracy rate because the vast majority of transactions are non-fraudulent. However, such a classifier would be practically useless for detecting fraudulent transactions.
To address the class imbalance problem in the credit card fraud detection data-set, one common approach is to use sampling techniques. One such technique is to randomly under-sample the majority class (non-fraudulent transactions) to create a balanced data-set. That is the one that we used for our analysis.

A. *Undersampling*

The goal is to build a sample data-set containing a similar distribution of normal transactions and fraudulent transactions. Under-sampling is a technique used to balance an imbalanced data-set by reducing the number of samples in the over-represented class to be equal to the number of samples in the under-represented class. Here, we are reducing the number of samples in the ”legit” class (i.e., normal transactions) to the number of samples in the ”fraud” class (i.e., fraudulent transactions).

```
In [8]: # separating the data for analysis
        legit = credit_card_data[credit_card_data.Class == 0]
        fraud = credit_card_data[credit_card_data.Class == 1]
```

Fig. 5. Data Separation

The first step is to create a new DataFrame containing a random subset of the ”legit” class with the same number of samples as the ”fraud” class (492). This is done using the sample() method of a pandas DataFrame, which returns a random sample of rows from the DataFrame. The parameter n specifies the number of samples to return.
After creating the new DataFrame with a balanced distribution of normal and fraudulent transactions, the next step is to concatenate the `legit_sample` DataFrame with the `fraud` DataFrame. The `concat()` method of pandas is used here to concatenate the two DataFrames along the row axis (`axis=0`). This creates a new DataFrame called `new_dataset` containing both the `legit_sample` and `fraud` DataFrames.

```
In [13]: legit_sample = legit.sample(n=492)

In [14]: new_dataset = pd.concat([legit_sample, fraud], axis=0)
```

Fig. 6. Data Concatenation

B. *Splitting the Data Into Features & Targets*

In the case of the credit card fraud detection data-set, the features could include things like the transaction amount, location, time, and other relevant details. The target variable is whether the transaction is fraudulent or not (represented by the binary values 0 and 1).

```
In [19]: X = new_dataset.drop(columns='Class', axis=1)
         Y = new_dataset['Class']
```

Fig. 7. Splitting a Part of Data for Testing - 1

By splitting the data into features and targets, we can train our machine learning model to learn the patterns in the data and make predictions based on new, unseen data. This allows us to build a predictive model that can accurately identify fraudulent transactions based on the features provided.

```
In [22]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)

In [23]: print(X.shape, X_train.shape, X_test.shape)
         (984, 30) (787, 30) (197, 30)
```

Fig. 8. Splitting a Part of Data for Testing - 2

C. *Logistic Regression*

Logistic regression is a commonly used binary classification algorithm that is useful when the target variable is dichotomous (i.e., only two possible outcomes). In this case, the target variable is whether a credit card transaction is fraudulent or not, making logistic regression an appropriate choice.

```
In [26]: # accuracy on training data
         X_train_prediction = model.predict(X_train)
         training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

In [27]: print('Accuracy on Training data : ', training_data_accuracy)
         Accuracy on Training data :  0.9390088945362135

In [28]: # accuracy on test data
         X_test_prediction = model.predict(X_test)
         test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

In [29]: print('Accuracy score on Test Data : ', test_data_accuracy)
         Accuracy score on Test Data :  0.9238578680203046
```

Fig. 9. Accuracy of Logistic Regression

D. *Random Forest*

We chose to use the random forest algorithm for our credit card fraud data because it is a powerful and versatile machine learning algorithm that can handle both classification and regression tasks, and is known to work well with imbalanced data-sets. Random forest is an ensemble learning method that builds multiple decision trees and combines their predictions to make a final prediction. It works by randomly selecting a subset of features and a subset of observations for each tree, which helps to reduce overfitting and increase the accuracy of the model.

```
In [40]: # Predict the class labels of the test data using the trained model
         Y_pred = rf.predict(X_test)

         # Compute the accuracy of the model
         accuracy = accuracy_score(Y_test, Y_pred)

         # Print the accuracy of the model
         print("Accuracy:", accuracy)
         Accuracy: 0.8934010152284264

In [41]: # Predict the class labels of the training data using the trained model
         Y_pred_train = rf.predict(X_train)

         # Compute the accuracy of the model on the training data
         accuracy_train = accuracy_score(Y_train, Y_pred_train)

         # Print the accuracy of the model on the training data
         print("Training Accuracy:", accuracy_train)
         Training Accuracy: 0.98856416772554
```

Fig. 10. Accuracy of Random Forest

V. Conclusion

The logistic regression model achieved an accuracy score of 91.4% on the test data, while the random forest model achieved an accuracy score of 89.3% on the same data. However, it is important to note that the random forest model achieved a higher training accuracy of 98.9% compared to the logistic regression model's training accuracy of 94.5%.

These results suggest that while the logistic regression model may have slightly better generalization performance on unseen data, the random forest model has a better ability to fit the training data. This could be due to the fact that the random forest model is more flexible and able to capture non-linear relationships between the features and the target variable.

Overall, both models are effective at detecting credit card fraud, with the logistic regression model being simpler and faster to train, while the random forest model has the potential to achieve higher accuracy with more complex data.