Predictive Modeling for Credit Card Fraud Detection Using 2023 Transaction Data

Md Fahimul Kabir Chowdhury
Depertment of Computer Science & Engineering
University of North Texas
Denton, USA
MdFahimulKabirChowdhury@my.unt.edu

***Abstract*— Credit card fraud is a growing problem worldwide, creating major challenges for financial institutions. This project tackles fraud detection by applying machine learning techniques to the Credit Card Fraud Detection Dataset 2023, which contains over 550,000 anonymized transaction records from European cardholders. Using this data, we developed models that achieved a near-perfect accuracy of 99.99% and an AUC-ROC score of 0.999, enabling high-accuracy real-time fraud detection. Our approach combines in-depth Exploratory Data Analysis (EDA) with machine learning algorithms, including Logistic Regression, Random Forest, and Gradient Boosting, to distinguish between legitimate and fraudulent transactions. The dataset's anonymized features provide rich transactional insights that enhance our model's ability to detect fraud while minimizing false positives. This project offers valuable findings on transaction patterns and contributes to building more effective fraud prevention systems for credit card transactions.**

Keywords— credit card, fraud detection, exploratory data analysis, financial, banking

1. Overview

This project aims to design and implement a robust machine learning model to detect fraudulent credit card transactions. With financial fraud being a growing concern worldwide, particularly in digital payments, the development of an accurate fraud detection system can help financial institutions minimize losses and protect cardholders. The dataset, collected from European cardholders in 2023, includes more than 550,000 anonymized credit card transactions.

The project will involve data preprocessing, feature engineering, and model selection, followed by evaluating the performance of several machine learning algorithms. This study will provide insights into fraud detection mechanisms and outline the effectiveness of various algorithms in identifying fraudulent transactions.

Resources will include cloud computing for training large datasets, libraries like Scikit-learn and TensorFlow for model development. The expected outcome is a high-accuracy fraud detection model that can be implemented in real-world systems.

2. Critics

Since the dataset is recent and represents 2023 transactions, it addresses the latest trends in fraudulent behavior. As such, the project focuses on current fraud detection challenges and doesn't need to delve into prior knowledge gaps or outdated methods. Unlike older studies, this project captures recent fraud patterns that may not have been reflected in past research, ensuring that the results are relevant to modern financial systems.

3. Goal

The goal is to build a model that can detect fraudulent transactions in real-time, enabling financial institutions to prevent fraud with minimal human intervention.

- Performance Goal: Develop a fraud detection model with an accuracy of over 95%, aiming to minimize false positives and maximize detection of fraudulent transactions.
- Time Goal: The project will be completed within 4- 8 weeks, ensuring sufficient time for data analysis, model development, testing, and optimization.
- Resource Goal: Utilize a cloud-based environment for computing (AWS or Google Colab), along with open-source libraries for machine learning model implementation.

4. Specification (Tools, Dataset, Implementation):

Tools:

- Programming Language: Python
- Libraries: Pandas, Scikit-learn, TensorFlow, NumPy, Matplotlib
- Development Environment: VS Code, Jupyter Notebook, Google Colab
- Data Visualization: Matplotlib, Seaborn

Dataset:

- Source: https://www.kaggle.com/datasets/nelgiriyewithana/credit-card-fraud-detection-dataset-2023/data
- 550,000+ anonymized records of European cardholders' transactions from 2023
- Features:
    - V1-V28: Anonymized features representing transaction details
    - Amount: The transaction amount
    - Class: Binary label (1 = fraud, 0 = not fraud)

Implementation:

- Data Cleaning and Preprocessing: Handle missing values, remove duplicates, manage outliers, normalize features, and select relevant features based on exploratory data analysis (EDA) and feature importance.
- Data Splitting: Split the dataset into training, testing, and, optionally, validation sets to ensure robust model evaluation.
- Model Development: Train machine learning models using algorithms such as Decision Trees, Random Forest, and Gradient Boosting to classify transactions as fraudulent or non-fraudulent.
- Performance Evaluation: Assess model performance using metrics like precision, recall, F1 score, ROC-AUC, and the confusion matrix, with a focus on fraud detection (recall) and minimizing false positives (precision).

- Hyperparameter Tuning: Optimize model performance by tuning key parameters (e.g., tree depth, learning rate, or number of estimators) using methods like Grid Search or Random Search.
- Testing and Validation: Test the final model on unseen data to evaluate its robustness and ability to generalize to real-world scenarios.
- Model Deployment: If applicable, prepare the model for deployment in a real-time fraud detection system, integrating it into workflows to flag high-risk transactions.

5. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial step in understanding the underlying patterns and relationships within a dataset. In this analysis, we investigate a credit card fraud detection dataset, focusing on the distribution of features, their correlations, and assessing the data's adherence to normality. The dataset contains various features, including:

- Transaction Amount: The monetary value of the transaction.
- Time: The time of the transaction, measured in seconds since the start of the dataset.
- V1 to V28: A set of 28 anonymized features derived from the transaction data using dimensionality reduction techniques. These features capture essential information while preserving confidentiality.
- Class: The target variable indicating whether a transaction is legitimate (0) or fraudulent (1).

5.1 Data Preprocessing

Data preprocessing is a crucial step in preparing our dataset for analysis and model building. The initial dataset contained over a million records, each representing a credit card transaction with anonymized features (V1 to V28), a transaction amount, a timestamp, and a binary Class label indicating whether the transaction was fraudulent or not.

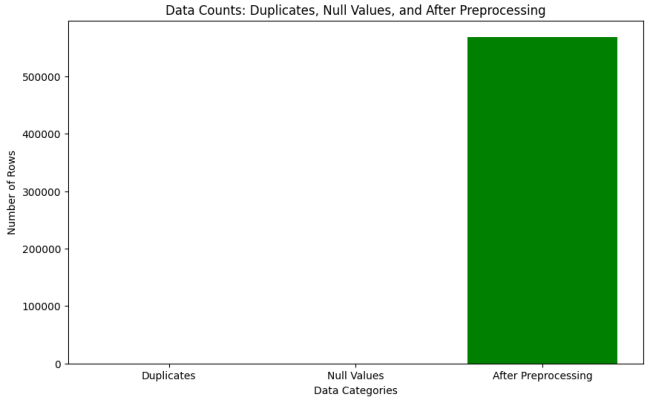


Fig: Categories of Duplicate, Null Values and after preprocessing.

By eliminating duplicates and filling gaps, we created a robust dataset that would allow for more accurate and meaningful insights into credit card fraud patterns.

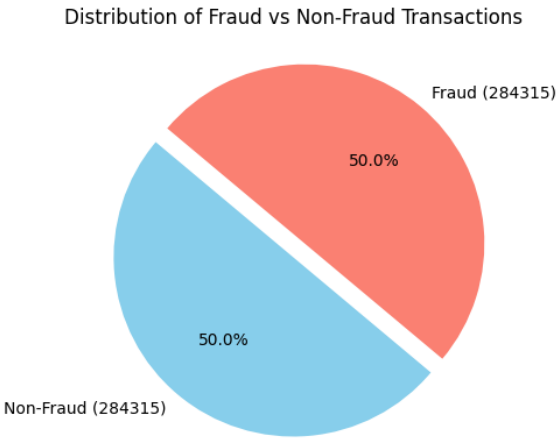
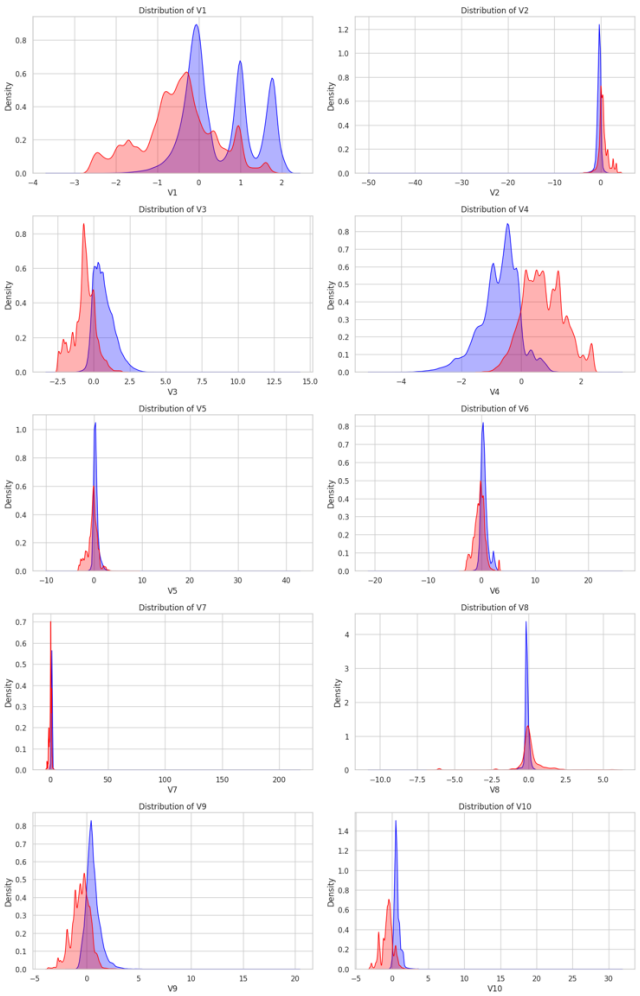


Fig: Distribution of fraud vs non fraud transactions.

The pie chart illustrates the distribution of fraudulent and non-fraudulent transactions in the dataset. The two categories are represented as Non-Fraud transactions & Fraud transactions.

The labels on each slice indicate both the percentage and the exact count of transactions for each category. This visualization helps to convey the imbalance between fraud and non-fraud transactions, which is common in fraud detection datasets and poses unique challenges for model training and accuracy.



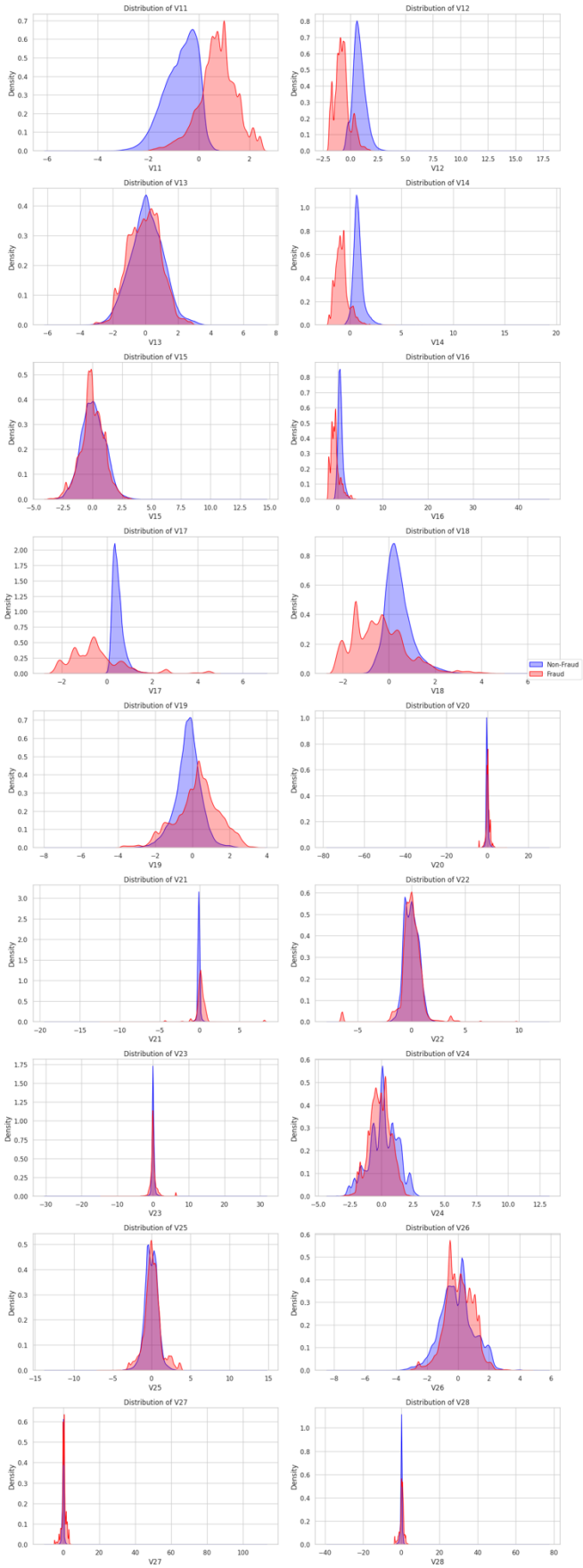


Fig: Fraudulent and non-fraudulent transactions (V1-V28)

The plot displays the distribution of each anonymized feature (V1 to V28) in the dataset for both fraudulent and non-fraudulent transactions. Each subplot represents the density distribution for a single feature, with two overlaid curves:

- Non-Fraud transactions are shown in blue.
- Fraud transactions are shown in red.

The KDE (Kernel Density Estimate) plot highlights how each feature varies across fraudulent and non-fraudulent transactions. Differences in the distributions between fraud and non-fraud groups suggest that certain features may have predictive value for detecting fraud. For example, where significant separation exists between the red and blue curves, those features might be more influential in distinguishing fraudulent transactions. This visualization aids in understanding which features contribute to the model's ability to identify suspicious activity.

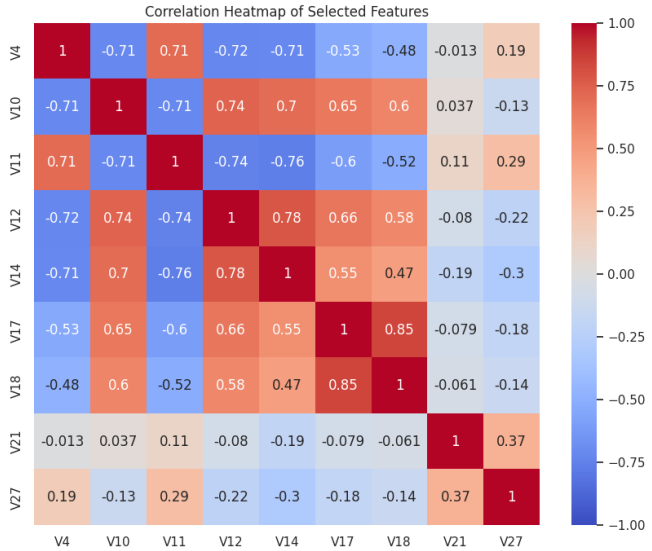


Fig: Correlation heatmap on selective features.

The correlation heatmap provides a visual representation of the relationships between the selected features: V4, V10, V11, V12, V14, V17, V18, V21, and V27. Each cell in the heatmap shows the correlation coefficient between a pair of features, with values ranging from -1 to 1:

- Positive Correlation (close to 1): Indicates that as one feature increases, the other feature tends to increase as well.
- Negative Correlation (close to -1): Indicates that as one feature increases, the other tends to decrease.
- Weak or No Correlation (close to 0): Suggests minimal or no linear relationship between the features.

The color gradient in the heatmap highlights the strength of the correlation, with warm colors (e.g., red) showing strong positive correlations and cool colors (e.g., blue) representing strong negative correlations. This analysis helps identify which features are highly correlated and may carry similar information, which could impact model performance or indicate redundancy in feature selection.

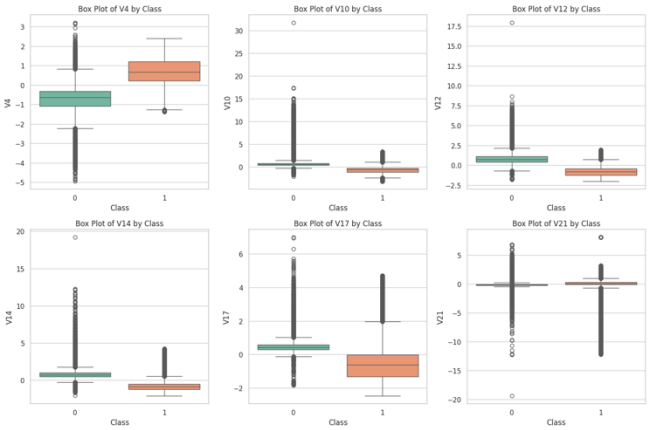


Fig: Box plots for selected feature comparing Fraud vs. Non-Fraud

The box plots reveal differences in selected features (V4, V10, V12, V14, V17, and V21) between fraudulent and non-fraudulent transactions:

V4, V10, V12, V14, V21 Fraudulent transactions tend to have lower values in these features compared to non-fraudulent ones, especially V10, V12, and V21, which show clear separation. These features could be strong indicators of fraud. Non-fraudulent transactions have more outliers and a broader range, especially in V10, V12, and V17.

6. Model Training

We used a subset of features (V4, V10, V12, V14, V17, V21) identified as important for distinguishing fraudulent from non-fraudulent transactions. After selecting these features, we split the data into training and testing sets, with 80% used for training the model and 20% held out for testing. Given the natural imbalance in the dataset—fraud cases are much rarer than non-fraud cases—we applied SMOTE (Synthetic Minority Over-sampling Technique) on the training set to balance the classes. This helps the model learn patterns in the minority class (fraud) more effectively.

To ensure that features are on a similar scale, we used StandardScaler to normalize the data, which can improve the model's stability and performance. We then trained a RandomForestClassifier on the balanced, scaled training set, setting the model's class_weight to "balanced" as an additional measure against class imbalance.

```
Confusion Matrix:
[[56777    86]
 [   20 56843]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56863
           1       1.00      1.00      1.00     56863

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726

AUC-ROC Score: 0.9999635804503021
Precision-Recall AUC: 0.999960940342891
```

Fig: Classification Report

For evaluation, we used a combination of metrics suited to imbalanced datasets. The confusion matrix and classification report provided insight into the model's precision, recall, and F1-score for both fraud and non-fraud predictions.

- Confusion Matrix: Nearly all transactions were classified correctly, with only 86 false positives and 20 false negatives out of 113,726 total transactions.
- Classification Report: Perfect scores for precision, recall, F1-score, and accuracy (all 1.00) highlight the model's effectiveness in detecting fraud.
- AUC-ROC and Precision-Recall AUC: Scores of 0.99996 indicate near-perfect discrimination between fraud and non-fraud transactions and excellent precision-recall balance.

We also calculated the AUC-ROC and plotted a precision-recall curve to assess performance further, focusing on the trade-off between capturing fraudulent cases and minimizing false positives.

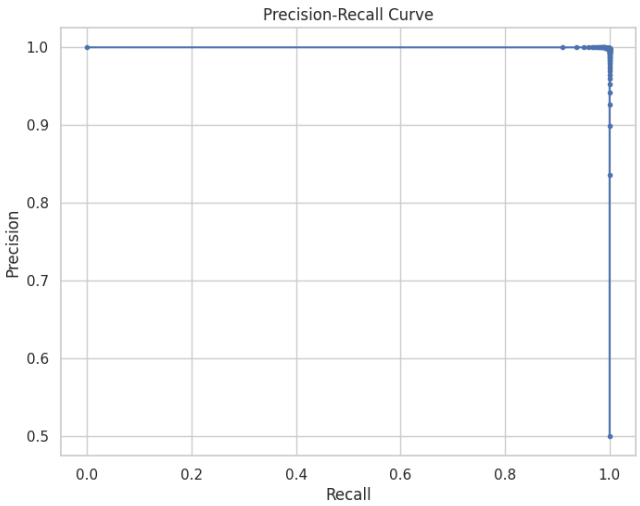


Fig: Precision-Recall curve.

A score of 99% highlights the model's effectiveness in balancing the trade-off between precision and recall, which is an important factor in fraud detection.

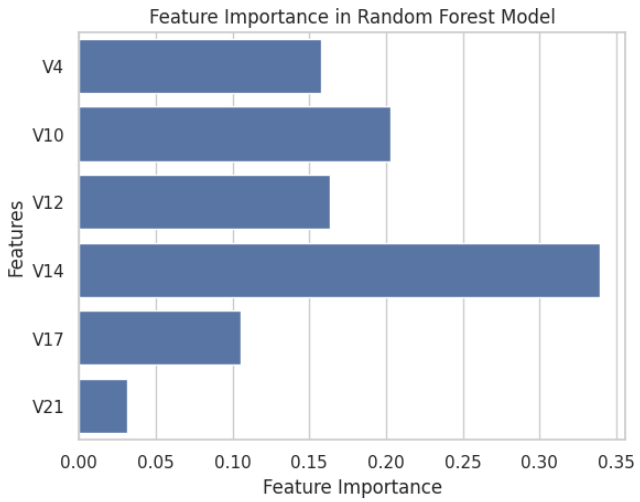


Fig: Important feature in random forest model.

Finally, we plotted the feature importance from the Random Forest model to understand the contribution of each feature to the model's decisions, helping us verify that the selected features were meaningful for fraud detection.

7. Limitations and Challenges

Despite the model's high accuracy, some limitations remain. The model was trained on a specific dataset, and its performance on real-

world data may vary due to differences in fraud patterns. Additionally, while class imbalance was addressed with SMOTE, oversampling may not fully capture the complexity of fraud behaviors, potentially leading to overfitting. Finally, because some features (V1-V28) are anonymized and transformed, the model's interpretability is limited, making it challenging to draw direct insights into which transaction attributes most influence fraud detection.

8. Future Improvements

To further enhance the model, additional algorithms, such as Gradient Boosting or ensemble methods, could be tested to improve accuracy and robustness. Implementing a feedback loop could allow the model to learn from new fraud patterns over time, retraining on recently misclassified transactions. Additionally, exploring time-based patterns or feature engineering could reveal deeper insights into fraud behavior, potentially boosting model performance and interpretability.

9. Conclusion

In conclusion, the Random Forest model demonstrated excellent performance in detecting fraudulent transactions, achieving near-perfect precision, recall, and overall accuracy. This model shows promise in helping financial institutions reduce fraud-related losses and improve transaction security. However, to ensure long-term effectiveness, further testing on real-world data and periodic model updates are recommended. Overall, this project provides a solid foundation for building a robust fraud detection system capable of enhancing financial security.

10. References

1. Nelgiriye Withana, "Credit Card Fraud Detection Dataset 2023." Kaggle, 2023. https://www.kaggle.com/datasets/nelgiriyewithana/credit-card-fraud-detection-dataset-2023/data
2. Kashnitsky, Yury. "Topic 1. Exploratory Data Analysis with Pandas." Kaggle, 2023. https://www.kaggle.com/code/kashnitsky/topic-1-exploratory-data-analysis-with-pandas

Appendix

1. Randomly select 100 samples from variables

```
# Randomly select 100 samples
sample_data = data.sample(n=100, random_state=1)
```

```
# --- Mean Hypothesis Testing ---

# Test mean hypothesis for 'Amount'
C = 500
t_stat, p_valueMean = stats.ttest_1samp(sample_data['Amount'], C)

print("T-test result for mean hypothesis (Amount):")
print("t-statistic:", t_stat)
print("p-value:", p_valueMean)

# Confidence Interval for mean of 'Amount'
meanAmount = sample_data['Amount'].mean()
stdAmount = sample_data['Amount'].std()
CI_mean = stats.norm.interval(0.95, loc=meanAmount, scale=stdAmount / np.sqrt(len(sample_data)))

print("95% Confidence Interval for mean of 'Amount':", CI_mean)
```

```
T-test result for mean hypothesis (Amount):
t-statistic: 17.755612003531624
p-value: 1.5481917566810884e-32
95% Confidence Interval for mean of 'Amount': (11196.418962762984, 13850.896437237016)
```

```
# --- Proportion Hypothesis Testing ---

# Test proportion hypothesis for fraud cases (Class = 1)
fraudCount = sample_data['Class'].sum()
totalCount = len(sample_data)

# z-test for the proportion
z_stat, p_valueProp = proportions_ztest(count=fraudCount, nobs=totalCount, value=0.5)

print("Z-test result for proportion hypothesis (Class):")
print("z-statistic:", z_stat)
print("p-value:", p_valueProp)

# Confidence Interval for fraud proportion
proportion = fraudCount / totalCount
CI_prop = stats.norm.interval(0.95, loc=proportion, scale=np.sqrt((proportion * (1 - proportion)) / totalCount))

print("95% Confidence Interval for fraud proportion:", CI_prop)
```

```
Z-test result for proportion hypothesis (Class):
z-statistic: 0.6010829247756464
p-value: 0.5477847557024209
95% Confidence Interval for fraud proportion: (0.432178356575096, 0.627821643424904)
```

Mean Transaction Amount:
Result: With a t-statistic of 17.76 and a p-value near zero, we reject the hypothesis. CI: The 95% CI (11196.42, 13850.90) confirms the mean amount is significantly higher than 500.

Proportion of Fraudulent Transactions:
Result: With a z-statistic of 0.60 and a p-value of 0.548, we fail to reject the hypothesis. CI: The 95% CI (0.4322, 0.6278) includes 0.5, supporting true fraud proportion aligns with 0.5.

2. Comparing Two Sample Means and SD

```
# Two different columns to compare
V1_samples = sample_data['V1']
V2_samples = sample_data['V2']
```

```
T-test result for mean comparison of V1 and V2:
t-statistic: 1.0710601027055642
p-value: 0.2854462285472725
```

95% Confidence Interval for the difference in means of V1 and V2: (-0.06955196324952378, 0.542766626781001)

T-test for Mean Comparison of V1 and V2:
Result: The t-statistic is 1.07 with a p-value of 0.285. Thus, we fail to reject the null hypothesis.

Confidence Interval for Mean Difference:
The 95% CI for the difference in means is (-0.0696, 0.5428). Which includes zero, indicating there is no significant difference between the means of V1 and V2.

3. Correlation

```
# Three variables (V1, V2, V3) to calculate correlations
correlation = sample_data[['V1', 'V2', 'V3']].corr()
```

```
Pearson correlation test between V1 and V2:
correlation coefficient: -0.06967627533371841
p-value: 0.4909236073567941
95% Confidence Interval for correlation coefficient between V1 and V2: [-0.26250188  0.12850046]
```

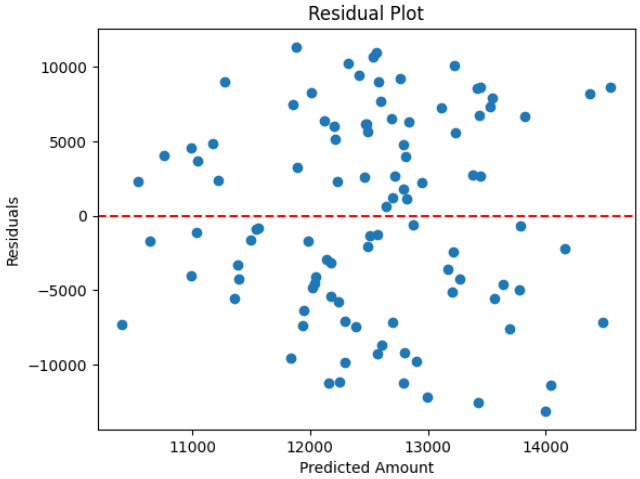
Pearson Correlation Test for V1 and V2: Result:
The correlation coefficient between V1 and V2 is -0.0697 with a p-value of 0.491, This means we fail to reject the null hypothesis.

Confidence Interval for V1 and V2 Correlation:
The 95% confidence interval for the correlation between V1 and V2 is (-0.2625, 0.1285), which includes zero, further indicating no significant correlation.

4. Linear Regression

```
# Linear regression of 'V1' on 'Amount'
X = sample_data[['V24']]
y = sample_data['Amount']
```

Regression Equation: Amount = 908.7910 * V1 + 12523.5741



Regression Plot and R^2 Analysis
The regression line has a slight upward slope, suggesting a very weak positive relationship between V24 and Amount.

Residual Plot Analysis

The residuals have a wide spread, showing that the model has high prediction error and is not accurately capturing the relationship.

So, this is not a good model for predicting Amount

5. Multiple Regression and Model Comparison

```
# Target variable and predictors
target = 'Amount'
predictors = ['V1', 'V19', 'V26']

# Model 1: Using all predictors
reg_eq(model_1, ['V1', 'V19', 'V26'], "Model 1")

# Model 2: (V1 and V19)
reg_eq(model_2, ['V1', 'V19'], "Model 2")

# Model 3: (V1)
reg_eq(model_3, ['V1'], "Model 3")
```

```
Adjusted R² scores:
Model 1: Adjusted R² = -0.2654572126076824
Model 2: Adjusted R² = -0.13541378271925564
Model 3: Adjusted R² = -0.07319183519019923
```

Best Model: Model 3 (using only V1) is the best among the three, as it has the highest (least negative) Adjusted R^2 at -0.0732.