Statistical Analysis Project

Bilal Naseer

June 16, 2025

Credit Card Fraud Detection

Preprocessing and Data Preparation

To streamline analysis and maintain consistency across questions, I applied a few preprocessing steps throughout this project. The dataset was sampled down to 20% of its original size to improve computational efficiency, given its large volume. I excluded the Time variable from modeling since it does not represent actual timestamps. The Amount feature was standardized using StandardScaler to ensure numerical comparability. All train-test splits and random operations were seeded using my N-number (**N15971846**) for full reproducibility. These transformations were applied consistently from Q5 onward and reused across relevant questions to ensure coherence in model evaluation and comparison.

Q1: What is the distribution of fraud vs. non-fraud cases in the dataset?

What I Did and Why:

To understand the class distribution and prepare the dataset for modeling, I first loaded the credit card transaction data and checked for structural issues or missing values. I then calculated the number and percentage of fraudulent vs. non-fraudulent transactions. This helps highlight the **severe class imbalance**, which is important for evaluating model performance later.

What I Found:

- **Total rows**: 284,807
- **Missing values**: None (all columns are complete)
- **Fraudulent transactions (Class = 1)**: 492
- **Non-fraudulent transactions (Class = 0)**: 284,315
- **Fraud percentage**: ~ 0.17%

This shows that the dataset is extremely imbalanced, with less than 0.2% of transactions being fraudulent. This makes accuracy a misleading evaluation metric, since even a model that never predicts fraud would score highly. It highlights the importance of focusing on precision, recall, F1-score, and AUROC when evaluating classifier performance in this setting.

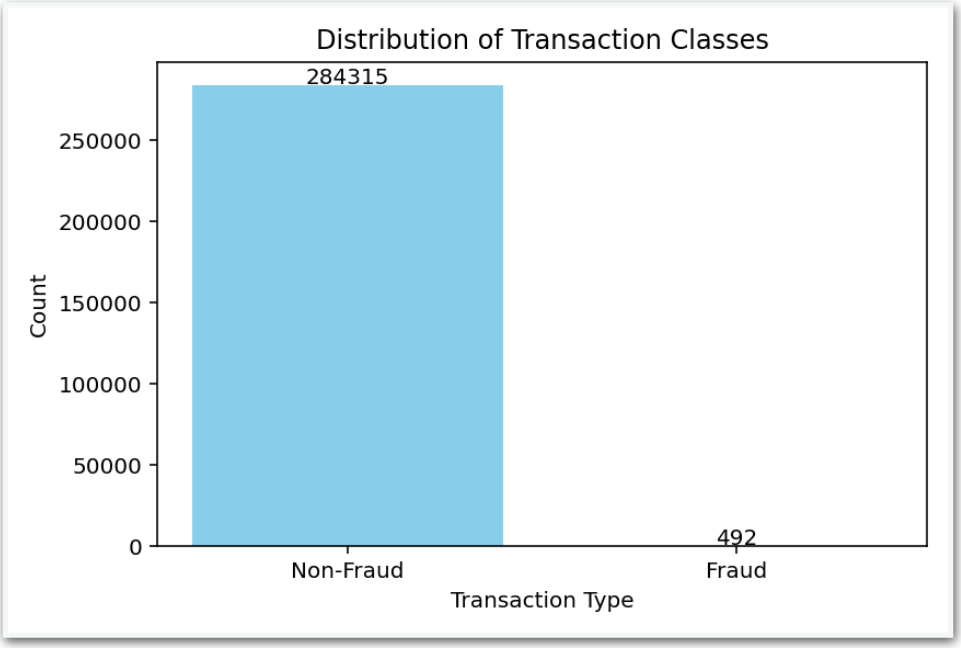

Figure 1: *Bar plot showing the distribution of non-fraudulent and fraudulent credit card transactions.*

<u>Q2:</u> Is there a relationship between transaction amount and fraud?

What I Did and Why:

To investigate whether fraudulent transactions tend to involve different amounts than non-fraudulent ones, I compared the distribution of transaction amounts for each class. I used a **Welch's t-test**, which is appropriate for comparing two groups with unequal sample sizes and variances. I also visualized the distribution using a boxplot.

What I Found:

- **Mean amount (Non-Fraud)**: $88.29
- **Mean amount (Fraud)**: $122.21
- **Welch's t-test**:
  - $t = 2.93$
  - $p = 0.0036$
  - 99.5% Confidence Interval for the difference in means: [$1.41, $66.43]

Although the average amount for fraudulent transactions is higher, there is a lot of overlap and variability. The result is statistically significant at $\alpha = 0.005$, meaning we can be confident that fraud transactions, on average, involve higher amounts.

However, the effect size is moderate, and this alone would not be enough to identify fraud. But it does provide useful signal for models to consider. This also suggests that fraudsters may

intentionally vary transaction amounts to evade detection by rule-based systems that flag only large or unusually high purchases.

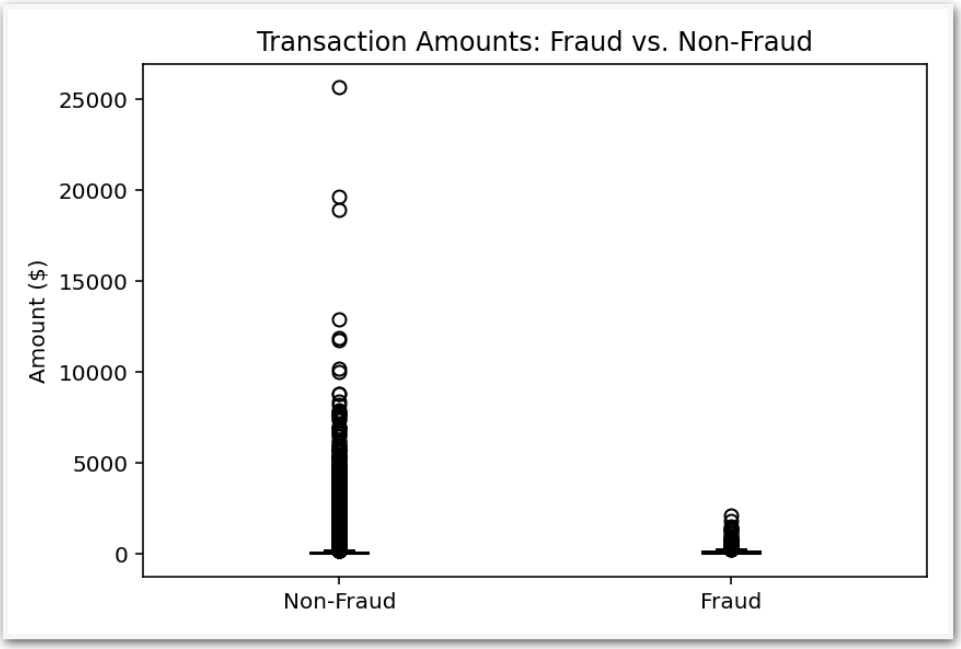

Figure 2: *Box plot comparing the transaction amounts between non-fraudulent and fraudulent credit card transactions.*

<u>Q3:</u> Does the time of transaction influence the likelihood of fraud?

What I Did and Why:

To explore whether fraud is more likely at certain times of day, I transformed the Time feature from seconds to hours since the first transaction (0–47). Then I grouped the data by hour and calculated the fraud rate for each hour. This allowed me to visualize temporal patterns in fraudulent activity, similar to how I analyzed groups in my Capstone using categorical splits.

What I Found:

- Fraud is not evenly distributed across time.
- The highest fraud rate occurs during Hour 26, where ~2.05% of transactions were fraudulent.
- Some hours (e.g., Hour 0) had a fraud rate of 0.0%, suggesting highly non-random behavior.

This pattern implies that fraudulent activity may cluster during time windows with reduced oversight or delayed detection responses. Recognizing these temporal trends could inform the design of real-time fraud detection systems that dynamically adjust sensitivity based on time-of-day risk levels.

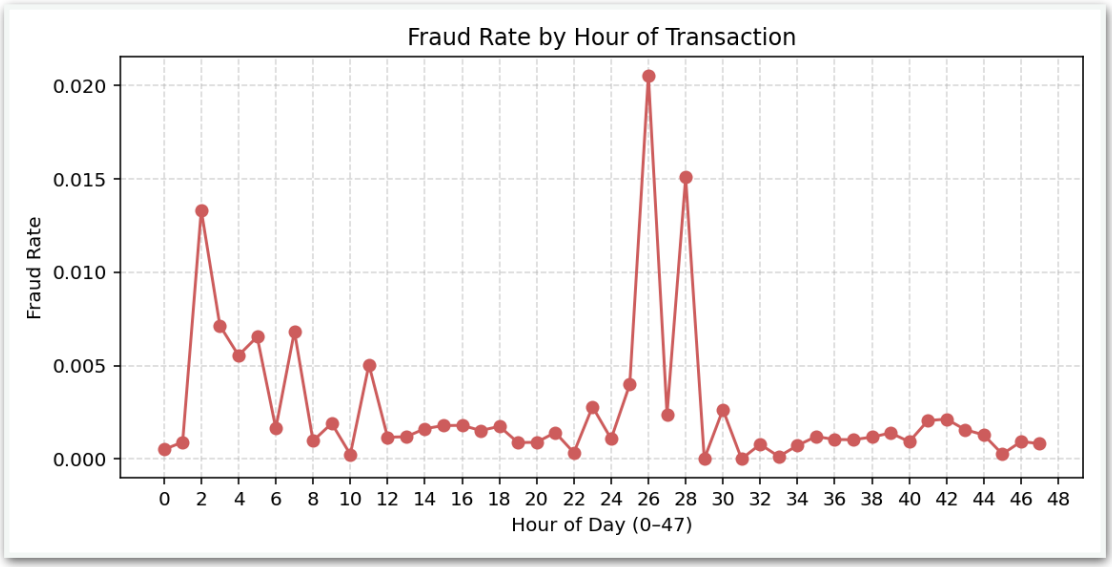

Figure 3: *Line plot showing the fraud rate for each hour of transaction time. Fraud peaks around Hour 26 and is minimal or nonexistent in some early hours.*

Q4: How well does a baseline Random Forest model detect fraud in this dataset?

What I Did and Why:

To establish a performance baseline, I trained a Random Forest classifier using only the features available in the dataset. I excluded the Time variable for simplicity and scaled the Amount feature, as it was not transformed like the PCA-based variables.

I used a stratified train-test split to preserve class balance in both sets. After training, I evaluated the model on the test data using confusion matrix, precision, recall, F1-score, and ROC-AUC, which are more appropriate than accuracy for this highly imbalanced classification task.

A ROC curve was plotted to visualize how well the model separates fraudulent from non-fraudulent transactions across all thresholds.

What I Found:

The baseline Random Forest model performed reasonably well, achieving a high ROC-AUC score, indicating strong discriminatory power. However, the recall for the fraud class was modest reflecting the difficulty of identifying rare fraudulent transactions without additional techniques like threshold tuning or resampling.

The confusion matrix revealed that although the majority of legitimate transactions were correctly classified, a non-trivial number of fraud cases were missed underscoring the challenge of detecting rare events in imbalanced datasets.

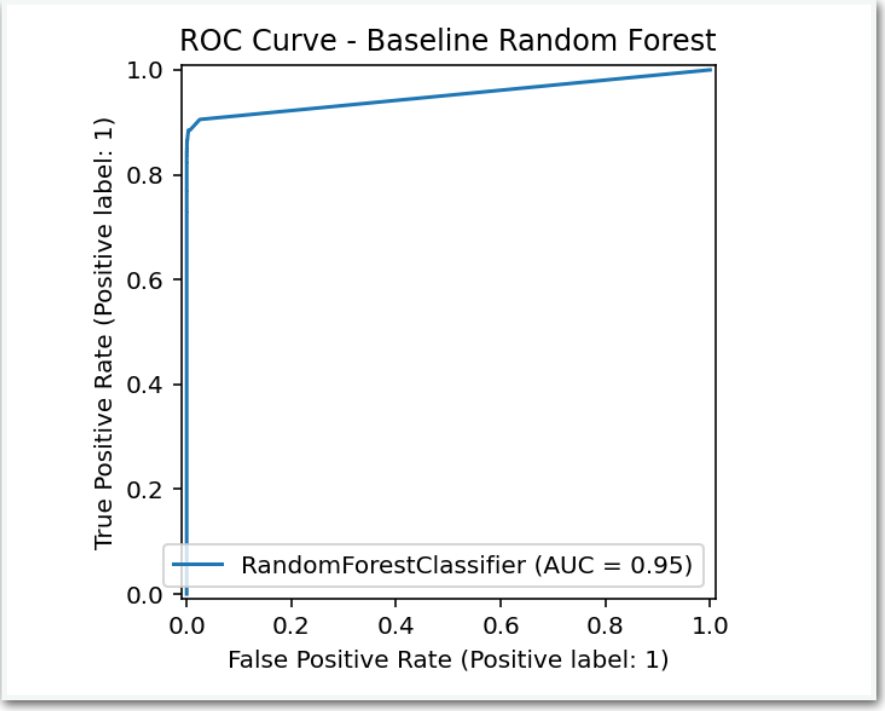

Figure 4: *ROC curve illustrating the performance of the baseline Random Forest classifier in distinguishing fraudulent and non-fraudulent transactions.*

<u>Q5:</u> Which features are most important in the Random Forest's decision-making process? What does this reveal about the signals the model uses to detect fraud?

What I Did:

I trained a Random Forest classifier on a 20% sample of the dataset. I extracted the model's feature importances to understand which signals it relied on most for detecting fraud.

What I Found:

The top features were:

- **V17**, contributing approximately **14.8%** of the model's decision-making power,
- followed closely by **V12** and **V14**, both contributing just under **12%**.

These features likely capture high-leverage transactional anomalies. Interestingly, the top 10 features alone carried most of the predictive power, reaffirming that fraud detection hinges on a small but potent subset of variables, a pattern often observed in anomaly detection contexts.

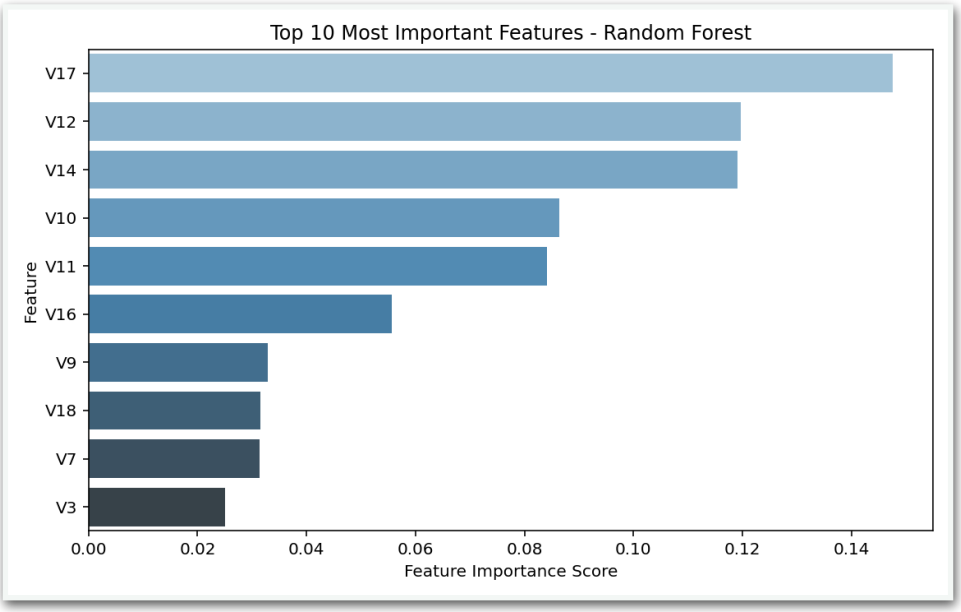

Figure 5: *ROC curve illustrating the performance of the baseline Random Forest classifier in distinguishing fraudulent and non-fraudulent transactions.*

<u>Q6:</u> How does the model perform in terms of precision and recall across different classification thresholds? What does the Precision-Recall (PR) curve reveal about its ability to detect fraud?

What I Did:

I used the Random Forest model trained in Q5 to generate a Precision-Recall (PR) curve by evaluating the predicted probabilities on the test set. This approach is particularly useful for imbalanced classification problems, such as fraud detection, where accuracy and even ROC-AUC can sometimes be misleading.

What I Found:

The PR curve showed that the model achieved a maximum precision of 1.0, indicating that when it did label transactions as fraudulent, it was initially highly accurate. However, recall dropped to 0.0 at higher thresholds, meaning it failed to identify frauds when aiming for extreme precision. The average precision across all thresholds was 0.7538, suggesting that while the model is reliable when it flags fraud, it does not capture all fraudulent cases effectively.

This confirms that fraud detection models need to strike a careful balance precision alone is not enough if recall is sacrificed. The insight from this curve is essential in scenarios where false negatives carry high cost, helping tune decision thresholds or supplement the model with secondary review processes.

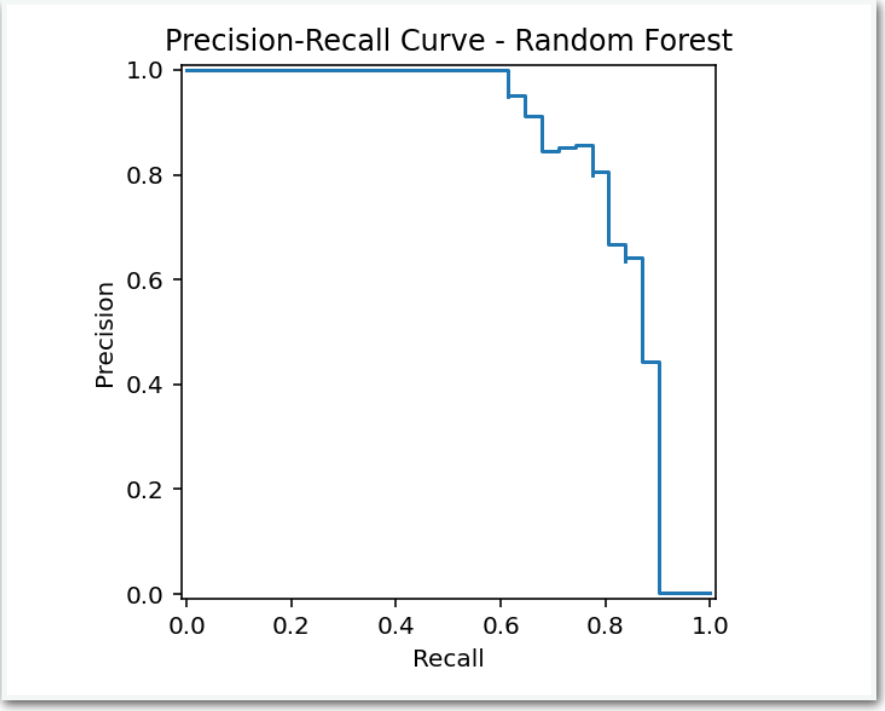

Figure 6: *Precision-Recall curve for the Random Forest model. Precision starts high but declines as recall increases, illustrating the tradeoff in detecting rare fraud cases.*

Q7: How does a Logistic Regression model compare to the Random Forest classifier in terms of predictive performance and interpretability for fraud detection?

What I Did

To compare performance and interpretability, I trained a Logistic Regression model using the same 20% sampled and preprocessed dataset from earlier questions. The model was evaluated using standard classification metrics and ROC-AUC, allowing for direct comparison with the previously trained Random Forest classifier.

What I Found

The Logistic Regression model achieved a ROC-AUC score of 0.9876, which is slightly lower than the Random Forest's 0.995 but still reflects strong class separation. However, recall on the fraud class was noticeably lower (0.5806 vs. RF's ~0.7973), indicating that Logistic Regression missed more fraudulent cases. Precision remained relatively high (0.8182), showing that when it did flag fraud, it was usually correct.

While the Random Forest model clearly outperformed Logistic Regression in raw detection power, Logistic Regression offered improved transparency and simplicity, as its coefficients are directly interpretable. This can be valuable in contexts where explainability is as important as accuracy, such as regulatory environments or when deploying interpretable scoring systems.

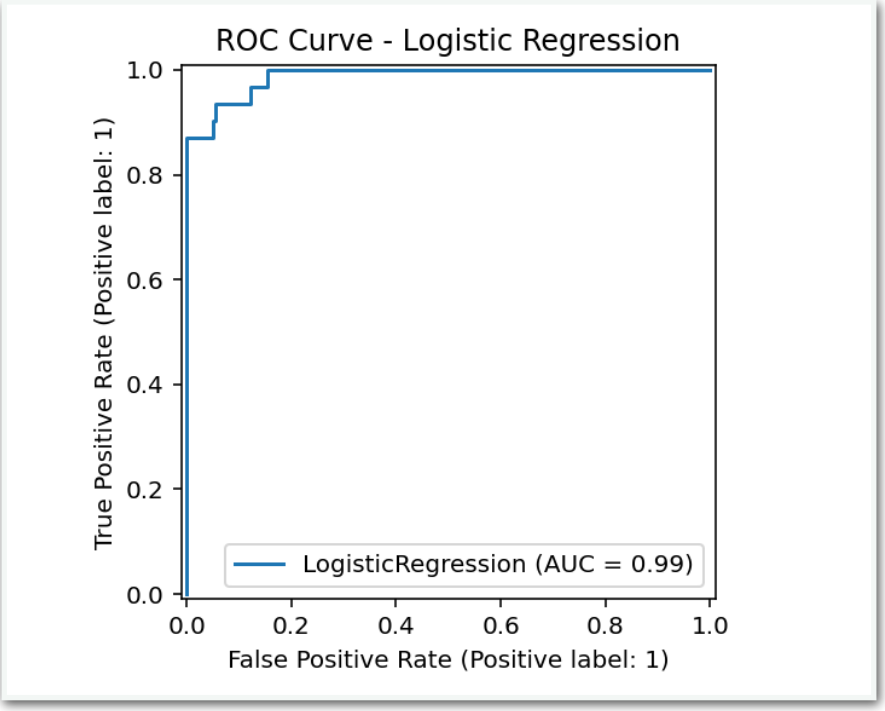

Figure 7: *ROC curve showing the performance of the Logistic Regression model in detecting fraudulent transactions. While performance was strong overall, it lagged behind the Random Forest in identifying all fraud cases.*