

DSCI478 Kaggle Project - Credit Card Fraud Detection

Nick Brady, Jakob Wickham

February 25, 2025

Note for us: - If you want the PDF to not display a cell, click the three dots on the cell, click “Add Cell Tag”, and put “remove_cell” - If you want to hide the code instead, put “remove_input”

1 Introduction

In 2024 alone, 134 million Americans have been victims of credit card fraud, with unauthorized purchases accounting for 6.2 billion dollars annually. This trend of fraud is on the rise in financial and business challenges. (Reference 1). While consumers are generally protected from fraudulent transactions, the impact is beyond the negative experience; financial institutions and businesses bear the cost.

To reduce these risks, financial institutions leverage Machine Learning (ML) and other statistical models to detect and prevent transactions from occurring. Creating effective fraud detection models has challenges, including correctly classifying fraudulent transactions, minimizing customer impact with false positives, and dealing with multiple fraud vectors.

In this project, we will explore the process of creating a fraud detection model, the data set we used, feature engineering, model evaluation metrics, and the challenges of imbalanced data sets. Since fraudulent activity is rare, accuracy metrics are not applicable when determining model evaluation as they only give a score on how well it detects *non-fraud*. In this project we will be using precision, recall, F1 score and the Precision-Recall Curve (PR - AUC) to provide an accurate representation of the model given the challenges in the data set

1.1 The Dataset

Due to the nature of the information for credit card purchases, we used a synthetic data set (References 2 & 3), which allows us to explore fraud detection techniques while also maintaining privacy and security concerns. At the end of this section is a list of the features.

This synthetic data can help us in research and model experimentation. It introduces limitations: the data may not reflect real world approaches that fraudsters use, customer transactions may lack variability, and `TX_FRAUD_SCENARIO` can create potential data leakage. With those limitations in mind, we want to be able to focus on building feature engineering and methods that can generalize well to actual scenarios. We must first explore the data cleaning and processing steps taken to prepare the data set for modeling.

Feature	Description
TRANSACTION_ID	A unique identifier for the transaction
TX_DATETIME	Date and time at which the transaction occurs
CUSTOMER_ID	The identifier for the customer. Each customer has a unique identifier
TERMINAL_ID	The identifier for the merchant (or, more precisely, the terminal). Each terminal has a unique identifier
TX_AMOUNT	The amount of the transaction
TX_FRAUD	A binary variable with the value for a legitimate transaction or the value for a fraudulent transaction
TX_TIME_SECONDS	Timestamp of transaction in seconds
TX_TIME_DAYS	Timestamp of transaction in days
TX_FRAUD_SCENARIO	Numerical indicator of the type of fraud scenario

2 Data Cleaning and Preprocessing

Before conducting exploratory data analysis and model development, we had to clean data and process the data.

Since this data was synthetically generated, no missing (NaN) values were present. In the data, the TX_FRAUD_SCENARIO column was dropped to prevent data leakage as it contained information directly to the fraud classification. TX_DATETIME was transformed into multiple features—TX_HOUR (hour of the transaction) and TX_DAYOFWEEK (day of the transaction week)—to capture temporal patterns. Once that transformation was completed, TX_DATETIME was dropped due to redundant data.

Completing these preprocessing steps ensure that the dataset was structured for feature extraction while also minimizing data leakage. Transforming TX_DATETIME allowed us to capture temporal fraud patterns. With these preprocessing steps done, the data set is now ready for exploratory data analysis (EDA), where we will be determining patterns in fraudulent and non-fraudulent transactions, key trends and assess potential predictive features.

3 Exploratory Data Analysis

Initially we performed basic data exploration to learn more about our data set. One of the first things in our exploration analysis that was noticeable was that the data set was highly imbalanced, with only fraudulent activity making up a small proportion of the total (which was a given considering we’re working with fraud). Being aware of class balance is important as it affects model performance and evaluation metrics. Additionally, we analyzed the average transaction that customers typically make with their cards. With this information we wanted to determine if going above or below the average transaction was more likely fraudulent. Since this was a significant factor in fraud detection, we further explored whether a customer’s personal average carried more of a determining factor than global average (See Figure 1).

	Metric	Count	Percentage
0	Fraud Above Personal Average	2169	0.62%
1	Fraud Not Above Personal Average	1303	0.38%

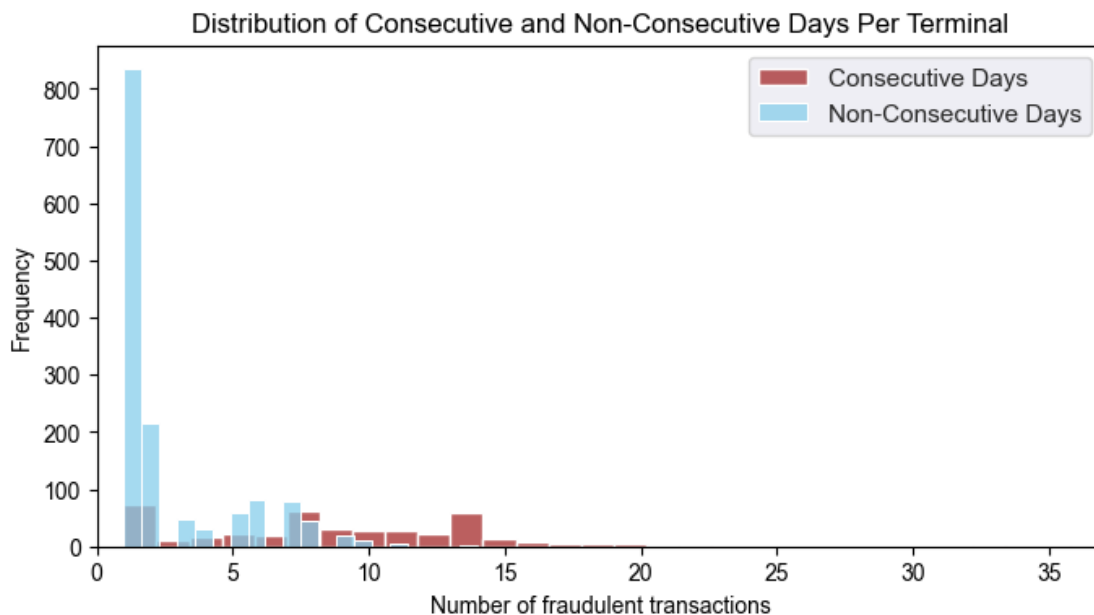
Our findings showed that incorporating a customer’s historical spending behavior improved the

chance of finding fraudulent activity, which could show that people who commit fraud are trying to stay within the expected spending patterns of the individual.

We also investigated whether certain terminals were consistent with fraudulent activity (See Figure 2).

fraud_pattern	Consecutive	Non-Consecutive
Terminal_Count	323.00	1139.00
Fraud_Count	7831.00	4030.00
Terminal_Percentage	22.09	77.91
Fraud_Percentage	66.02	33.98

This revealed that a small number of terminals were responsible for a large portion of fraud, often occurring over consecutive days. However, there was no fixed pattern in the number of days fraud occurred and no clear distinction between whether a certain day should be a concern (See Figure 3).



To address this, we had to implement a way in our classification approach that acknowledges that past fraudulent activity at a terminal is a strong factor, but not all future transactions at that terminal should be flagged as fraud.

Our exploration analysis showed many key significant insights into fraud detection, which include class imbalance, transaction amount and terminal level indicators. Those findings gave us information that was necessary to implement our feature engineering and modeling approach to account for the customer and terminal behavior patterns and minimize our bias in our model.

4 Feature Engineering

To create a robust fraud detection, we engineered features that capture both population fraud trends and individual customer-based spending behaviors. This creates a model that can detect a wide range of fraud tactics while also identifying personalized anomalies that could indicate fraudulent activity at a customer level.

Transaction amount serves as a critical indication in fraudulent activity. We applied a Z-score normalization to standardize transactions relative to the dataset mean, which ensures that large transaction values were flagged as unusual. However, some of the fraudulent activities attempted to fall within the customer expected ranges but still would appear abnormal compared to overall transaction patterns. Since specific terminals were exploited, we implemented rolling seven-and twenty-eight day rolling fraud rates. Acknowledging that a terminal who has had fraudulent activity doesn't mean they will always be fraudulent we implemented an exponential decay factor to ensure old occurrences of fraud had a diminishing influence in our model and attempting to prevent model but still capturing consistent fraud risk. To further improve our population fraud trend, we implemented a Local Outlier Factor (LOF), and an unsupervised anomaly detection method that assigned a likelihood of fraud score based on its behavior. This method creates context-based detecting and maintains to not solely rely on extreme transaction values. During the feature engineering process we considered using an Isolation Forest for our unsupervised choice, but the anomaly score made our model rely on the Isolation Forest prediction, which reduced its ability to learn other fraudulent trends.

Listed below shows all population-based features that were created.

Feature	Purpose
TX_AMOUNT_Z_Score	Check for extreme transaction amounts compared to data set's mean
TX_AMOUNT_Percentile	Check transaction amounts relative to all others
TERMINAL_FRAUD_RATIO	Terminal-level fraud check that decays over time
ANOMALY_SCORE (LOF)	Unsupervised learning to apply a score beyond deviation of previous features

The population-based trend provides a big picture view of our sample, but fraudulent transactions often are different from customer behavior rather than population norm. To recognize this, we engineered features that will adapt to each customer's unique spending habits instead of over relying on dataset wide fraud indicators. The key feature of detecting fraud at a customer level is track spending behavior over time. We computed a 14-day rolling window that tracks whether a customer suddenly makes a significant change in their spending habits. This can help with identifying fraud scenarios where fraudsters make large transactions over a short period, deviation from previous spending habits. We normalized the transaction amount based on each customer's historical data instead of the entire data set. This implementation prevents the model from flagging frequently high spending larger transactions, while still detecting unexpected large purchases from lower spending customers.

The table below shows all customer-based features that were created.

By looking at both the population and customer-based fraud detection the purpose is to create a model that can identify fraud trends that are occurring at large scale but also adapting to customer level behavior. This combination with dynamic adjusting and personalized profiles attempts to

Customer-Based Features	Purpose
SPENDING_RATIO_CHANGE	Checks for sudden shifts in customer spending behavior
SPENDING_Z_SCORE_28D	Identifies how unusual a transaction is for a specific customer based on their historical spending patterns

create a fraud detection model that is both effective and adaptable at identifying ever changing fraud tactics. With the creation of these engineered features, we now will focus on model selection and evaluating different approaches at models to classify fraudulent activity effectively

5 Model Selection

Isolation Forest, Balanced Random Forest, etc

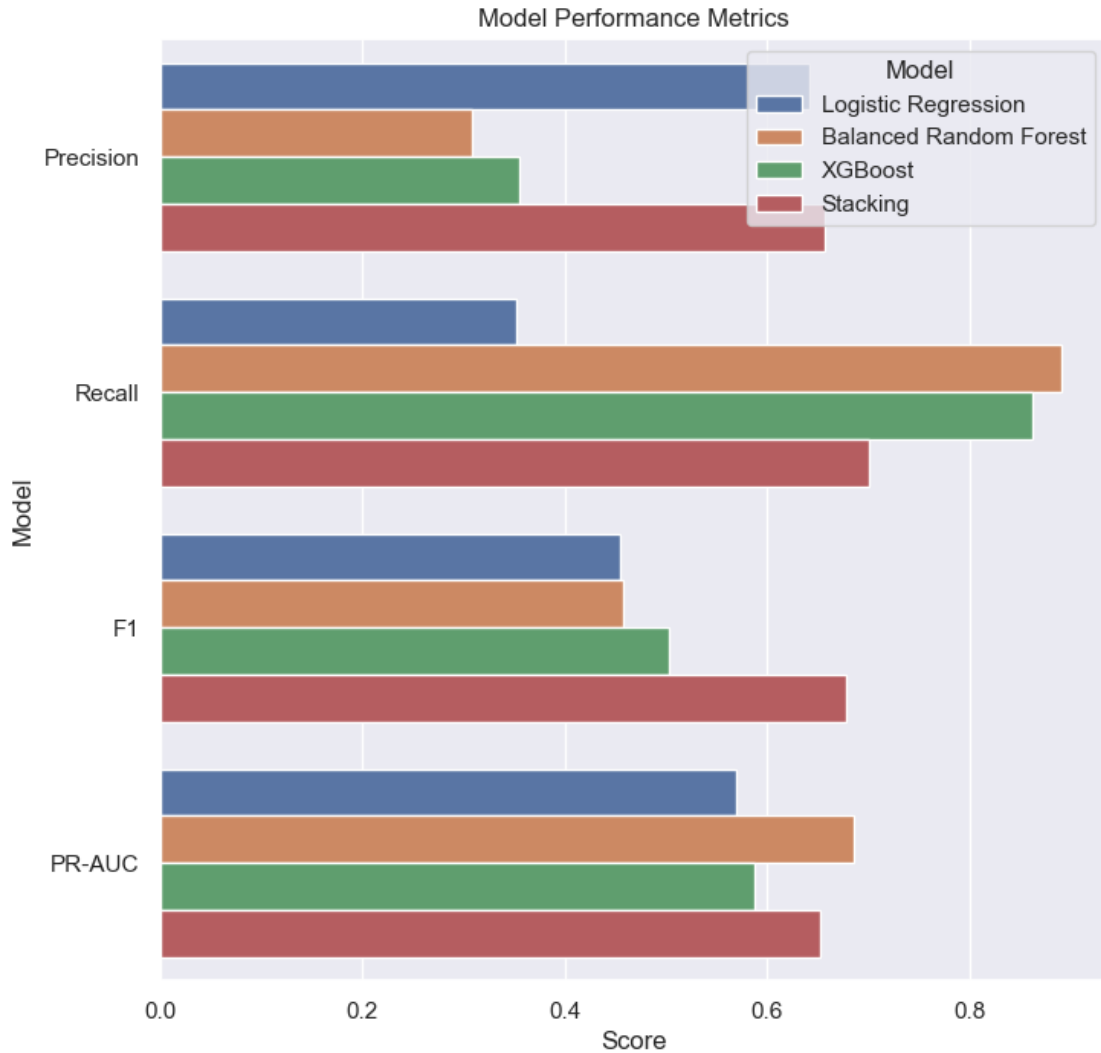
6 Model Training and Evaluation

Let's talk a little bit about how we're going to be scoring our models. We'll be using 4 scores to determine how well our models perform: Precision, Recall, F1, and a PR-AUC score.

- Precision (PPV): Correctly identified fraudulent cases across all *classified* fraudulent cases
 - Also called the “positive predictive value”
- Recall (TPR): Correctly identified fraudulent cases across all *truly* fraudulent cases
 - Also called the “true positive rate”, power, or sensitivity
- F1: A metric providing a balanced measure of the harmonic mean of precision and recall
- PR-AUC: The model's ability to distinguish between classes

The score we'll be mainly looking at is the F1 score as it'll tell us how good our model is at classifying both non-fraud and fraud cases. The higher the score, the better the model is at both *predicting* and *detecting* a fraudulent case.

We won't be working with accuracy scores as with imbalanced data, accuracy scores tend to mislead people into believing the model fits the data well by reporting high numbers, when in reality it's only that high because it can accurately predict the *majority* class while being horrible at accurately predicting the *minority* class.



7 Model Interpretation and Explainability

Now that we have the results of our models done, let's try and make some sense of them.

7.0.1 BRF using the Base Dataset

Considering that we decided to feature-engineer the dataset to the point where none of the original features were left being used by our models, how well does the original dataset perform with one of the model types we used?

Classification Report:

	precision	recall	f1-score	support
0	0.99	0.82	0.90	351466
1	0.02	0.43	0.04	3177

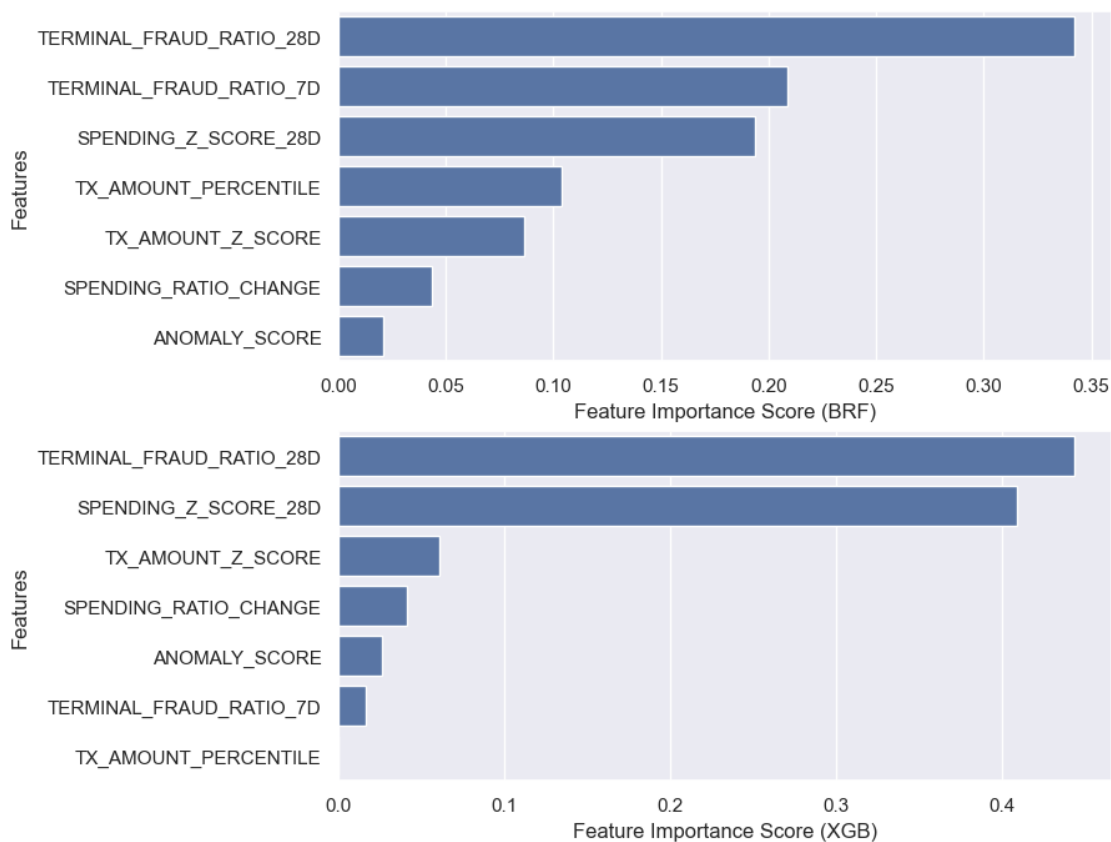
accuracy			0.82	354643
macro avg	0.51	0.63	0.47	354643
weighted avg	0.99	0.82	0.89	354643

PR-AUC: 0.2375

With a precision rate of 0.02, recall rate of 0.43, and F1 score of 0.04, the original dataset does not hold up well to the feature-engineered dataset. This makes sense as we assume the underlying nature of the data includes temporal information—as discussed in the feature engineering section—that is not explained in the original dataset.

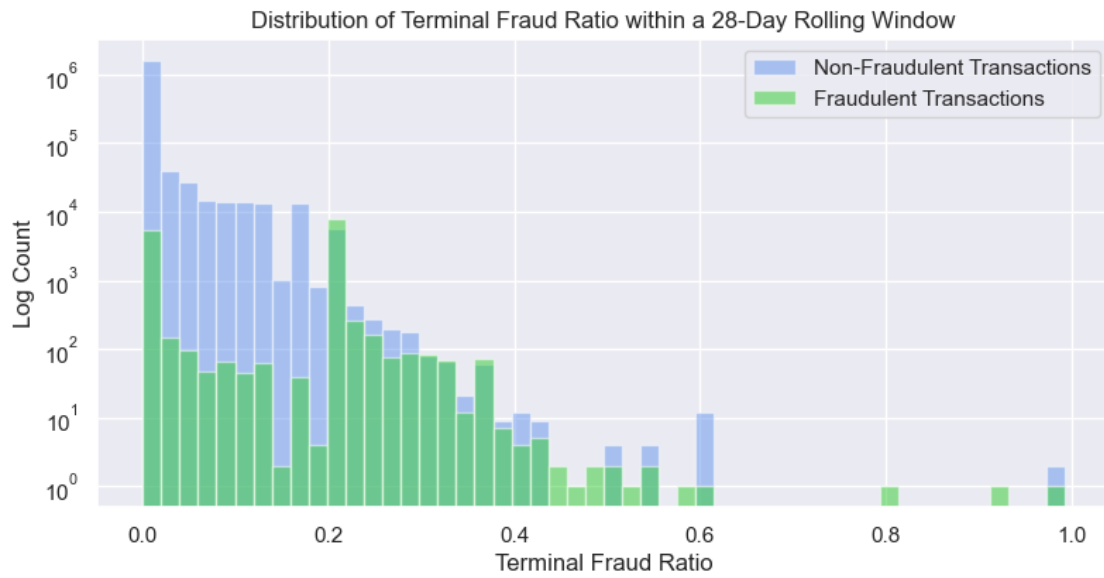
7.0.2 Feature Importance

One thing we decided to look into after performing the models was seeing if the feature importances were similar between the Balanced Random Forest and XGBoost models.



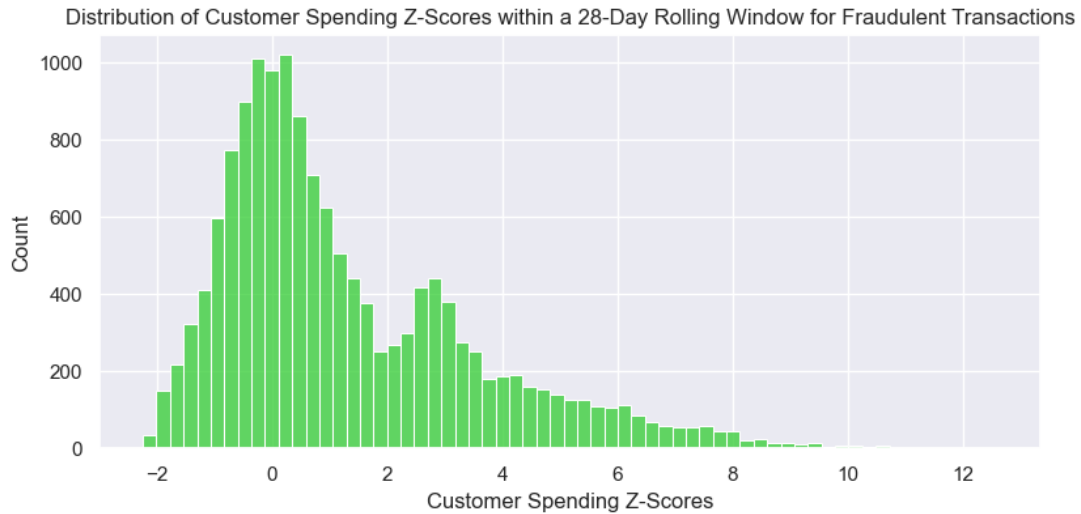
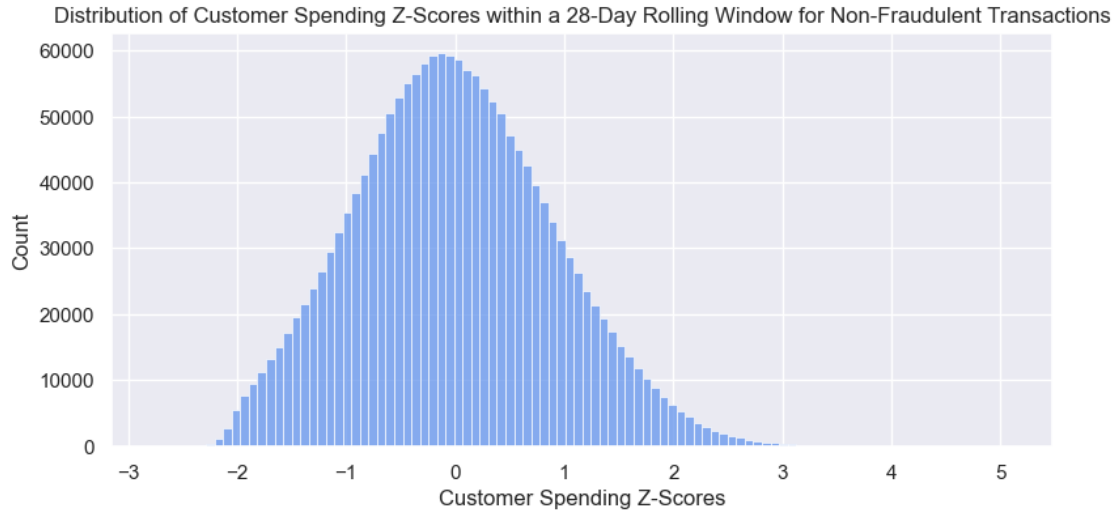
Both models look to have `TERMINAL_FRAUD_RATIO_28D` as the feature with the highest importance score, but the `SPENDING_Z_SCORE_28D` feature appears 3rd in the BRF and 2nd in the XGB feature importance. We decided then to graph the distributions of these features, seeing if these distributions can explain why the models have these features as the highest out of the others. We didn't pay much attention to `TERMINAL_FRAUD_RATIO_7D` as it's 28 day counterpart is more interesting.

First, we looked at the most important: `TERMINAL_FRAUD_RATIO_28D`



The graph doesn't look too remarkable at first, but for the fraud transactions, the feature does have a spike at around 0.2.

Next, we looked at `SPENDING_Z_SCORE_28D`



These graphs show a lot more important information. For non-fraudulent transactions, the distribution is centered at 0 and tapers off, almost looking like a normal distribution of some sort. As for the fraudulent transactions, the distribution also has a majority of the data around 0, but the range of values is much higher, and these higher Z-scores have more transactions within them, possibly meaning that for transactions with a high customer spending Z-score, they may have a good chance of being fraudulent.

7.0.3 Interpretability vs Accuracy

Why did we choose to use more complex models such as random forests and XGBoost to fit our data instead of simpler models such as logistic regression? Well it all boils down to what we really want to get out of this data. Simpler models like logistic regression output coefficients that can explain each feature's impact on the response assuming the other features are held constant, but these kinds of models assume a linear or parametric relationship between the features and the response, thus not being able to fit complex relationships well. Complex models like BRF and XGBoost are

able to fit complex relationships well without the assumptions, leading to them being considered “accurate” models.

But with that accuracy comes a downside: lack of interpretability. Most of these complex models are what’s known as “black-box models”, which means simply that we don’t know what the underlying nature is the model sees. We can use things like feature importance for BRF and XGBoost to try and understand what features the model believes are important, but if we want to know a *direct* relationship between the features and the response, these models won’t help.

In the case of fraud detection, we mainly care about seeing if a model can accurately predict whether a transaction was fraud or not, so we didn’t care about having a lack of interpretability, but with the models we chose, we did use feature importance to try and interpret if what the models thought was important showed themselves in graphs, which is what we found.

8 Conclusion

- What could we change and improve upon
 - Strengths
 - Key Takeaways
 - Additional real time features?
- Talk about the challenges we faced
 - Did Certain fraud patterns remain undetected
- Talk about if we choose to implement this into real world
 - How can it be used
 - Impact on customers
 - Could this be deployed
 - How much human interaction would be necessary due to this implementation

9 References

TODO: Figure out how to wrap text

- <https://www.security.org/digital-safety/credit-card-fraud-report/>
- <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>
- https://fraud-detection-handbook.github.io/fraud-detection-handbook/Chapter_3_GettingStarted/Simulation.html
- <https://www.aporia.com/learn/ultimate-guide-to-precision-recall-auc-understanding-calculating-using-pr-auc-in-ml/>