Faculty of Engineering & Applied Science

Software & Computer Security

Final Report:

Credit Card Fraud Detection

Submission date: March 27th, 2023
Group Number: 6
Section CRN: 002
Course Instructor: Khalid Abdul Hafeez
TA: Taghreed Alghamdi

Student Name	Student Id	Contribution
Minhal Syed	100618744	25%
Shahroze Butt	100701891	25%
Rubbia Pasha	100702075	25%
Hemshikha Sultoo	100670616	25%

Table of Contents:

List of Tables

List of Figures

1. Abstract

Financial institutions are generally aware of credit card fraud, which raises worries among banks, credit card providers, and various consumers. However, fraud has long been a persistent problem in society. Not only is user information frequently stolen and exploited for fraudulent purposes, but it also frequently causes problems for the user. Having financial losses and/or exploiting private information while doing so, sometimes without being able to recover those losses. Credit card companies must be able to detect fraudulent credit card operations in order to stop users from being charged for goods they did not buy. In order to identify and stop fraudulent transactions in real-time, our project will create cutting-edge credit card fraud detection software, by combining a variety of algorithms such as decision tree, KNN, and logistic regression/classification model types with a vast amount of data, it may be possible to spot any distinctive patterns that point to fraudulent actions, while also displaying the data for individuals to monitor.

2. Introduction

Credit card fraud is a widespread issue that afflicts the financial sector, with hundreds of millions of dollars lost each year as a result of fraudulent transactions. The increased use of credit cards for both online web and in-person expenditures has really only exacerbated the problem, making it even more critical for financial organizations and business enterprises to take fraud prevention measures. To tackle this problem, countless organizations are increasingly turning to credit card fraud detection applications, which are using sophisticated methods and algorithms to identify and avoid fraudulent transactions in real time.

Throughout this report, we will look at credit card fraud detection tools and their advantages, with an emphasis on using Kaggle's numerical dataset. This dataset includes upwards of 284,000 credit card transactions, approximately 500 of which are fraudulent or counterfeit. We will look at the detection of fraudulent transactions and the advertence in regards to the accuracy of the results through the use of machine learning methods like logistic regression, decision trees, neural networks, and other outlier detection techniques to compare and contrast the results plus their own accuracy . We will also go over the difficulties and constraints of developing a successful credit card fraud detection software, including concerns over confidentiality and security of data information.

Overall, this report offers a thorough breakdown of credit card fraud detection, our project purpose, methodology and implementation, as well as our results and tests conducted. We hope to offer helpful insights into the creation of efficient credit card fraud detection initiatives that can aid in

preventing both customers and enterprises from suffering monetary loss by analyzing the numerical dataset provided by Kaggle and looking at the most recent trends and techniques.

3. Project Objective

The main objective of the project is to develop a secure and reliable software solution that will accurately detect and prevent fraudulent credit card transactions by extracting the dataset from kaggle, after that we will preprocess the data to get rid of any unnecessary information that could affect the accuracy. Moreover, we will apply different learning models to and run the results for each model to then compare the accuracy of each technique for the best optimal solution. Along with that, our goal is to assist banks and merchants in lowering the incidence of fraudulent transactions as a result assisting businesses in their increasing sales. The following objectives of our project would include:

- Obtaining a dataset from kaggle which comes as a csv file with a list of credit card transaction information and work with it. This would be achieved by analyzing user demographics' activity in order to detect, identify, or prevent inappropriate behavior, such as fraud, conduct violations, and any failures like a late outstanding payment that was required to be fulfilled.
- Analyze the dataset of credit card transactions using statistical models and machine learning algorithms such as decision trees, logistic models, and k-nearest neighbors. This will entail refining the framework on Kaggle's numerical dataset to look for trends that can indicate fraud, such as the number of instances of fraud detected for a single person and what instance. As well as fine-tuning the parameter sets to attain maximum efficiency.
- Implementing a safe and user-friendly interface that will allow users to view and access the many types of transactions (fraudulent or not).
- Confirming that the program complies with applicable security norms and rules, such as PCI DSS.
- Maintaining the program under regular review and updating it to maintain its effectiveness.
- This system uses machine learning algorithms to analyze historical transaction data and identify patterns that may indicate fraud. The results will be presented in an approachable way through a web application that will list all the diagrams and results. The model will be built using Python and the scikit-learn library.
- To ensure the software's dependability and accuracy in spotting fraud, it is put through testing and quality control.

- The program's installation on the client's network infrastructure and personnel management training.
- Conduct final QA & UX testing before final release.

To obtain a successful and accurate result there are some limitations that we had to keep in mind due to the time constraint. The limitations are as followed:

- Potential for imbalanced datasets: Given the low incidence of credit card fraud, databases may show a marked discrepancy in the proportion of fraudulent to valid transactions. This might have an impact on our framework's success because it might overfit to the majority class and could potentially miss detecting a fraudulent transaction. To combat this we utilized methods like oversampling the minority class and undersampling the majority class to try and balance the data and regards to enhance the performance of the model on fraudulent transactions.
- Quality of the data: The model's degree of accuracy can be greatly impacted by the calibre of the training data. Poor data , such as incomplete data or inconsistencies, can reduce the program's efficacy and result in inaccurate forecasts. To address this issue we applied data preprocessing methods like normalization, and outlier removal.
- Overfitting: overfitting takes place when a system knows the variability of the information rather than the fundamental patterns (2). Because of this, the program might perform well enough on the training data but badly on new, untainted data. To resolve this, we implement strategies like cross-validation, and stopping it earlier so that it can greatly reduce overfitting and enhance the generalization of the results.
- Privacy issues: Because credit card transactions contain confidential information pertaining to particular individuals, it is important to carefully oversee how it is utilized for fraud prevention and detection in order to comply with all applicable privacy laws, including the GDPR. To address security issues and maintain conformity to relevant data protection laws, we can always use techniques such as data anonymization and secure multi-party computation. But since our findings won't be broadly disseminated, privacy doesn't currently appear to be a major factor because of time limitations and the overall bias/findings of our application.

4. Methodology & Implementation

Although there are many different implementations of the approach for credit card fraud detection systems, we will concentrate on a few of them and use an accuracy score to determine which implementation produces the best results and the accuracy of our model. We are using Comma Separated Values (CSV) files for the data collection so that we may use them as test data. Since CSV files are typically difficult to process and analyze, we create data frames, which structure the table, using tools like NumPy and pandas. We would separate our data into training and testing sets. To prevent unauthorized access or exploitation of sensitive customer information, we would also handle user privacy and security concerns. The sensitive data would be encrypted during transit as well as rest. To assess if a transaction is fraudulent or not, our system's components—the data set, the training data, the model, and our user interface—would work together. We employ and intend to contrast the Decision Tree, KNN, and Logistic Regression classification model types. To maximize the optimum solution for the fraud detection problem, we apply the KNN algorithm and outlier detection techniques. This increases the rate of fraud detection while minimizing false alert rates. As a classification algorithm, we are also using logistic regression. If a variable is categorical, continuous, or binary, the model built by the logistic regression algorithm can be used to characterize the relationship between the variables. This framework allows us to calculate the likelihood that a variable falls into the category of fraud or not. Likewise, it also allows us to to compare the accuracy results of each method (Decision Tree, KNN, and Logistic Regression classification) to give the user a accurate representation on whether a transaction is fraudulent or not

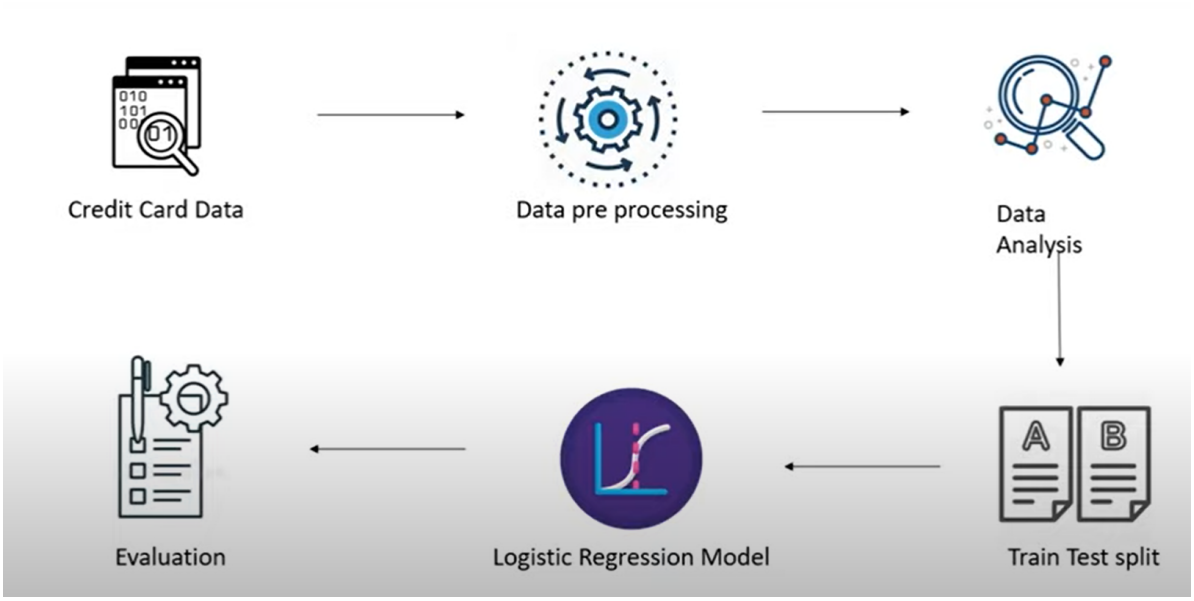

Figure 1.0 Shows the Workflow of how our model will function

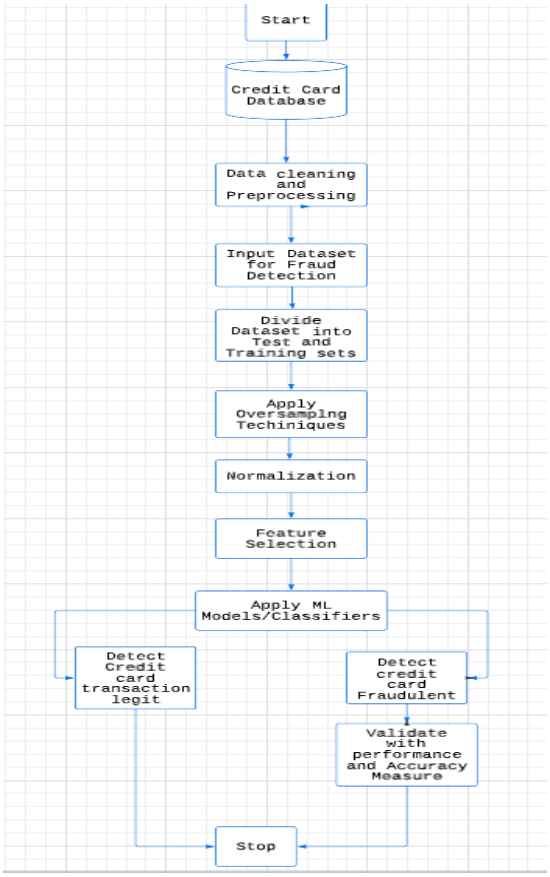

Figure 2.0 Shows the workflow or steps of how we will be implementing the program

Figure 1.0 and 2.0 highlights the details of the project components and better helps us visualize the methodology. After extracting the necessary data from the CSV files, we begin to process the data to remove inconsistencies and ensure that the data is clean for analysis. When operating with datasets, it is essential to fix imbalances because they tend to happen. Additionally, when creating tests, it is best to avoid using data that is either oversampled or undersampled since doing so will damage the model and may result in the loss of information in datasets that may be crucial. We will have evidence that would demonstrate the transaction and its associated value. The dataset that is then constructed for our system would provide many mistakes when employing an imbalanced dataset, indicating that transactions that could be considered fraud are often not.

We spread the dataset to determine where the skew is occurring when we manipulate our dataset in order to see these distortions as seen below.

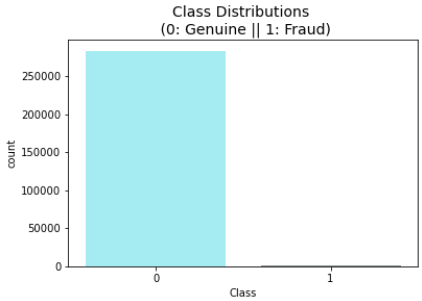

Figure 3.0 Shows the distribution of the Class feature in the imbalanced dataset.

To be able to visually see the data and identify outliers and/or trends, we must expand upon the samples of the data we currently possess and partition them into two distinct diagrams. We sorted them into two categories based on the amount being spent and the time the transactions last. In our hypothetical scenario, we will classify half of those transactions as fraudulent while the other half as not. We are doing this to the data in order to correct the data's imbalance and make it possible for us to see the correlation that does arise.

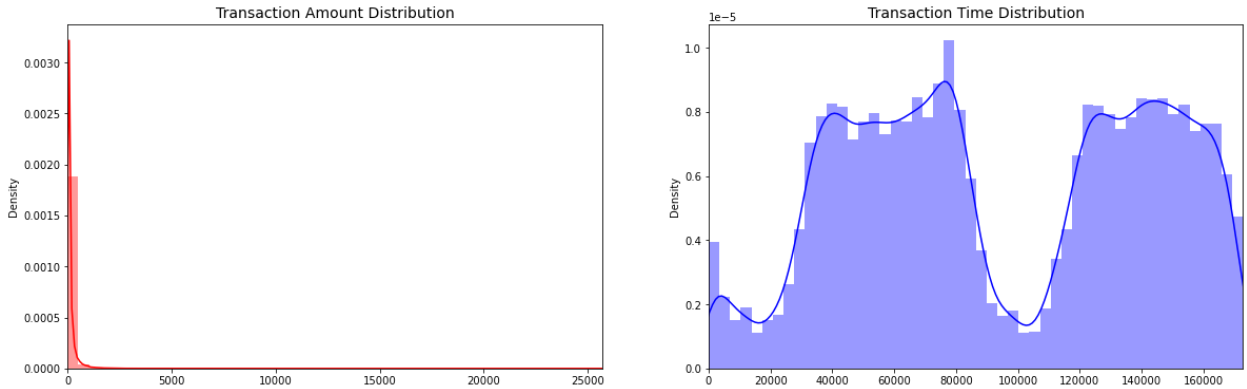

Figure 4.0 Shows the distribution of the Amount and Time feature in the imbalanced dataset.

Once seeing the data we created a random under-sampling that allowed us to remove the data to create a more cohesive dataset thus not resulting in any overfitting. Also after seeing how the data was imbalanced we ended up training it to make sure there was the right amount of cases and that they were precise/correct as well. However, we are doing this with a smaller sample set and not with the entire dataset. Thus we reshuffle the data and allow the system to properly categorize the fraudulent charges alongside the non-fraudulent charges.

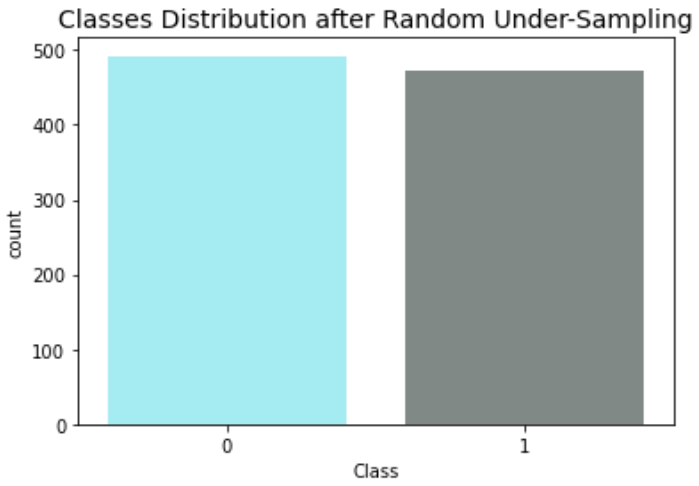

Figure 5.0 Shows the Class of Genuine and Fraudulent to be 50-50 after subsampling.

Then, in order to understand both positive and negative correlations with reference to fraud transactions, we employ correlation matrices. The distribution of these characteristics to fraudulent and non-fraudulent charges can also be seen using box plots to get a better understanding of the data (subsample data).

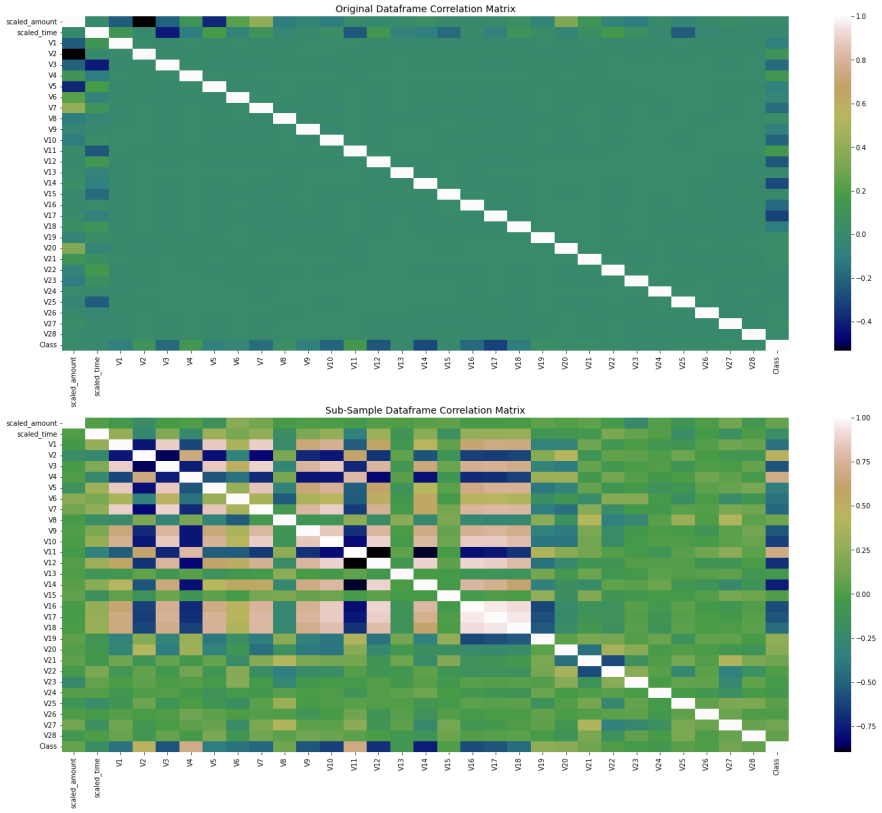

Figure 6.0 Shows the correlation matrix for the original imbalance dataset and correlation matrix for the subsamples.

After looking at the data and the skews that were presented we remove the outliers that have a high correlation so it helps with the accuracy of our models. However, the tradeoff for those would be a lower accuracy by removing extreme outliers rather than outliers. Now working on the algorithm we needed to reduce and cluster.

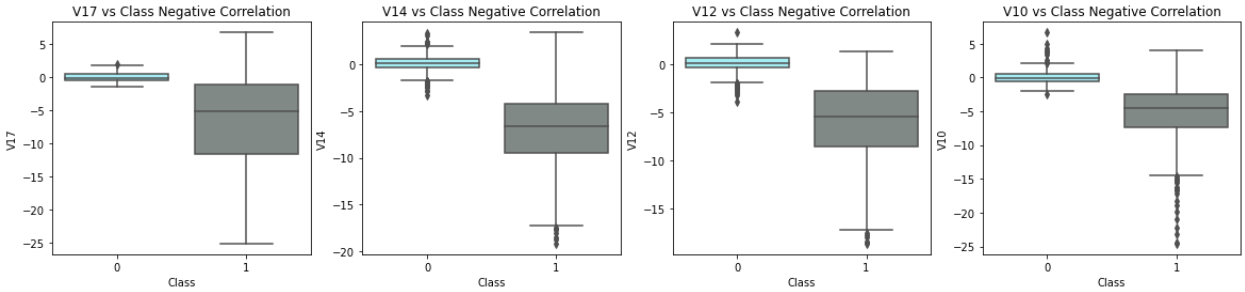

Figure 7.0 Shows the negative correlations to show how they will likely be fraud transactions.

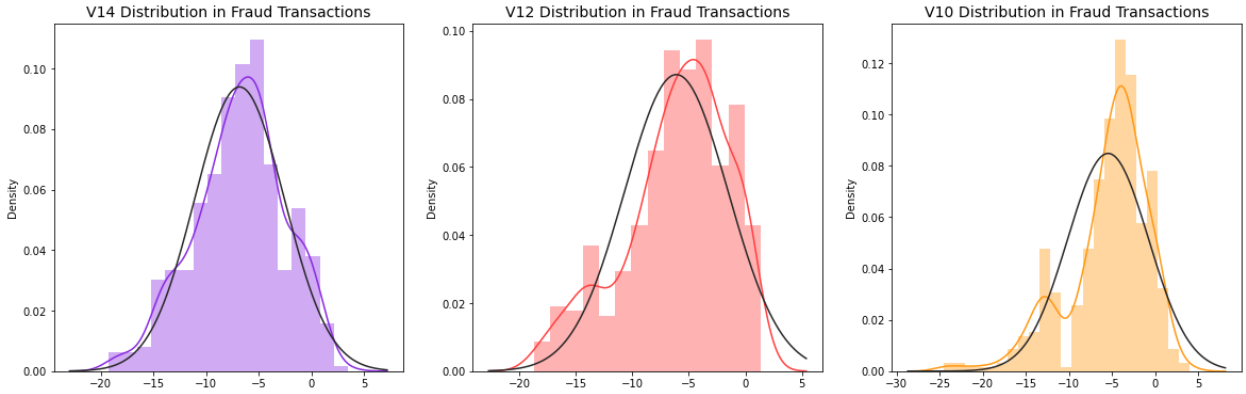

Figure 8.0 Shows the negative correlations distribution.

Comparing scores of different models after sub-sampling and undersampling

```
LogisticRegression : Accuracy Training Score = 94.0 %
KNeighborsClassifier : Accuracy Training Score = 93.0 %
SVC : Accuracy Training Score = 92.0 %
DecisionTreeClassifier : Accuracy Training Score = 88.0 %
```

Scores of Logistic Regression model with sub-sampling and undersampling training and test sets

```
Training Data Accuracy :  0.9524456521739131
Test Data Accuracy :  0.9402173913043478
Precision score :  0.971830985915493
Recall Score :  0.8846153846153846
F1-Score:  0.9261744966442953
```

Scores of Logistic Regression model with original training and test sets

```
Training Data Accuracy :  0.9524456521739131
Test Data Accuracy :  0.9402173913043478
Precision score :  0.971830985915493
Recall Score :  0.8846153846153846
F1-Score:  0.9261744966442953
```

Since F1- score is too low with this method, oversampling using Synthetic Minority Over-sampling Technique (SMOTE) has been performed on the data. The screenshots below compares the scores for Decision Tree Classifier versus Logistic Regression versus Random Forest Classifier. Random Forest Classifier has the highest F-score, therefore, we create a Random Forest Classifier model for the dataset.

```
Decision Tree Classifier
Accuracy Score :  0.9978651113775937
Precision Scoree :  0.9972219195990849
Recall Score :  0.9985091722269694
F1-Score:  0.9978651307720959

Logistic Regression
Accuracy Score :  0.9457102365638286
Precision Scoree :  0.9730053063193439
Recall Score :  0.9167863571077941
F1-Score:  0.9440596098401169

Random Forest Classifier
Accuracy Score :  0.999918238308078
Precision Scoree :  0.9998363993310551
Recall Score :  1.0
F1-Score:  0.9999181929736854
```

Video link on model training, the procedure, approach and genuine transaction detection:
https://drive.google.com/file/d/1Juvts6wE7vcB0gh74S1HDJWBEhFKWv_R/view?usp=share_link

4.1 Data-Sets:

The inputs that will be used to create the models are listed below.

The Kaggle Credit Card Fraud Dataset

This numerical dataset is provided by Kaggle and comes in a CSV format where credit card transactions have been anonymously populated through Principal Component Analysis (PCA) transformation. Each transaction has been labeled as fraudulent or genuine. Out of 284,807 transactions, there are 492 frauds, thus, representing 0.172% of all transactions. These low fraud numbers depict data imbalances which can be dealt with through several techniques discussed in the report. To present the features (characteristics that define the credit card fraud problem) from the dataset:

i) Features V1 to V28: These are the principal components that have been PCA transformed.

ii) Feature Time: This feature has not been PCA transformed and represents the time elapsed between transactions.

iii) Feature Amount: This feature represents the amount of money spent in each transaction, where each row represents one transaction.

iv) Feature Class: This feature categorizes the transaction into fraudulent and genuine transactions through values 1 and 0 respectively.

1) The train set and validation set

The dataset will be split into;

i) Training set - This set of data will be used to train the model.

ii) Validation set - This set of data will be used to tune the parameters/variables in order to construct an accurate model.

iii) Test set - This set of data will be used to evaluate the performance of the model.

The outputs of the models are listed below; they serve as a measure to help understand which model is better and more efficient.

1) Recall Score - This score ranges between 0 and 1 and measures the ratio of true positives versus false negatives. Ideally, we would want it to be closer to 1.
2) Precision Score - This score measures the success of prediction when dealing with imbalanced data. Ideally, a score of 90% or more represents a better model.
3) F1 Score - This score is the average of the recall and precision scores and works well on imbalanced data. Ideally, we would want it to be closer to 1.

4) Accuracy Score - This score is the ratio of the number of correct predictions versus the total number of predictions. Ideally, a score of 90% or more is better.

The inputs to test the trained model:

1) Test set - This set of data will be used to evaluate the performance of the model.

The output of the trained model:

1) The output is displayed at the bottom of a GUI that takes the features V1 to V28 and Amount as input. Furthermore, the detection is also printed on the notebook.
2) Output looks like the following:

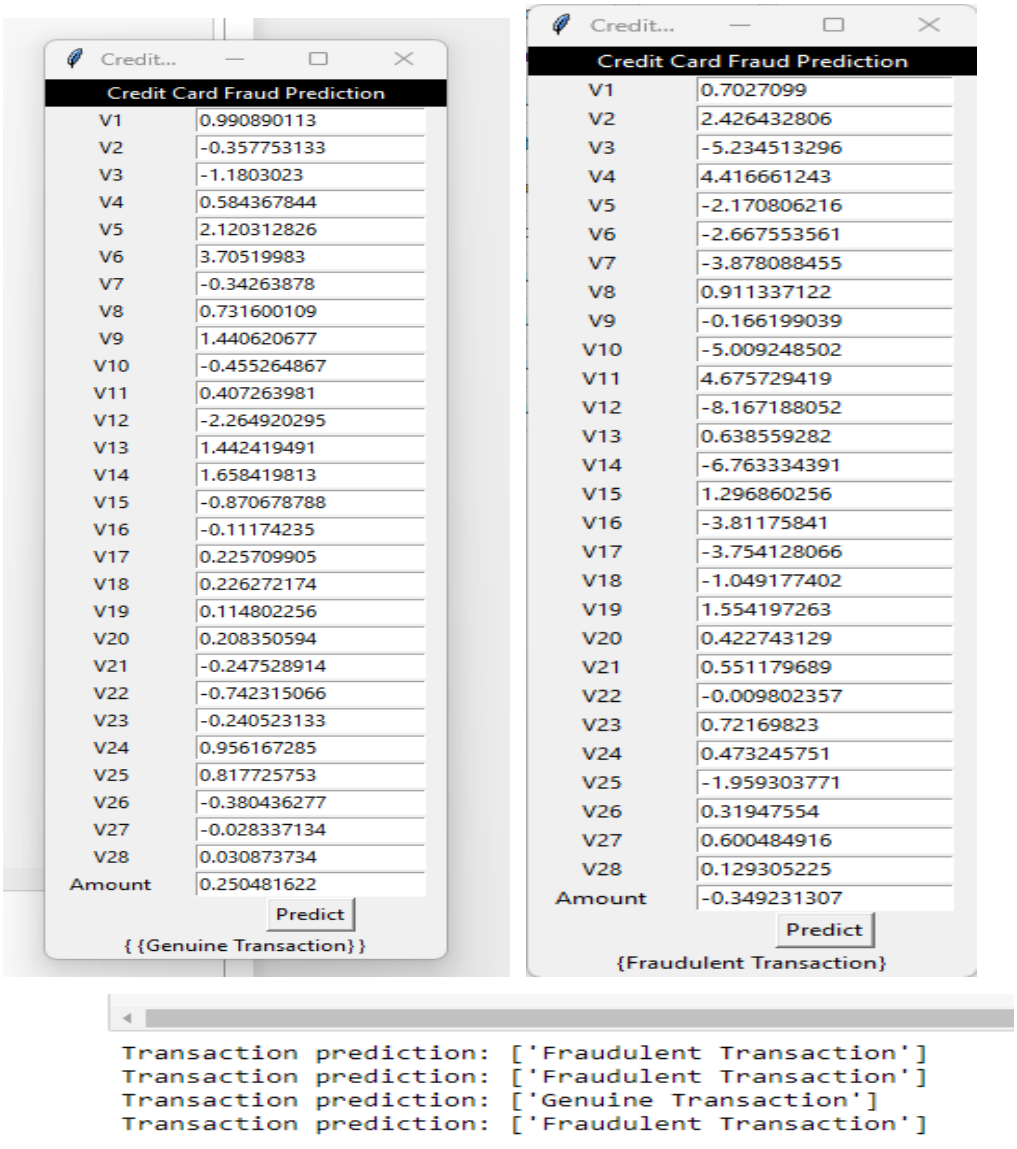

Figure 9.0: Output

Figure 9.0 below shows the sequence diagram which maps out the course of action for each functionality in the application. The diagram below depicts the nature of how a user makes a transaction by logging in their credit card info to a site or a machine, then that transaction gets recorded to a dataset in our case we are using the kaggle dataset as mentioned above. The application has 3 main functions: first gathering and cleaning the data of any unnecessary or repetitive information. Second is the classification of the data by implementing different machine learning algorithms such as logistic regression, random forest, K-nearest neighbors, and decision trees, which should help in not only detecting credit card fraudulent activity but it will also help in assuring an accurate result as we will be implementing many methods. The last function is testing the models and outputting an accuracy score for each machine learning model to determine how accurately our program detects credit card fraud. That being said if no fraud is detected it will be ruled out as a legit credit card transaction done by the user.

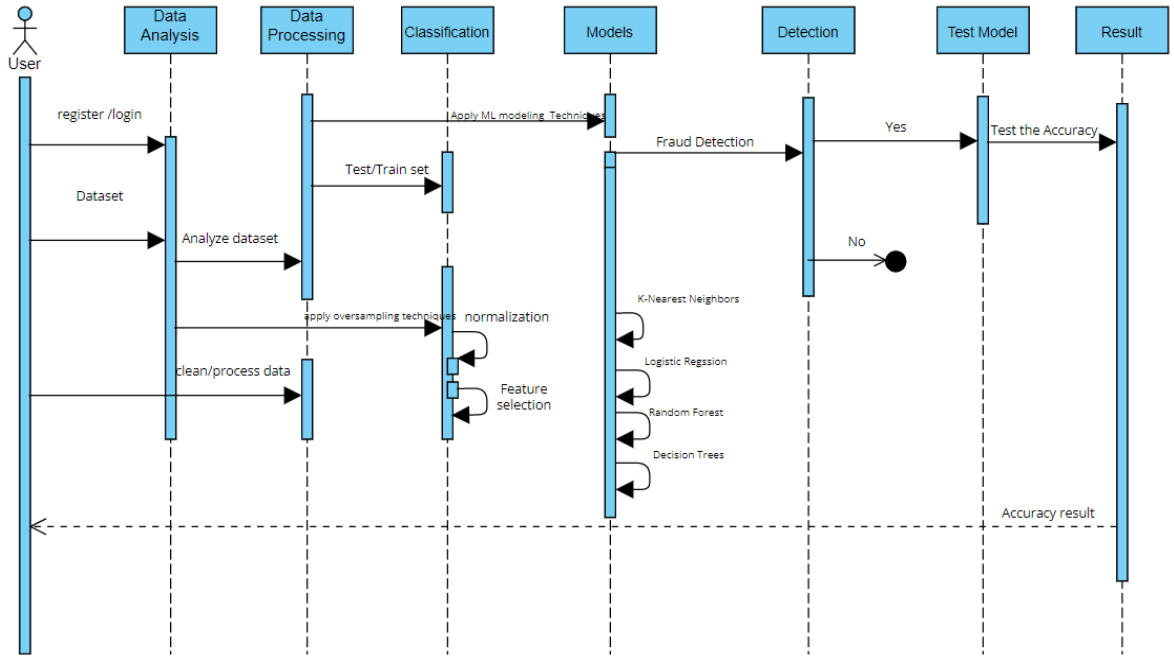

Figure 10.0 Sequence Diagram

5. Results & Discussion

As demonstrated above, the program was able to effectively classify transactions as genuine or fraudulent with a high degree of accuracy. Nevertheless, due to the system's complexity, we simplified it for users by ensuring that when a new transaction is submitted for evaluation, the program interprets the data using the trained machine learning algorithm and outputs the results on a graphical user interface

(GUI), making it straightforward for the user to comprehend and take action on the programs result. So, if the model concludes that the transaction is genuine (Class 0), the program displays a Genuine Transaction on the GUI to show that the transaction is genuine. However, if the model determines that the transaction is fraudulent (Class 1), the program will generate an output of Fraudulent Transaction on the GUI to indicate that the transaction is indeed fraudulent. The picture below shows the GUI and the generated outputs being fraudulent and being genuine.

Genuine Transaction (towards the end of the video):
https://drive.google.com/file/d/1Juvts6wE7vcB0gh74S1HDJWBEhFKWv_R/view?usp=share_link
Genuine Transaction Screenshot:

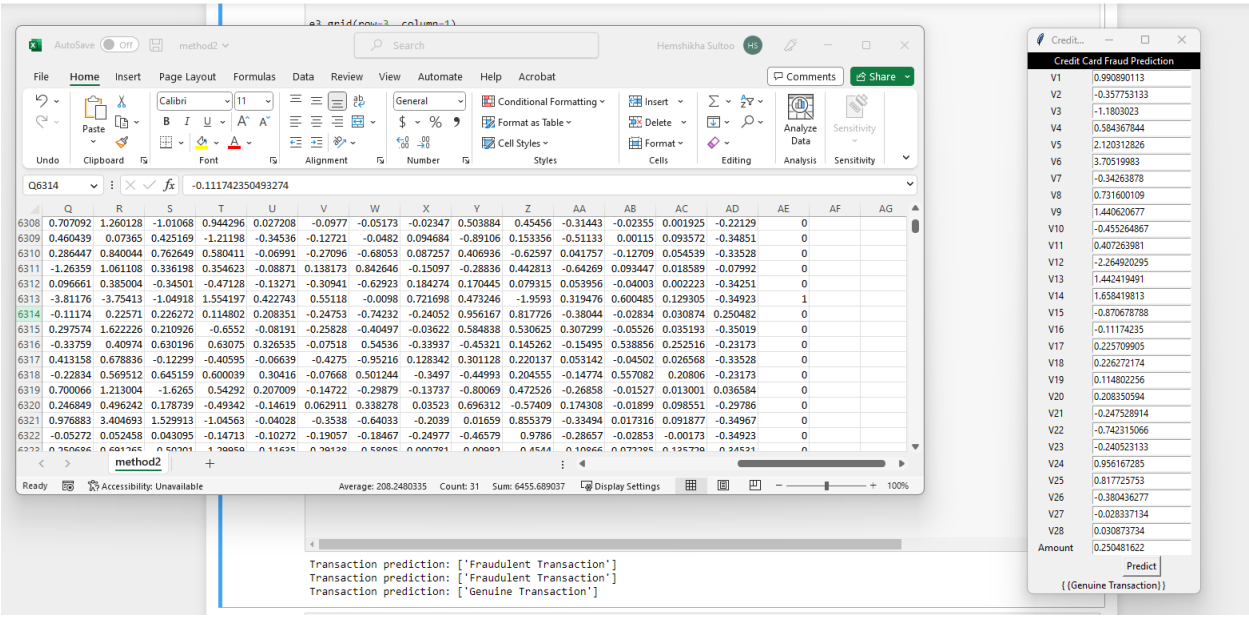

Figure 11.0 Detection of Genuine Transaction

Fraudulent Transaction Video Link:
https://drive.google.com/file/d/1lY1wTBr7yfV8jktKXw_WHeOcbvLbw4Fn/view?usp=share_link

Fraudulent Transaction Screenshot:

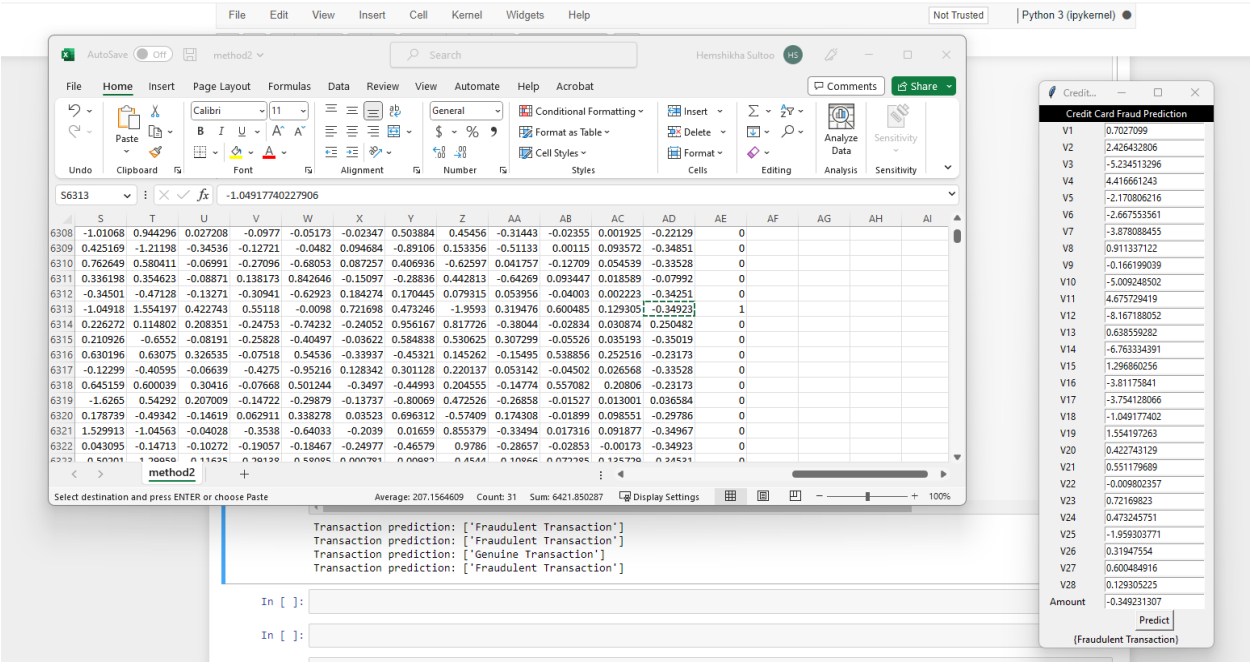

Figure 12.0 Detection of Fraudulent Transaction

Furthermore, we ended up adding a GUI to show the detection of fraud and genuine findings to assist in making the program easier to navigate and more accessible to people, especially for users with no technical expertise who might not be familiar with complex statistical methods. This implies that the user is able to quickly and easily determine that the transaction has been categorized as genuine or fraudulent without having to comprehend the more intricate outputs shown in the implementation section. The user should be able to understand that the model determines that the transaction is fraudulent, if the output is straightforward numerical outcome such as "1" and "0" otherwise to denote genuine transactions.

Code, Powerpoint and Final Report Link:

https://drive.google.com/file/d/1ppGW8j2l1VVUXB9PvioVyAnQIM09M6td/view?usp=share_link

6. Conclusion & Recommendations

In summation, we think it is critical to create a system that effectively detects credit card fraud if you want to stop economic losses and safeguard customers from scams. In our system we used these models for accurate and efficient fraud detection which were created using machine learning methods like logistic regression, decision trees, and KNN. These models could, however, also have drawbacks like unbalanced datasets, poor data quality, overfitting, confidentiality issues, and insufficient resources. To overcome those concerns we implemented solutions like oversampling, and undersampling, to balance the dataset in order to meet these shortcomings. Normalization, cross-validation, and catching errors early in the program are data preprocessing methods that can reduce overfitting while also improving the quality of the data. We also researched possible tactics that protect privacy and guarantee conformance with applicable laws.

In the future, we intend to consider ways to employ more advanced techniques for machine learning such as deep learning, AI techniques, and anomaly detection. Likewise, integrating user behavior, location information, as well as other elements into the program can also assist with enhancing its precision and performance. Additionally, we would analyze the effect of various regulatory systems and data security implications on the establishment of our credit card fraud detection application.
On the whole, there remains a great deal of scope for advancement and progress for the establishment of an even more effective credit card fraud detection system. Due to the time constraint we still see more room to create an even more accurate and effective identification of credit card fraud systems to safeguard customers and limit financial losses by employing the most advanced algorithms for machine learning and AI.

7. Project Timeline

Task	**Duration**	**Details**
Obtain and analyze dataset	February 7th, 2023 to February 10th, 2023	Obtain multiple datasets and analyze the accuracy
Apply Machine Learning Algorithms	February 11th, 2023 to February 28th, 2023	Statistical models and Machine Learning algorithms to detect fraudulent transactions.

Conduct Tests of the system	March 4th 2023 to March 10th 2023	Test the completed software to determine accuracy and confidence levels.
Perform QA & UX Testing	March 11th 2023 to March 20th 2023	Perform additional quality assurance and user experience testing.
Complete Final Report	March 21th 2023 to March 27th 2023	Prepare a report consisting of all steps in creating the software as well as reporting on the the dataset of the user and apprehending to a scale of accuracy to determine if our results are accurate.
Complete Presentation	March 25th 2023 to March 27th 2023	Prepare a presentation consisting of our findings and a demo.

Table 1.0 Project Timeline

We also wanted to have several phases that we intended to complete our project inside while creating the project timeline.

1. Data Preparation
2. Model Development
3. Deployment

While examining the transaction data, these characteristics enable us to compile information that we can handle and that our system can comprehend. As well as, we educate our system to comprehend what we must learn from these massive amounts of data. Likewise, different models were developed to see which would be most effective at detecting credit card purchases. Last but not least, we carefully constructed our system during deployment to enable users to access the data deployed within our program so it can be observed by these institutions that other users are criticizing.

8. References

[1] D. Varmedja, M. Karanovic, S. Sladojevic, M. Arsenovic and A. Anderla, "Credit Card Fraud Detection - Machine Learning methods," 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 2019, pp. 1-5, doi: 10.1109/INFOTEH.2019.8717766.

[2] The University. (1978). What is OverFitting Amazon. Retrieved March 25, 2023, from https://aws.amazon.com/what-is/overfitting/

[3] https://www.facebook.com/jason.brownlee.39. “8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset.” *Machine Learning Mastery*, 7 June 2016, machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/. Accessed 27 Mar. 2023.

[4] Mazumder, Saikat. “What Is Imbalanced Data | Techniques to Handle Imbalanced Data.” Analytics Vidhya, 21 June 2021, www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/. Accessed 27 Mar. 2023.