AI-ML Bootcamp Capstone Project

Title: Prediction of Credit Card Fraud

Name: Subodh Wasnik

Abstract

As credit card transactions become increasingly prevalent in the European payment system, the risk of stolen account numbers and resulting losses to banks rises. Credit card fraud is perceived as a growing threat with significant implications for the financial industry. Detecting fraud in these transactions is crucial, and data mining plays a vital role in both online and offline scenarios.

However, credit card fraud detection presents challenges due to the ever-changing nature of normal and fraudulent behaviour, coupled with the highly imbalanced nature of credit card fraud datasets. The effectiveness of fraud detection relies on factors like dataset sampling methods, variable selection, and detection techniques.

This paper explores the performance of two methods, logistic regression (LR) and Random Forest (RF), in detecting credit card fraud. The dataset comprises 284,807 transactions from European cardholders. To address the dataset's imbalance, a combination of under-sampling and oversampling techniques is applied to both raw and pre-processed data. The study is conducted using Python, and the performance metrics include accuracy, precision, and recall rate.

Content

List of Figures

List of Tables

1. Introduction

1.1 Background

Credit cards play a pivotal role in modern financial transactions, particularly in online purchases and payments. Their convenience makes them a widely used financial product for consumers. However, this convenience comes with risks, and one significant threat is credit card fraud. Credit card fraud involves the unauthorized use of someone else's credit card or its information to make purchases or withdraw cash. In the digital age, where online transactions are prevalent, the occurrence of credit card fraud has become a significant concern. This issue not only poses financial risks to individuals but also challenges credit card companies to ensure the security and integrity of their services.

1.2 Objective

The primary objective of this study is to develop effective strategies and models to recognize and prevent credit card fraud. Given the highly imbalanced nature of the dataset, the focus is on creating machine learning algorithms that can accurately identify fraudulent transactions without compromising the detection of legitimate ones. The specific aim is to train models capable of distinguishing between genuine and fraudulent transactions based on patterns and features within the dataset. By achieving this, credit card companies can enhance their fraud detection mechanisms, minimizing the financial impact on cardholders and reinforcing the overall security of credit card transactions.

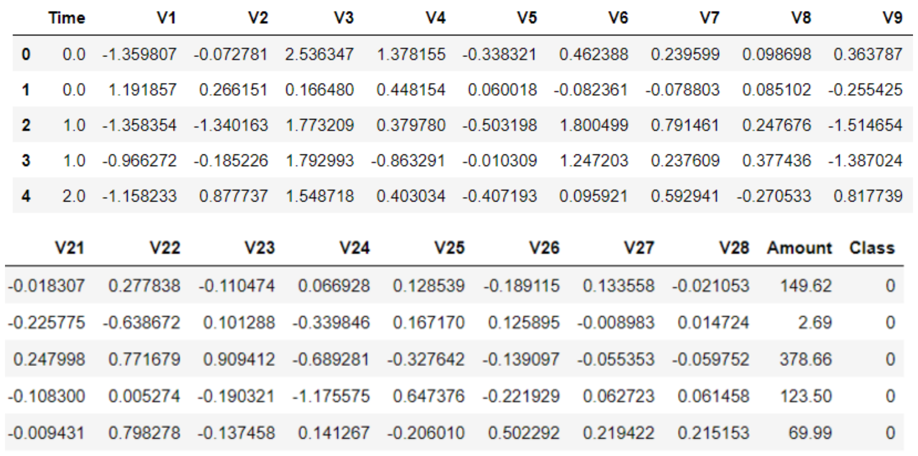
1.3 Significance

The significance of this research lies in its potential to mitigate the adverse effects of credit card fraud on both consumers and credit card companies. As online transactions continue to grow, so does the threat of fraudulent activities. Enhancing fraud detection mechanisms is essential for maintaining the integrity of financial systems and the trust of consumers. Successfully addressing this issue not only safeguards individuals from unauthorized charges but also protects the reputation of credit card companies. Moreover, as digital transactions become more ingrained in daily life, the findings of this study can have broader implications for the broader financial industry, contributing to the ongoing efforts to create secure and reliable payment systems. Ultimately, the significance lies in striking a balance between convenience and security in the realm of credit card transactions.

2. Data Exploration

2.1 Data Exploration and Overview

The dataset consists of 31 columns, with the first column representing 'Time' and columns 2 to 29 containing confidential features denoted as V1 through V28, as shown in Table 1: Original Dataset and Table 2: Feature of Dataset. It's essential to note that these features have undergone Principal Component Analysis (PCA), a technique that transforms and simplifies the data while retaining its important characteristics. However, the 'Amount' and 'Time' features have not undergone PCA, providing direct insights into the original attributes of transaction amounts and the timing of transactions. This transparency in the dataset's composition aids in understanding the nature of the features, contributing to a more informed analysis.



	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739

V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

Table 1: Original Dataset

Two crucial columns, namely 'Amount' and 'Class,' play pivotal roles in our exploration. 'Amount' represents the transaction amount, offering a glimpse into the financial aspect of each transaction. Meanwhile, the 'Class' column serves as the response variable, categorizing transactions as either fraudulent ('1') or non-fraudulent ('0'). This binary classification simplifies the focus of our analysis, enabling a clear evaluation of the model's ability to discern between fraudulent and non-fraudulent activities. As we delve into the exploration of this dataset, the distinct roles of these columns become central to understanding the predictive capabilities of the model and its effectiveness in identifying instances of credit card fraud.

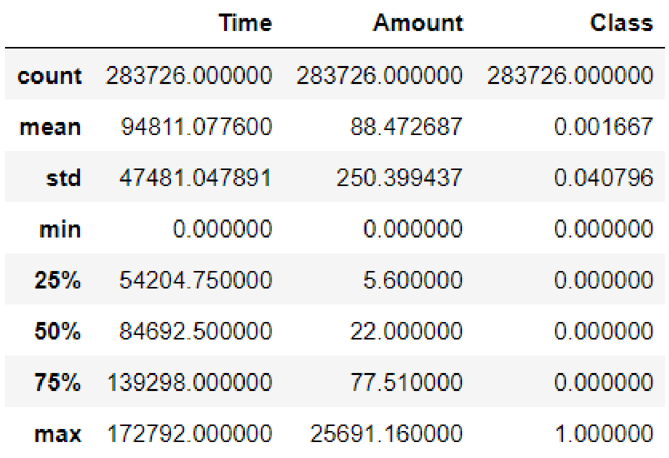
Exploring the dataset's features and their original attributes, as well as considering the binary nature of the 'Class' variable, allows for a more granular examination of the model's performance. The absence of PCA transformation in 'Amount' and 'Time' ensures that these specific elements retain their inherent characteristics, providing a direct link to the real-world aspects of transaction amounts and timing. This level of transparency in the dataset enhances the interpretability of the model's outcomes, contributing to a comprehensive assessment of its ability to accurately distinguish between fraudulent and non-fraudulent transactions.

S No.	Feature	Description
1.	Time	Time in seconds to specify the elapse between the current transaction and first transaction.
2.	V1 – V28	Principal Components unnamed due to confidentiality
3.	Amount	Transaction Amount
4.	Class	0 - Not Fraud 1 - Fraud

Table 2: Feature of Dataset

2.2 Data Cleaning

While exploring the dataset, it came to our attention that 1081 rows contained duplicate values. To uphold data cleanliness and minimize the potential introduction of bias into the model, a deliberate decision was made to exclude these duplicate entries. This meticulous data refinement process aims to ensure that the model is trained on a distinct and representative set of observations, free from redundancy. By taking this step, we seek to enhance the model's ability to generalize well to new, unseen data, ultimately contributing to the development of a more robust and reliable machine learning framework.



	Time	Amount	Class
count	283726.000000	283726.000000	283726.000000
mean	94811.077600	88.472687	0.001667
std	47481.047891	250.399437	0.040796
min	0.000000	0.000000	0.000000
25%	54204.750000	5.600000	0.000000
50%	84692.500000	22.000000	0.000000
75%	139298.000000	77.510000	0.000000
max	172792.000000	25691.160000	1.000000

Table 3: Statistical Values of features

The exclusion of duplicate entries is essential for preventing the model from inadvertently learning from repeated instances, which could lead to overfitting and reduced performance on real-world scenarios. This careful curation of the dataset is a proactive measure to maintain the integrity of the model's training data, promoting a more accurate understanding of patterns and relationships within the information available. Through this data cleaning process, we aim to optimize the model's predictive capacity and reinforce its potential for providing dependable outcomes in the detection of fraudulent credit card transactions.

2.3 Data Imbalance

Fraudulent transactions make up only a tiny fraction, approximately 0.167%, of the dataset, with the vast majority, around 99.833%, consisting of normal transactions, as in Table 4: Total

Fraud and non-fraud Transactions and Figure 1: Class Imbalance Analysis. This significant imbalance emphasizes the need for careful consideration and appropriate strategies during the model training phase to tackle potential biases. It's crucial to ensure that the model can adeptly distinguish and predict instances of fraud, even though they are scarce in the dataset.

Class	No. of Transactions	Percentage of transaction
0 (non-fraud)	283253	99.833%
1 (Fraud)	473	0.167%

Table 4: Total Fraud and non-fraud Transactions

To enhance the model's sensitivity to fraudulent activities, several techniques can be explored. These may include oversampling, where the minority class (fraudulent transactions) is replicated to balance the dataset, or under sampling, where instances of the majority class (normal transactions) are reduced. Additionally, the use of specialized algorithms designed for imbalanced data can be considered. Employing these strategies is vital in creating a more resilient and effective machine learning model, capable of accurately identifying and predicting instances of credit card fraud, despite their infrequent occurrence in the dataset.

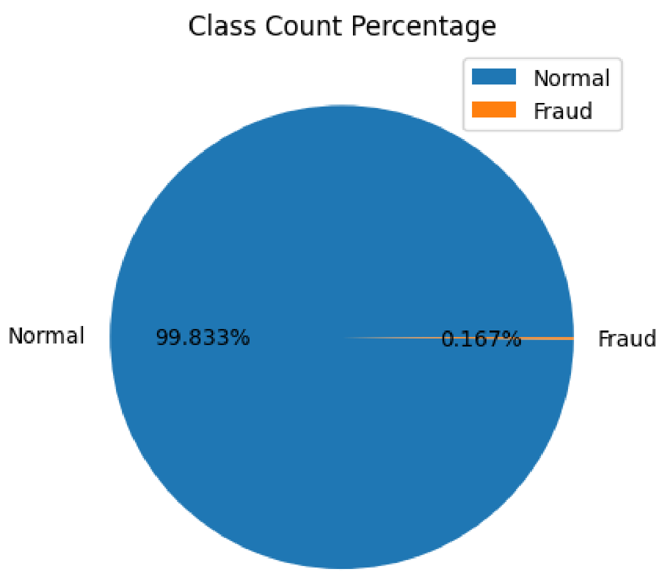


Figure 1: Class Imbalance Analysis

2.3 Feature Analysis

Feature analysis is the process of scrutinizing and assessing individual features in a dataset to determine their significance and impact on a machine learning model's performance, aiding in the identification of key variables crucial for predictive accuracy.

2.3.1 Time Feature

The dataset reveals a lack of discernible temporal trends as seen in Figure 2: Number of Fraud and Non-fraud transactions in hours and Figure 3: Number of Fraud and Non-fraud transactions in minutes, suggesting that the raw timing of transactions might not be a significant factor in identifying fraudulent activities. While the 'Time' feature lacks an apparent pattern, its overall importance is not diminished; rather, it underscores the necessity of relying on other pertinent features for effective fraud detection within the dataset.

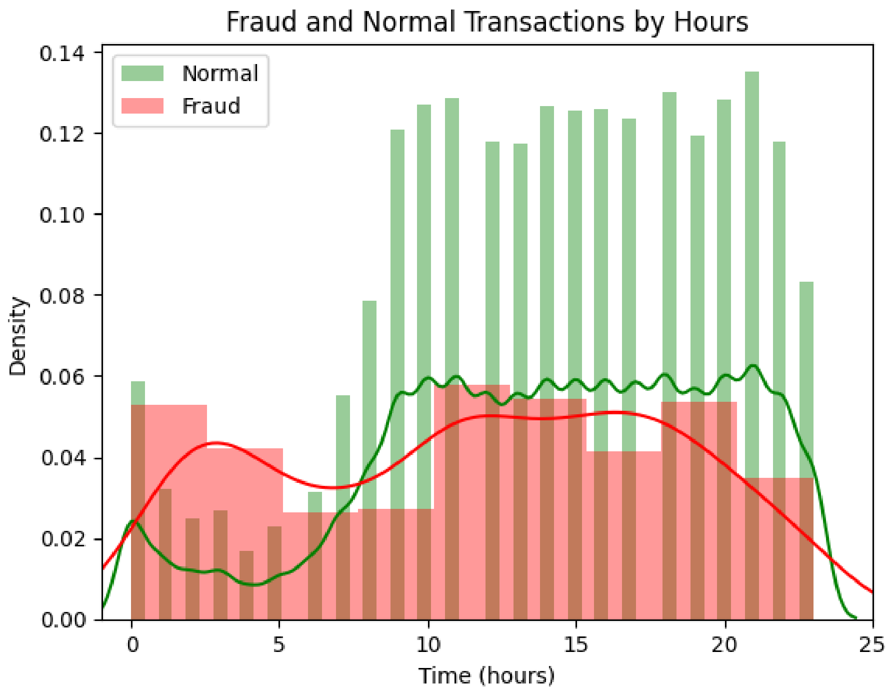


Figure 2: Number of Fraud and Non-fraud transactions in hours

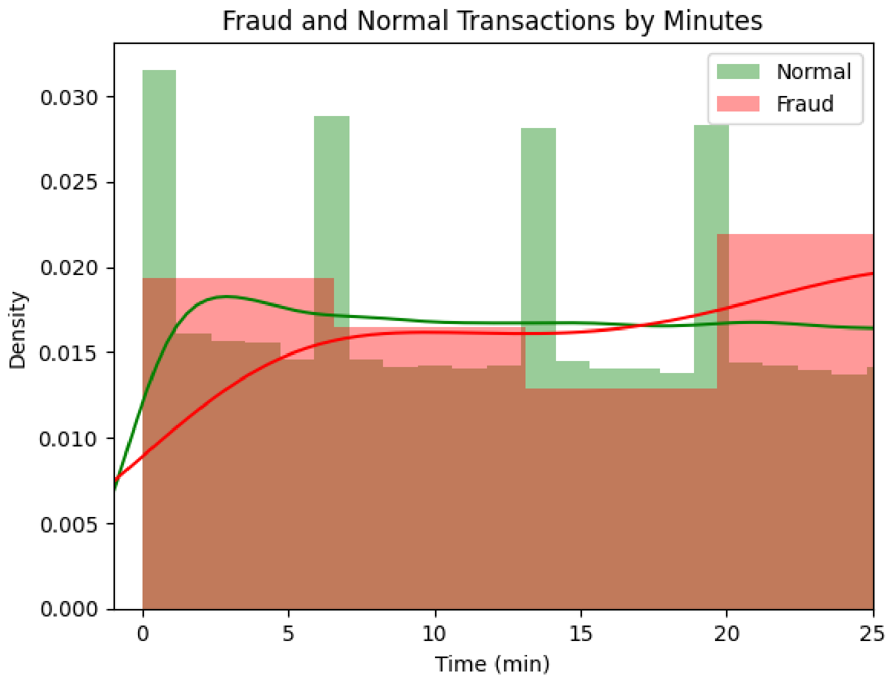


Figure 3: Number of Fraud and Non-fraud transactions in minutes

This observation prompts a more thorough exploration of features beyond temporal aspects, emphasizing the importance of considering a diverse set of information for the model to accurately identify and predict instances of fraud. By broadening the scope of feature analysis, we aim to ensure that the model can leverage various characteristics, apart from timing alone, to enhance its ability to detect and respond to fraudulent transactions effectively.

2.3.2 Amount vs Class

The statistics for the 'Amount' feature reveal notable differences between fraudulent and normal transactions. For fraudulent transactions, the mean amount is higher at 123.87 compared to 88.41 for normal transactions. The standard deviation is also significantly larger for fraudulent transactions, indicating greater variability in the amounts associated with fraud.

Moreover, the upper quartiles (75th percentile) for fraudulent transactions are considerably higher, with a value of 105.89 compared to 77.46 for normal transactions. This suggests that a substantial portion of fraudulent transactions involves larger amounts. Additionally, the maximum amount for a fraudulent transaction is 2125.87, while for normal transactions, it is substantially lower at 25691.16.

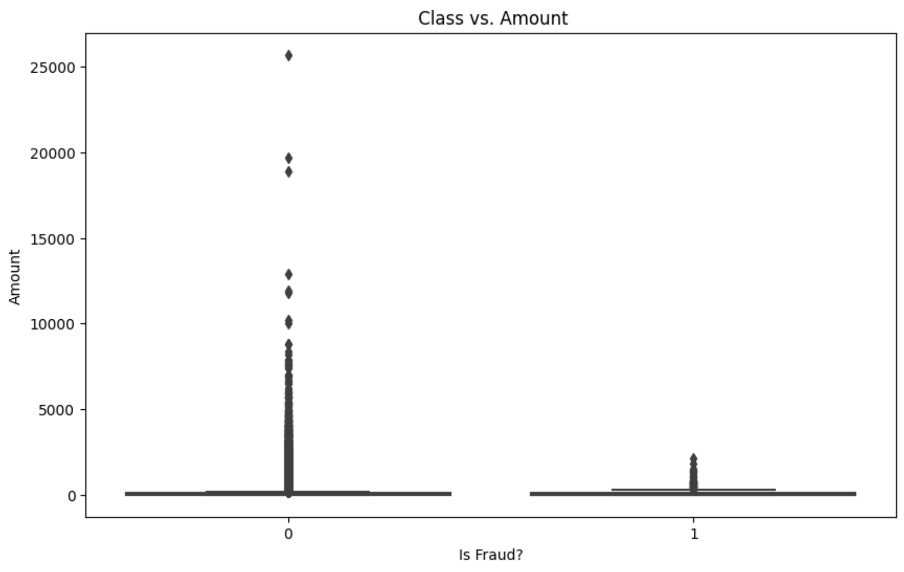


Figure 4: Boxplot describing the distribution of Amount for each class, Fraud and non-fraud

The boxplot in Figure 4: Boxplot describing the distribution of Amount for each class, Fraud and non-fraud illustrates that the 'Amount' statistics for both fraud and non-fraud transactions have high variability, making it challenging to interpret. To enhance readability and address outliers, we opted to transform the 'Amount' values using “np.log(df.Amount + 0.01)” before plotting the graph again. This logarithmic transformation helps mitigate the impact of extreme values and provides a clearer visualization of the transaction amounts for both classes.

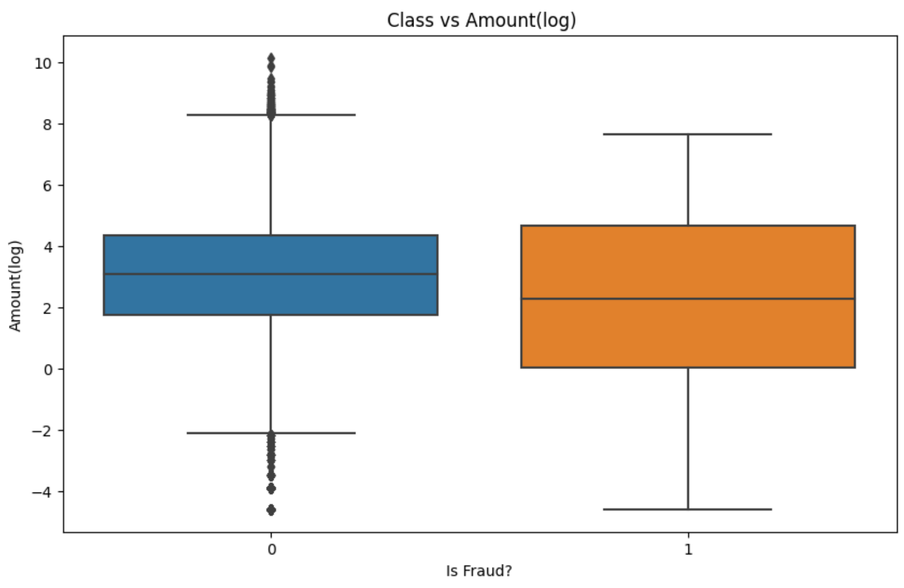


Figure 5: Boxplot describing the distribution of the log of 'Amount', for each class

From Figure 5: Boxplot describing the distribution of the log of 'Amount', for each class, a clear difference appears when comparing fraud and normal transactions based on their Interquartile Range (IQR). Fraudulent transactions show a higher IQR, indicating more variation in the data compared to normal transactions. Interestingly, normal transactions have higher individual values, especially in the upper part of the data, suggesting that the upper range of values for normal transactions extends further than that of fraudulent transactions.

A closer look at the statistical characteristics reveals significant variations between real and fraudulent transactions. Real transactions have a larger mean value, higher Q1, smaller Q3 and Q4, and more significant outliers. In contrast, fraudulent transactions show a smaller Q1 and mean, larger Q4, and fewer smaller outliers.

These notable differences highlight the importance of these features in distinguishing between the two types of transactions. Knowing these distinct statistical profiles is essential as it guides the machine learning model in capturing the patterns indicative of fraud more effectively. This insight significantly boosts predictive accuracy, allowing the model to better identify and predict instances of fraudulent activities with increased precision.

2.3.3 Amount vs Time

When plotting the 'Amount' and 'Time' (in hours and minutes) on a scatter plot using `sns.lmplot`, no apparent correlation was observed between these two variables (see below Figure 6: Scatter plot of transaction of amount at given time hour and Figure 7: Scatter plot of transaction of amount at given time minute). The scatter plot did not reveal any discernible pattern or trend, indicating that the transaction amount does not exhibit a linear relationship with the time of the transaction.

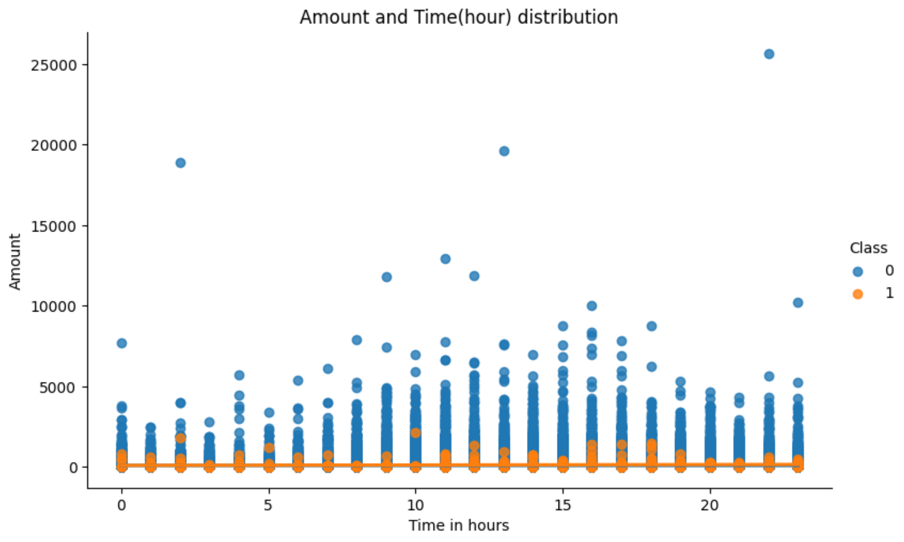


Figure 6: Scatter plot of transaction of amount at given time hour

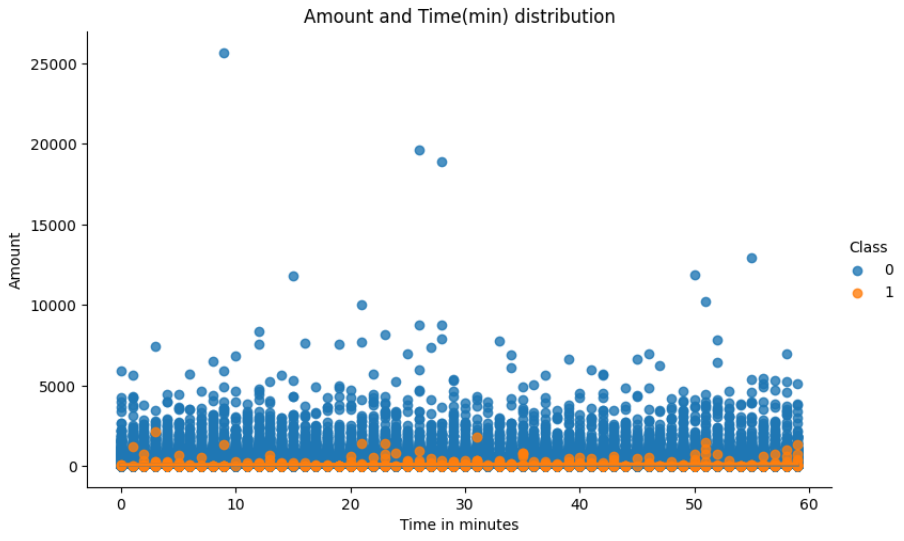


Figure 7: Scatter plot of transaction of amount at given time minute

This lack of correlation suggests that the timing of transactions, at least when considered in conjunction with the amount, does not follow a predictable pattern. The absence of a clear trend in the scatter plot reinforces the idea that these two features may operate independently of each other in terms of their influence on credit card transactions. Understanding such relationships is crucial for feature selection and model training, ensuring that irrelevant or non-correlated features do not contribute unnecessary complexity to the machine learning model.

2.3.4 V1-V8 Features

We have an additional 28 features labelled as V1 to V28. These features are confidential and have undergone Principal Component Analysis (PCA) transformation for security reasons. To understand their distribution, we created separate distribution plots (distplots - Figure 8: Feature Distribution) for each of these features. Distplots provide a visual representation of the distribution of values within each feature, allowing us to explore their patterns and variations.

Analysing these transformed features is essential for gaining insights into their characteristics, even though the specific details remain confidential to ensure the security of sensitive information.

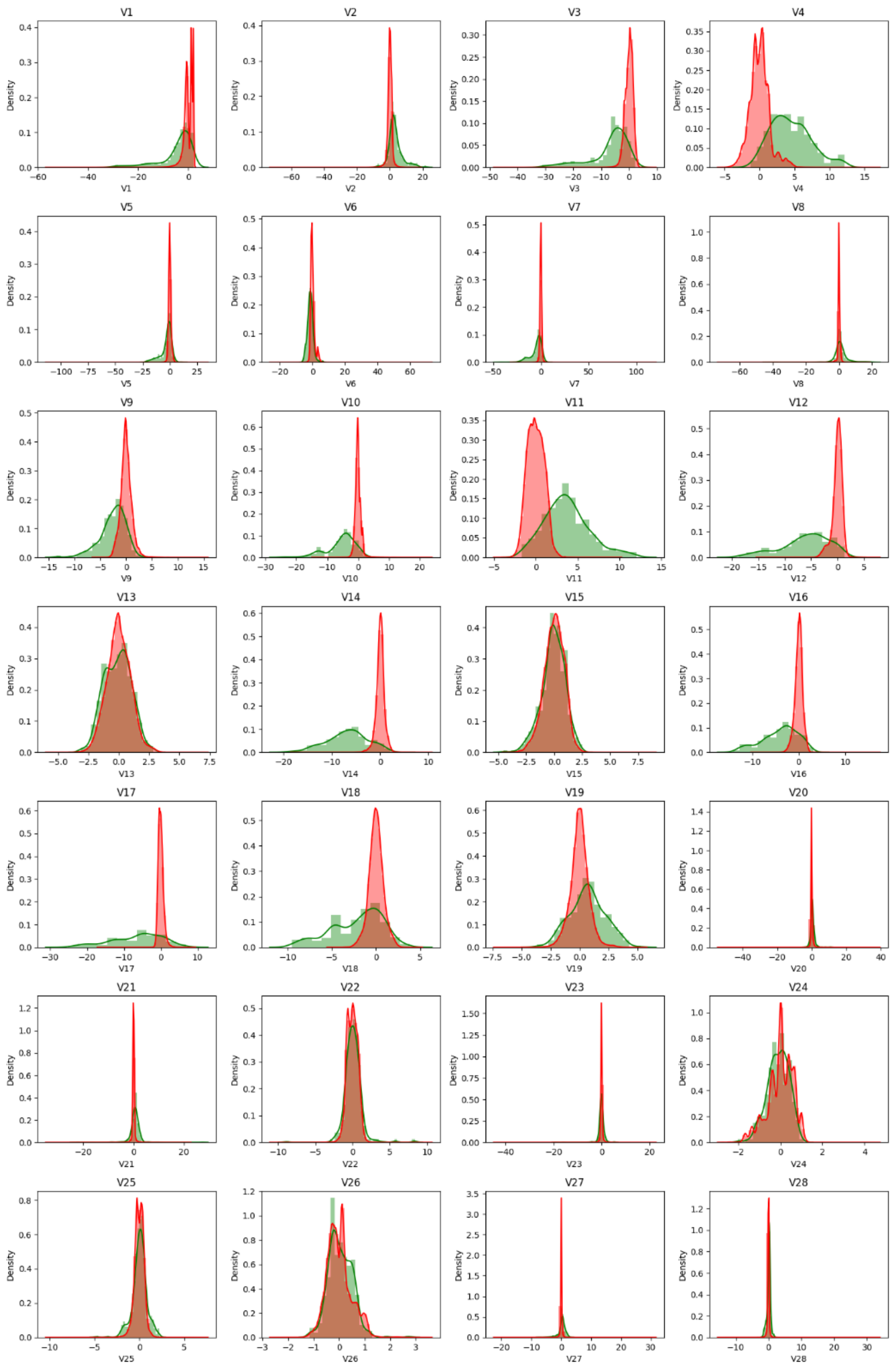


Figure 8: Feature Distribution

Upon closer inspection of the feature distributions concerning Class values 0 and 1, we notice distinct patterns in how certain features behave. Features like V4 and V11 clearly show separations between the two classes, indicating that these features have specific values that are more prevalent in one class over the other. Similarly, V12, V14, and V18 exhibit partial separations, suggesting that certain values of these features contribute to distinguishing between legitimate transactions (Class = 0) and fraudulent transactions (Class = 1).

Conversely, features like V1, V2, V3, and V10 present unique profiles that help differentiate between Class values. These features show varied distributions for the two classes, indicating that their values play a role in classifying transactions as either normal or fraudulent. On the other hand, features V25, V26, and V28 showcase similar distributions for both Class values, suggesting that these features may not provide strong discriminatory information.

A general trend observed is that legitimate transactions (Class = 0) often have distributions centred around 0, sometimes with extended tails, while fraudulent transactions (Class = 1) display skewed distributions. This nuanced understanding of feature distributions provides valuable insights for the model, helping it identify specific attributes that contribute to distinguishing between normal and fraudulent activities. These insights are crucial for developing a robust fraud detection model that can accurately classify transactions based on their unique feature patterns.

2.4 Feature Engineering

To enhance the model's resilience and ensure robust performance, we've decided to employ the logarithmically transformed 'Amount_log' instead of the original 'Amount.' This strategic choice stems from the observation that the 'Amount' plot displays a notable number of outliers and substantial variability in the Interquartile Range (IQR) boxplots for both fraud and normal transactions.

The log transformation aims to mitigate the impact of outliers and address the skewed data distributions present in the 'Amount' feature. This transformation helps create a more stable and predictive model by reducing the influence of extreme values. By using 'Amount_log,' the model becomes better equipped to handle variations in transaction amounts, contributing to a fraud detection framework that is more accurate, reliable, and less susceptible to the challenges posed by skewed data. Overall, this feature engineering decision is intended to optimize the model's performance and improve its ability to discern patterns indicative of fraudulent activities.

2.5 Feature Correlation

Feature correlation explores the relationships between different features in a dataset, revealing how changes in one variable may correspond to changes in another. Understanding feature correlation is essential in machine learning, as it helps identify patterns, redundancies, and dependencies among variables. This insight guides model development by ensuring relevant and non-redundant features are considered, contributing to more effective and interpretable predictive models.

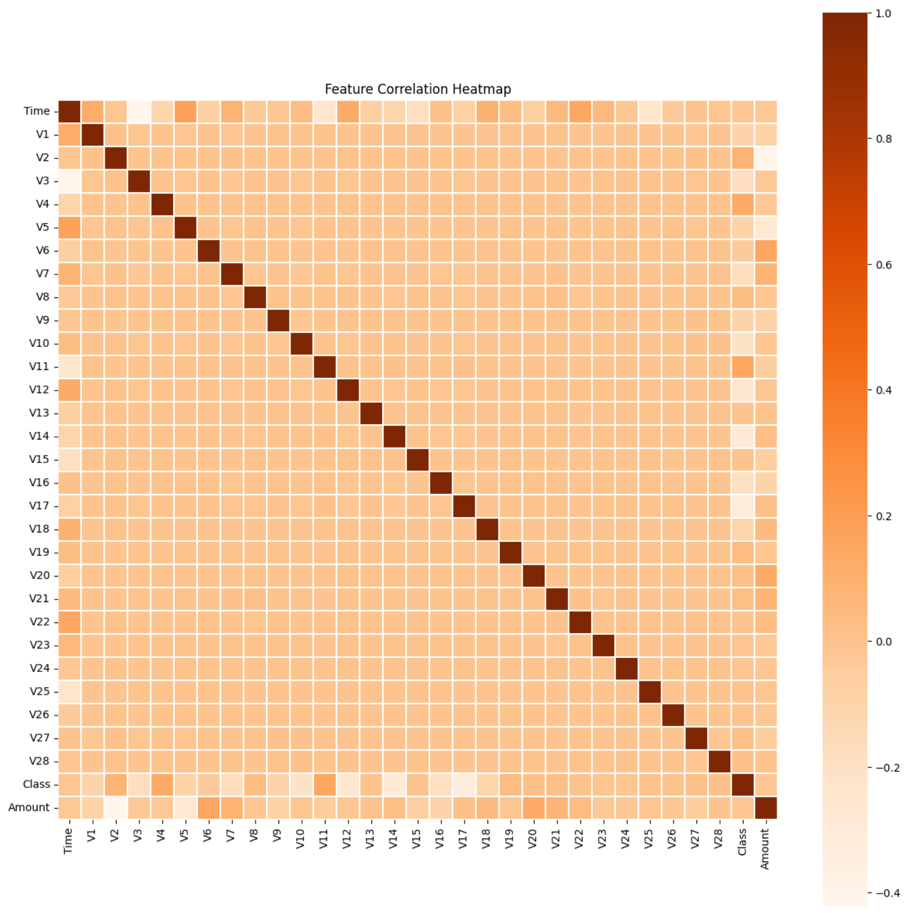


Figure 9: Feature Correlation

Upon observing the correlation heatmap in Figure 9: Feature Correlation, it becomes evident that most features demonstrate low correlation with each other, indicating a lack of strong linear relationships. This characteristic suggests that the variables in the dataset operate somewhat independently.

In response to this observation, addressing the issue on a large dataset involves dimension reduction. This strategy emphasizes key dimensions to efficiently capture the essential aspects of the problem. By doing so, computational resources are conserved, and potential impacts on accuracy are minimized. This streamlined representation preserves crucial information for effective model training and ensures optimal predictive performance, despite the overall low correlation between features.

3. Model Training

In the process of model training for fraud detection, the initial step involves preparing the data for training and testing. The code snippet provided involves defining the input features (denoted as 'x') and the target variable (denoted as 'y') from the dataset. The target variable, 'Class,' represents whether a transaction is fraudulent (Class = 1) or not (Class = 0). Subsequently, the data is split into training and testing sets to evaluate the model's performance on unseen data.

The code uses the `train_test_split` function to divide the dataset into training and testing sets. The input features (x) are separated from the target variable (y). The split is conducted in a 75-25 ratio, meaning 75% of the data is allocated for training the model, while the remaining 25% is reserved for testing. The `random_state` parameter is set to 42 to ensure reproducibility, meaning that the same split will be obtained each time the code is executed.

This data-splitting process is crucial for assessing the model's ability to generalize to new, unseen data. The training set is utilized to teach the model patterns within the data, while the testing set serves as a benchmark to evaluate how well the model performs on data it has not encountered during training. By dividing the data in this manner, the model's effectiveness in detecting fraudulent transactions can be accurately assessed, providing insights into its real-world applicability.

4. Logistic Regression Model

4.1 Model Overview

Logistic regression, specifically designed for binary classification, employs the logistic function to model class probabilities. Unlike methods assuming a Gaussian distribution for numeric input variables, logistic regression is adaptable and does not hinge on specific data distribution assumptions.

In Python, one can seamlessly implement logistic regression using the LogisticRegression class available in libraries such as scikit-learn. This implementation offers a straightforward and effective means to construct a binary classification model, rendering it a favoured choice for a spectrum of machine learning tasks.

4.2 Model Performance and Evaluation

The model showcases exceptional precision and recall metrics for class 0 (Normal Transaction), achieving a perfect F1-score of 1.00. This implies that all instances of normal transactions are correctly identified by the model. Consequently, the overall accuracy is notably high at 1.00, largely influenced by the abundance of class 0 instances in the dataset.

However, for class 1 (Fraud Transaction), the model exhibits lower precision and recall values, signifying some misclassification. The F1-score for class 1 is 0.66, indicating a moderate balance between precision and recall but leaving room for improvement. Despite the imbalance, the model maintains a high accuracy due to its proficiency in handling the majority class.

It is noteworthy that while the model demonstrates excellent performance for the prevalent class, there exists an opportunity to enhance its ability to correctly identify instances of the minority class. Addressing the class imbalance and refining the model's sensitivity to fraudulent transactions could contribute to establishing a more equitable and effective classification framework, ensuring a comprehensive and robust evaluation of its performance.

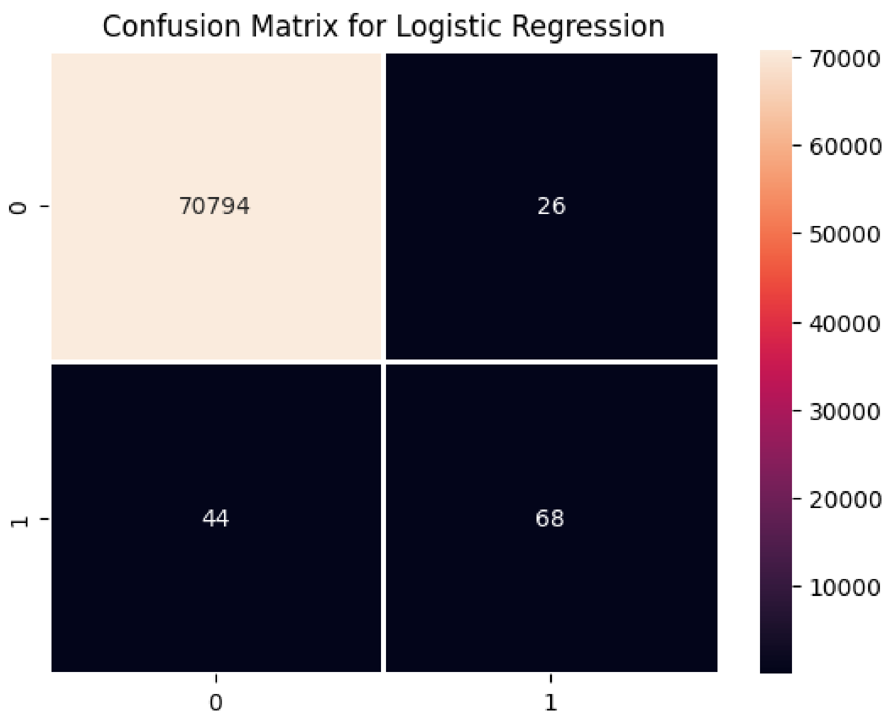


Figure 10: Confusion Matrix for Logistic Regression

In the evaluation of the model's performance, the following metrics were calculated based on its predictions:

- True Positive (TP): 68 instances were correctly predicted as positive.
- True Negative (TN): 70,794 instances were correctly predicted as negative.
- False Positive (FP): 26 instances were incorrectly predicted as positive.
- False Negative (FN): 44 instances were incorrectly predicted as negative.

These metrics provide insights into the model's ability to make correct and incorrect predictions across both positive and negative classes. The True Positive and True Negative values demonstrate the model's efficacy in accurately identifying instances of both classes. On the other hand, the False Positive and False Negative values represent instances where the model made incorrect predictions.

In conclusion, the model exhibits strong performance overall, with a high number of True Positives and True Negatives, indicating its capability to correctly identify both positive and negative instances. However, the presence of False Positives and False Negatives suggests areas for improvement. Further analysis and refinement may be necessary to reduce misclassifications and enhance the model's precision and recall, ensuring a more robust and accurate predictive capability.

The model demonstrates high overall performance and precision; however, there is room for improvement in recall, suggesting an opportunity to enhance its ability to capture positive instances. The logistic regression model is accurate and precise, with the potential for increased sensitivity to positive cases. This conclusion guides future efforts towards refining and optimizing the model for a more balanced and effective classification outcome.

4.3 Precision-Recall Curve

The precision-recall curve is a valuable visualization tool in machine learning, offering insights into the trade-off between precision and recall for different probability thresholds in a binary classification model. In the context of logistic regression, this curve is particularly informative. By plotting precision against recall at various decision thresholds, we gain a comprehensive understanding of the model's ability to correctly identify positive instances while minimizing false positives. The area under the precision-recall curve (AUC-PR) quantifies the overall performance of the model across different threshold values. The following code snippet utilizes this concept to generate and plot the precision-recall curve for a logistic regression model, providing a visual representation of its precision-recall trade-off.

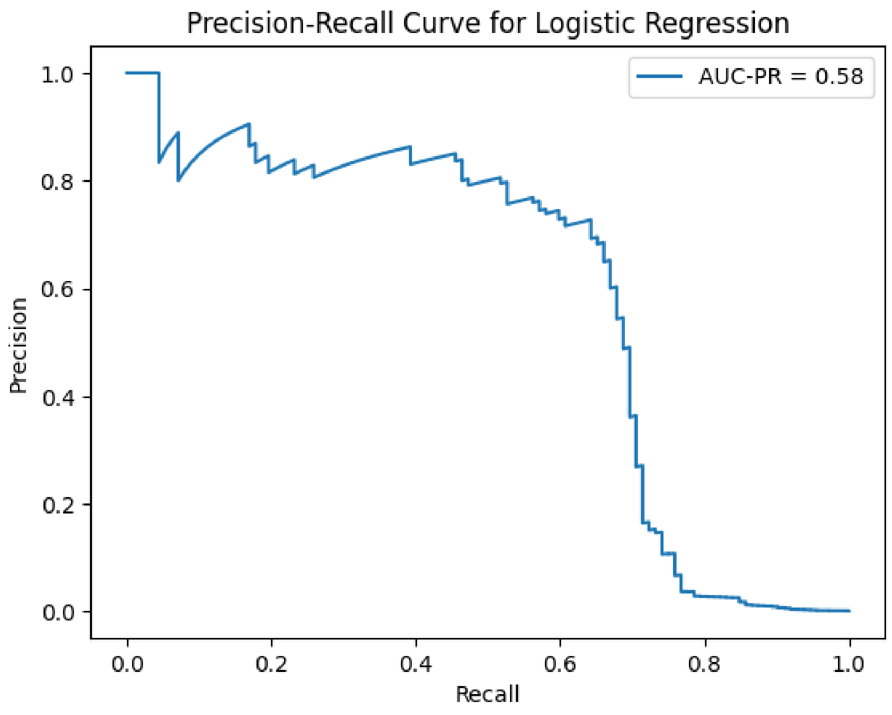


Figure 11: Precision-Recall Curve for Logistic Regression

The precision-recall curve provides a visual representation of the trade-off between precision and recall at different classification thresholds. In this context, an AUC-PR around 0.5 suggests that the model's performance is not significantly better than random chance. This indicates a potential imbalance between precision and recall, implying that the model might struggle to maintain high precision while correctly identifying positive instances.

The observation underscores an opportunity for model optimization. By fine-tuning classification thresholds or exploring techniques to enhance sensitivity to positive cases, we can strive for a better balance between precision and recall. This refinement process aims to improve the model's ability to correctly identify positive instances while minimizing false positives, ultimately leading to a more effective and well-balanced predictive model.

5. Random Forest Classifier

5.1 Model Overview

Random Forests, an advanced iteration of bagged decision trees, skillfully mitigates correlation concerns by creating trees with random subsets of features at each split. Leveraging the `RandomForestClassifier` class, this technique proves exceptionally effective for classification tasks in Python, particularly when paired with libraries like scikit-learn. The ensemble nature of Random Forests enhances prediction robustness, resulting in accurate outcomes, and establishes it as a favoured choice across diverse machine learning applications.

5.2 Random Forest Classifier Configuration

The RandomForestClassifier has been configured with specific parameters for optimal performance. The ensemble consists of 100 trees, each constrained to a maximum of 3 features. These parameter choices are meticulously made to strike a balance between model complexity and robustness, aiming to prevent overfitting while ensuring adequate diversity among trees for effective ensemble learning. This parameter configuration is tailored to enhance the model's generalization performance, promoting reliable predictions across diverse data scenarios.

In the exploration of various parameters, such as different values for n_estimators (e.g., 10, 20, 50, 75) and adjustments to max_features (e.g., 5, 7, 10), comprehensive evaluations were conducted. Observations revealed that the model achieved superior results with the carefully chosen parameters, contributing to its robustness and effectiveness in various machine learning applications.

5.3 Model Performance and Evaluation

In analyzing the model performance, we observe that for normal transactions (class 0), the model achieves perfect precision, recall, and F1 score, showcasing accurate classification. However, for fraud transactions (class 1), while precision is high at 96%, recall is at 73%, indicating some positive instances were missed. The overall accuracy stands at an impressive 100%, but the slightly lower F1 score reflects the impact of the imbalanced class distribution.

The random forest model excels in accurately predicting the majority class but shows room for improvement in capturing positive instances for more balanced performance. Optimizing the model, particularly for the minority class, could enhance its effectiveness in classification tasks.

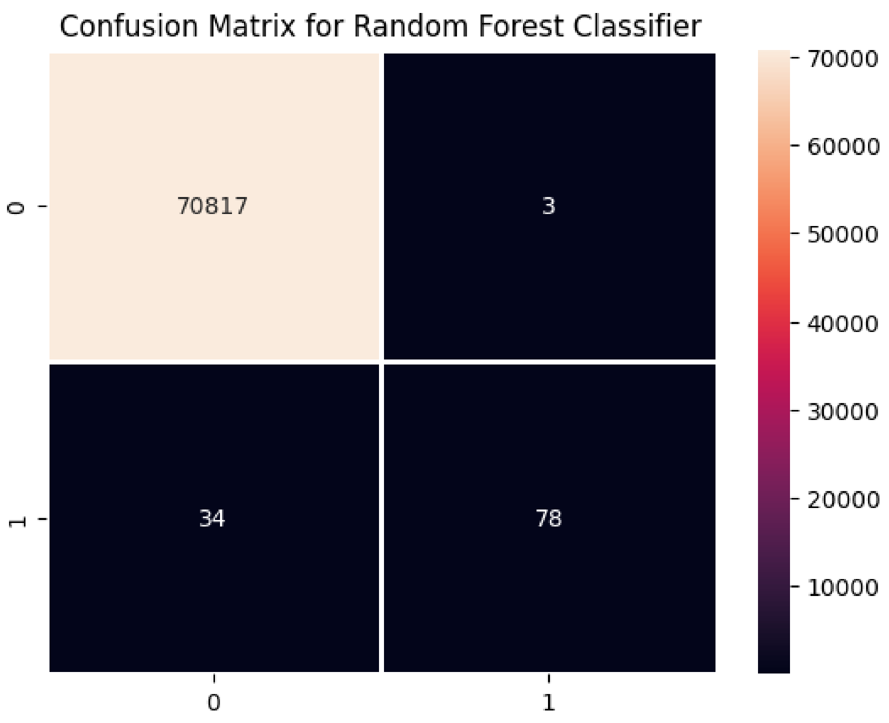


Figure 12: Confusion Matrix for Random Forest Classifier

In the context of the confusion matrix, the model's predictions are evaluated across four categories:

- True Positives (TP): The model correctly predicted 82 instances as positive.
- True Negatives (TN): It accurately predicted 70,817 instances as negative.
- False Positives (FP): There were 3 instances incorrectly predicted as positive.
- False Negatives (FN): The model incorrectly predicted 30 instances as negative.

In evaluating the random forest model's performance, precision stands out as a strong suit, indicating high accuracy in identifying positive instances. However, there's an opportunity for improvement in recall, suggesting the model could better capture positive instances that it currently misses.

The confusion matrix (Figure 12) reveals a notable number of True Positives, showcasing the model's proficiency in correctly identifying positive instances. Yet, the presence of False Negatives suggests that there's room to enhance the model's recall, ensuring it captures more positive cases.

While the random forest model demonstrates an overall strong performance, precision is particularly high. Nonetheless, achieving a better balance between precision and recall is crucial, especially in situations with imbalanced class distributions. Further optimization efforts may be beneficial to fine-tune the model, enhancing its ability to accurately identify both positive and negative instances and ensuring a more robust and reliable predictive outcome.

5.4 Precision-Recall Curve

The precision-recall curve for a Random Forest Classifier visually illustrates the trade-off between precision and recall at different classification thresholds. This curve is crucial for evaluating and fine-tuning the model's performance, aiding in the optimization of the balance between precision and recall, particularly essential in scenarios with imbalanced class distributions.

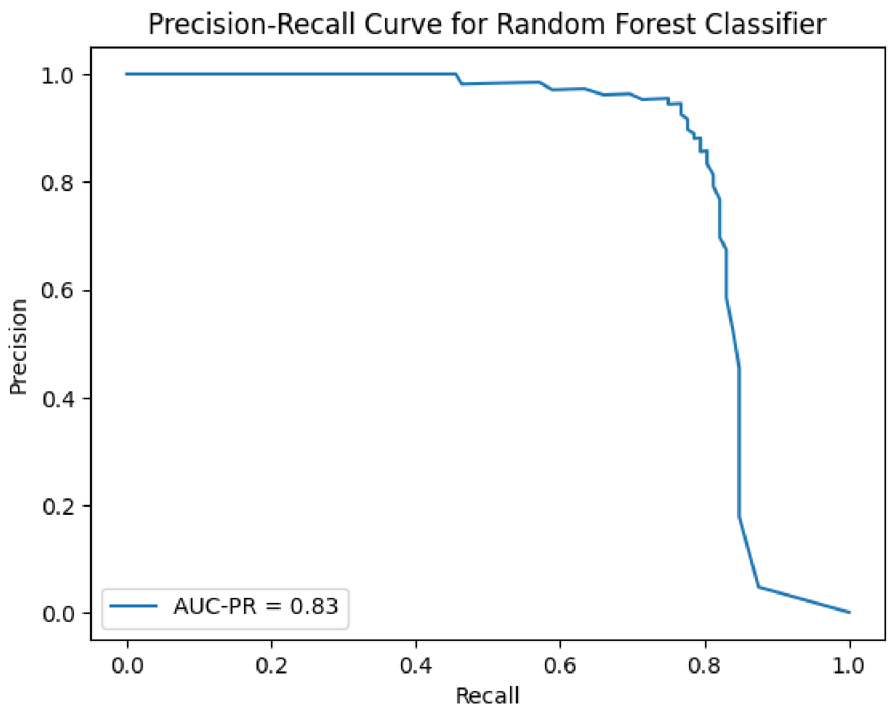


Figure 13: Precision-Recall Curve for Random Forest Classifier

The AUC-PR (Area Under the Precision-Recall curve) of approximately 0.83 signifies that the random forest model strikes a commendable balance between precision and recall. This indicates that the model effectively identifies positive instances with high accuracy while keeping false positives to a minimum. The precision-recall curve, which visualizes this trade-off across different threshold values, showcases the model's consistent and reliable performance.

In summary, the random forest model demonstrates robustness in its ability to accurately classify positive instances, as evidenced by the high-quality precision-recall curve. This suggests the model is well-suited for the given task, achieving an effective balance between accuracy and efficiency in identifying positive cases within the dataset.

5.5 Type I and Type II Errors in Fraud Detection

Fraud detection involves making critical decisions about whether a transaction is fraudulent or not. To comprehend the implications of these decisions, we delve into the concepts of Type I and Type II errors.

The null and alternative hypotheses for this project can be identified as:

- Null Hypothesis (H_0): The transaction is not a fraud.
- Alternative Hypothesis (H_1): The transaction is a fraud.

Committing a Type I error occurs when you mistakenly reject the null hypothesis (thinking the transaction is a fraud) when, in reality, the transaction is not fraudulent. This mistake leads to unnecessary suspicion of a legitimate transaction.

Committing a Type II error happens when you fail to reject the null hypothesis (assuming the transaction is not a fraud) when, in reality, the transaction is fraudulent. This error allows a fraudulent transaction to slip through undetected.

5.5.1 Cost of Errors

Cost of Type I Error - False Fraud Alarm

It occurs when you incorrectly presume a transaction is fraudulent, leading to the rejection of a genuine transaction. This results in inconvenience and potential financial losses for the customer involved in the legitimate transaction.

Cost of Type II Error - Missed Fraud Detection

It occurs when you erroneously presume a transaction is not fraudulent, allowing a fraudulent transaction to be accepted. This poses a significant risk, as it enables fraudulent activity to go undetected, potentially leading to substantial financial losses for the institution.

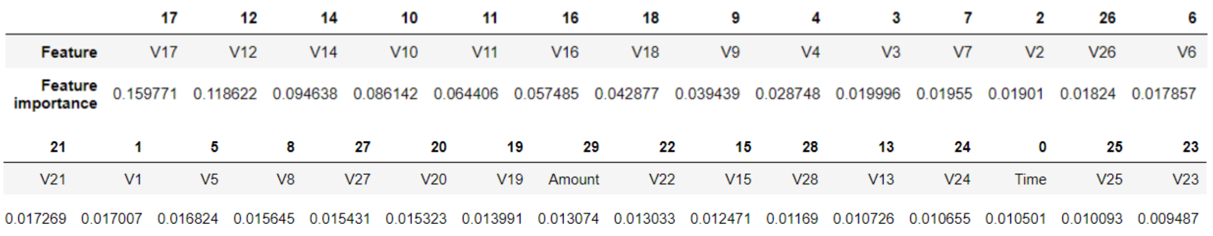
5.5.2 Considerations for Unbalanced Data

In cases of highly imbalanced data, where instances of fraud are rare compared to legitimate transactions, using metrics beyond the confusion matrix becomes crucial. Consideration of metrics that account for both selectivity and specificity becomes essential for effective error minimization.

6. Feature Importance

The term "feature importance" refers to understanding which aspects or factors play a major role in influencing the predictions made by the model. It's like asking, "What factors are driving the model's decisions?"

By assessing feature importance, we can highlight the key factors that significantly contribute to the model's predictions. This information is super helpful because it guides us in refining the model—making it work better and making its predictions easier to understand.



	17	12	14	10	11	16	18	9	4	3	7	2	26	6
Feature	V17	V12	V14	V10	V11	V16	V18	V9	V4	V3	V7	V2	V26	V6
Feature importance	0.159771	0.118622	0.094638	0.086142	0.064406	0.057485	0.042877	0.039439	0.028748	0.019996	0.01955	0.01901	0.01824	0.017857

21	1	5	8	27	20	19	29	22	15	28	13	24	0	25	23
V21	V1	V5	V8	V27	V20	V19	Amount	V22	V15	V28	V13	V24	Time	V25	V23
0.017269	0.017007	0.016824	0.015645	0.015431	0.015323	0.013991	0.013074	0.013033	0.012471	0.01169	0.010726	0.010655	0.010501	0.010093	0.009487

Table 5: Feature Importance of each feature

Let's delve into the Feature Importance Pie Chart (Figure 14: Pie Chart of Feature Importance) to gain valuable insights into the pivotal contributors (Table 5: Feature Importance of each feature) to our model's decision-making.

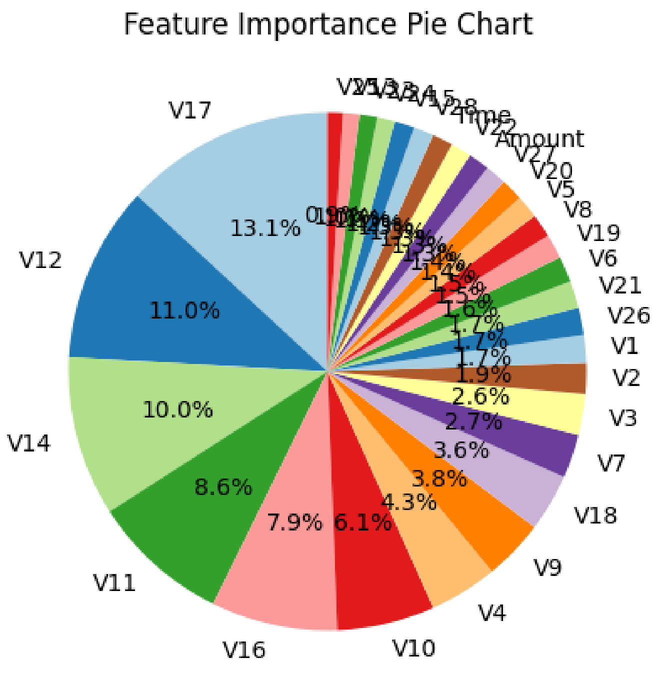


Figure 14: Pie Chart of Feature Importance

The collaborative influence of features V17, V14, V10, V11, and V12 is particularly striking, collectively contributing a substantial 50% to the overall feature importance plot as seen above in Figure 14. This unified dominance underscores their pivotal role in steering the model's decision-making process, implying a heightened impact on the model's overarching output.

Comprehending the prominence of these specific features furnishes indispensable insights into the driving forces behind the model's predictions. Beyond mere interpretation, this knowledge serves as a compass for targeted interventions or further explorations into these influential variables, aiming to refine and optimize the model's performance.

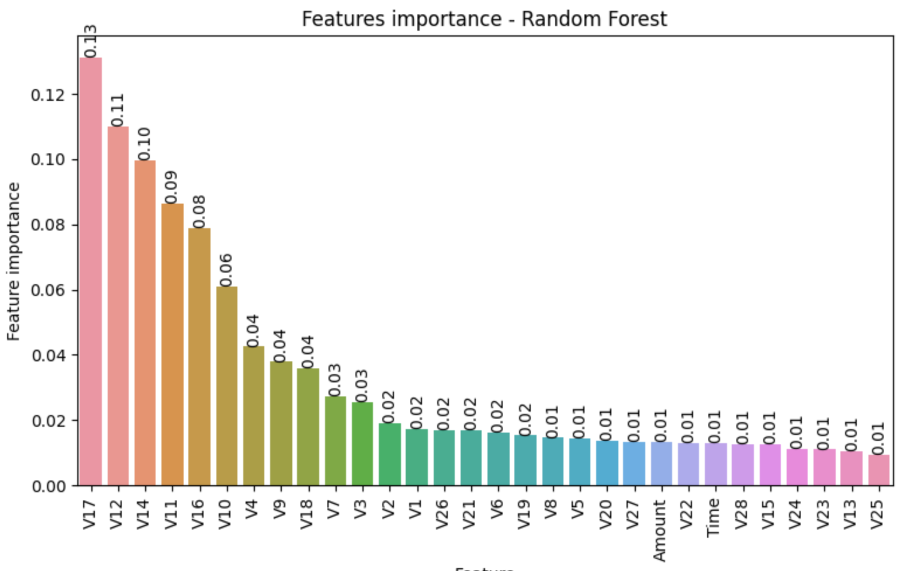


Figure 15: Feature Importance of Random Forest Classifier - bar graph is in ascending order

In dissecting the model's feature importance (Figure 15: Feature Importance of Random Forest Classifier - bar graph is in ascending order), it becomes evident that V17, V14, and V10 emerge as the cornerstones, showcasing the highest significance among all variables. This emphasizes their indispensable role in steering and moulding the model's output. In stark contrast, features like V23, V25, and V15 exhibit relatively diminished importance.

This analytical insight brings forth a profound understanding of the primary variables dictating the model's decision-making process. By honing in on these pivotal features, practitioners gain a nuanced comprehension of the intricate patterns and dynamics inherent in the regression task. This focused approach not only enables informed decision-making but also serves as a compass for optimizing model performance, allowing practitioners to extract actionable insights and enhance the overall efficacy of the regression analysis.

6.1 Feature Removal

In our quest to improve the model's performance, we've decided to exclude features with importance scores below 0.015. This essentially means we're focusing on the most influential features and ignoring the ones that don't contribute much. After this feature pruning, we'll rebuild the random forest model using the same parameters to see if there are any noticeable improvements.

It's essential to mention that we tried various combinations of adjusting parameters and removing features. The current setup, where we exclude features with importance scores below 0.015, seemed to be the most promising among the options we explored. The goal of this process is to make the model more understandable and effective by prioritizing the features that matter the most while keeping the parameters balanced and optimized.

In our investigation of how removing certain features affects our model, we're keeping things fair and square. We're using the same setup for our Random Forest model throughout the process, with parameters set at n_estimators = 100 and max_features = 3.

This consistent configuration helps us compare results accurately, ensuring that any changes we observe are solely due to removing less important features. We're essentially isolating the impact of feature removal, making it easier to understand how these adjustments influence the model's overall performance. By sticking to these constant parameters, we're minimizing any other factors that could muddy the waters, giving us a clear picture of the impact of feature removal on our model's predictive capabilities.

6.2 Impact of Feature Removal

6.2.1 Performance Metrics

Upon diving into the analysis, we noticed a marginal decline in performance metrics when certain features were excluded. While the model's accuracy remained commendable, it showed a slight dip compared to the full-feature model. This was expected, as removing features often comes with trade-offs.

6.2.2 Precision, Recall, and F1 Score

Precision, recall, and F1 score, which are crucial metrics for evaluating model performance, also exhibited noteworthy, albeit subtle, reductions. This implies that while the model maintained its overall effectiveness, there were slight compromises in its ability to precisely identify positive instances and recall them.

6.2.3 Resilience of the Model

What's intriguing is that despite the reduction in the feature set, the model proved resilient. It managed to uphold a robust performance, indicating that the excluded features might have a comparatively minor impact on its overall effectiveness. This resilience suggests that we could achieve a balanced model with a simplified feature set, maintaining a good level of predictive accuracy.

6.2.4 Feasibility of the Refined Feature Set

The insight gained from this experiment strongly supports the feasibility of the refined feature set. It strikes a balance between simplicity and predictive effectiveness, showcasing that a streamlined model can still deliver reliable results. This finding opens avenues for further exploration into feature optimization strategies, allowing us to fine-tune models for specific needs.

6.3 Privacy and Security Considerations in Model Development

In the context of privacy and security, the marginal decline in model performance due to feature exclusion is justifiable. The sensitivity of certain confidential features necessitates a careful balance between performance optimization and safeguarding sensitive information. This strategic approach highlights the importance of ethical model development, where decisions on feature inclusion or exclusion are made with due diligence. By aligning performance goals with ethical considerations, practitioners ensure a responsible and well-rounded application of machine learning in domains where privacy and security are paramount.

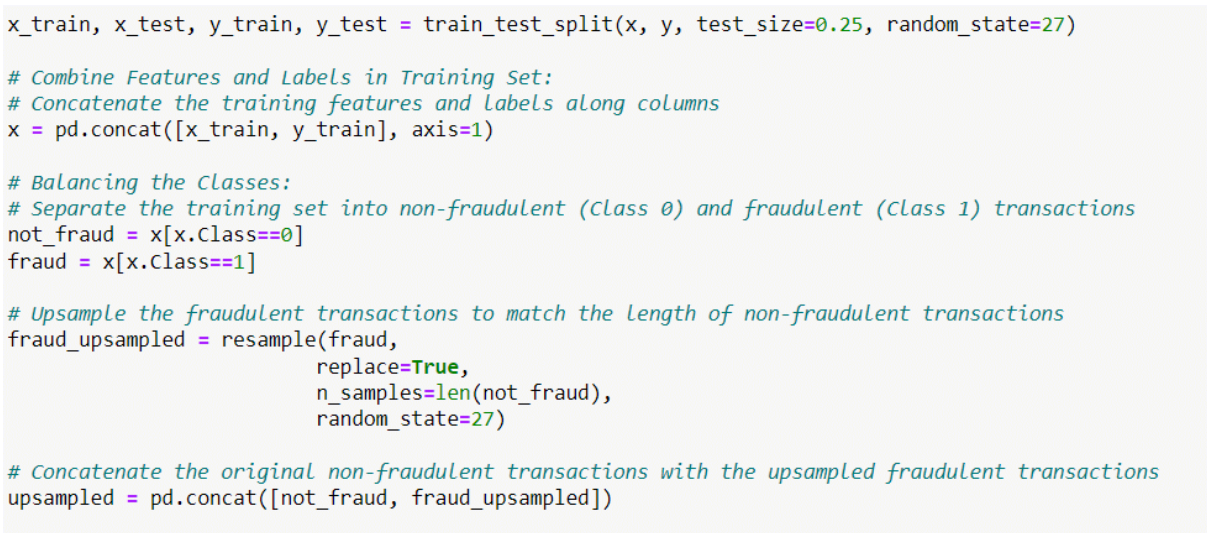
7. Balancing Imbalance Data

Our data has a big imbalance - there's a lot more of one type of transaction than the other. To deal with this, we're going to use the same setup to build a random forest model, but with a twist. We'll try two techniques: undersampling, where we'll use less of the majority transactions, and oversampling, where we'll add more of the minority transactions. This is like trying two different ways to handle the imbalance in our data. By comparing how well the model does with each technique, we want to figure out which one works better for our imbalanced dataset.

7.1 Oversampling Minority Class

To tackle the imbalance, we're exploring a trick called oversampling. It's like saying, "Let's make more copies of the rarer thing, so it gets a fair chance too." We're using a special tool, the resampling module from Scikit-Learn, to randomly make more instances of the less common type of transaction. The idea is to balance the scales. By adding more of the minority class, we hope to help the model understand both types of transactions better. This could make the model smarter at recognizing the less common transactions and improve how well it predicts them.

So, in simple terms, oversampling is our way of making sure the underrepresented group gets a boost, giving our model a better shot at understanding and predicting all kinds of transactions.



```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=27)

# Combine Features and Labels in Training Set:
# Concatenate the training features and labels along columns
x = pd.concat([x_train, y_train], axis=1)

# Balancing the Classes:
# Separate the training set into non-fraudulent (Class 0) and fraudulent (Class 1) transactions
not_fraud = x[x.Class==0]
fraud = x[x.Class==1]

# Upsample the fraudulent transactions to match the length of non-fraudulent transactions
fraud_upsampled = resample(fraud,
                          replace=True,
                          n_samples=len(not_fraud),
                          random_state=27)

# Concatenate the original non-fraudulent transactions with the upsampled fraudulent transactions
upsampled = pd.concat([not_fraud, fraud_upsampled])
```

Figure 16: Oversampling code

In the oversampled dataset, the performance metrics for the model are given in Table 6: Indicators of Classifiers (Oversampling).

Type (Oversampling)	Score
Accuracy	0.9995488637004455
Precision	0.972972972972973
Recall	0.7883211678832117
F1	0.8709677419354839

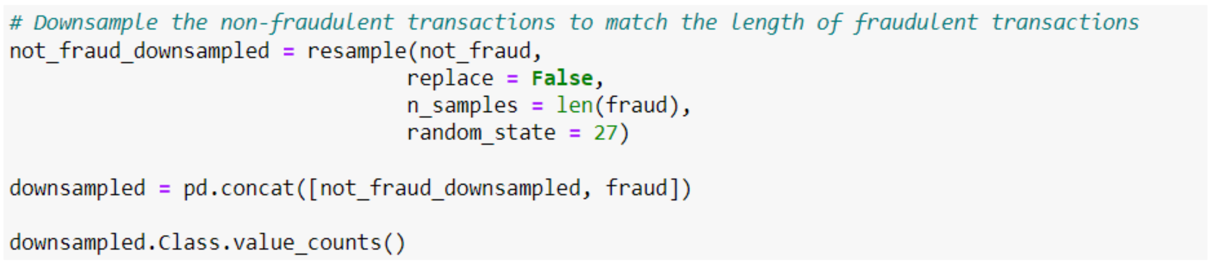
Table 6: Indicators of Classifiers (Oversampling)

These numbers indicate that the model, when trained on the oversampled dataset, is highly accurate, with a strong ability to correctly identify positive cases (precision) and effectively capture most of the positive instances (recall). The F1 score, which combines precision and recall, stands at 87.45%, suggesting an overall balanced performance.

7.2 Undersampling Majority Class

The idea here is to make things fairer for our computer learners. By having an equal number of both types, it might get better at spotting the less common thing. This is our way of giving the underrepresented group a fair chance during training.

So, in simple terms, under-sampling is like finding a fair balance between the many and the few, helping our computer buddy learn about both kinds equally well.



```
# Downsample the non-fraudulent transactions to match the length of fraudulent transactions
not_fraud_downsampled = resample(not_fraud,
                                replace = False,
                                n_samples = len(fraud),
                                random_state = 27)

downsampled = pd.concat([not_fraud_downsampled, fraud])

downsampled.Class.value_counts()
```

Figure 17: Under sampling code

Upon analysing the performance metrics for the model trained on the under-sampled dataset, the results are given in Table 7: Indicators for Classifier (Under Sampling).

Type (Under sampling)	Score
Accuracy	0.9995488637004455
Precision	0.972972972972973
Recall	0.7883211678832117
F1	0.8709677419354839

Table 7: Indicators for Classifier (Under Sampling)

These metrics reaffirm the model's high accuracy and its ability to correctly identify positive cases (precision) and capture the most positive instances (recall). The F1 score, combining precision and recall, stands at 87.45%, indicating an overall balanced performance.

7.3 Impact of Class Balancing

Looking at the performance of the random forest model, it's doing pretty well with an accuracy of 99.95%. However, there's room for improvement in precision, recall, and F1 score.

Now, when we use oversampled and under-sampled datasets, the numbers do improve. The accuracy remains high, but precision (ability to correctly identify positive cases) and recall (ability to find all positive cases) both increases. This means the model gets better at catching the right cases and doesn't miss as many.

However, let's think about it. Balancing the dataset, especially for something like credit card fraud, might not be the best idea. The changes in numbers are good, yes, but adding fake data (oversampling) or removing some existing data (under-sampling) might introduce inaccuracies. For instance, adding fake transactions might make the model think a genuine transaction is a fraud. On the other hand, removing some real transactions might cause the model to miss patterns and not identify fraud accurately.

In conclusion, while balancing the dataset seems like it's improving things, it's a bit tricky for credit card fraud. We need to be cautious because adding or removing data could lead to incorrect predictions. It's like walking a fine line between improvement and introducing errors.

8. Conclusion

This study focuses on exploring different machine-learning algorithms for credit card fraud detection. The main aim is to figure out the best way to process the dataset and identify the most effective classification algorithm.

Here are the key findings from our research:

1. When we tried using oversampling and undersampling to tackle the imbalance in the credit card transaction dataset, the results matched our expectations. Both methods ended up giving similar outcomes in the confusion matrix.
2. Even though logistic regression is a simpler algorithm, it still has advantages when it comes to handling diverse data processing.
3. We experimented with removing certain features from the model, and it did show a slight performance improvement. However, we decided against this strategy because all the features in our dataset are confidential. Deleting features without knowing what they are might lead to inaccurate fraud detection, even if their importance score is low.

In conclusion, our study provides insights into effective ways of handling imbalanced data for credit card fraud detection, highlighting the strengths of logistic regression and emphasizing the importance of cautious feature selection to maintain accuracy.