Springboard--DSC
Capstone Project 2

Credit Card Fraud Detection

By Christopher Chung
November 2021

1. Introduction

Instead of cash transactions, more and more credit cards are being used for day-to-day purchases. While this allows for more opportunities to spend money, it also opens up the risk of credit card fraud, which affects the merchant, the issuing bank, and the consumer.

In this scenario, we will assume that the intended stakeholders are the executives of a bank that issues credit cards. Our charge is to create and train a model that can identify whether a transaction is fraudulent or not using labeled data of over 1 million transactions. For this particular business problem, we will assume a situation where both false positives (valid transactions incorrectly labeled as fraud) and false negatives (fraud transactions incorrectly labeled as valid) are equally costly to the bank.

False negatives are costly to the bank, particularly for card present transactions. If a fraudulent transaction is not blocked at the time of purchase, the issuing bank is liable for the charge once the transaction is deemed as fraudulent by the consumer.

On the other hand, false positives can decrease consumer trust in a bank and negatively affect the credibility of the issuing bank when a legitimate purchase is blocked. In addition, false positives also hurt the merchant. An article by Forbes affirms that a study found that 1 in 3 consumers in the US say they will not do business again with a merchant which declined the card of a legitimate purchase.

Ultimately, there will be a tradeoff between the number of false positives and the number of false negatives in our model. Keeping in mind that both are equally costly in this scenario, our goal will be to optimize the f1-score of the fraud class because it best captures the combination of both precision and recall.

After testing multiple models with a variety of class balancing techniques, our XGBoost model performed the best with an f1-score of 83%. The significance and meaning of this will be discussed in later sections of this report.

The implementation details can be found in the notebooks, along with all the project deliverables, in this GitHub repository.

2. Approach

Data Acquisition and Wrangling

The dataset is a simulated credit card transaction dataset from Kaggle. The dataset was simulated by using Sparkov Data Generation tool created by Brandon Harris. It contains legitimate and fraudulent transactions from the duration 01 Jan 2019 - 31 Dec 2020. It covers credit cards of 1000 customers doing transactions with a pool of 800 merchants.

The raw dataset contained 1,852,394 rows and 22 columns. There were no rows with any missing data. There were, however, multiple categorical variables that would most likely not be useful to generate insights or be used for modeling. As a result, we dropped the ‘jobs’, ‘street’, ‘city’, ‘zip’, and ‘unix_time’ columns.

Additionally, the ‘amt’ column, which represents the transaction amount, had a significant number of outliers, as seen below in figures 1 and 2.

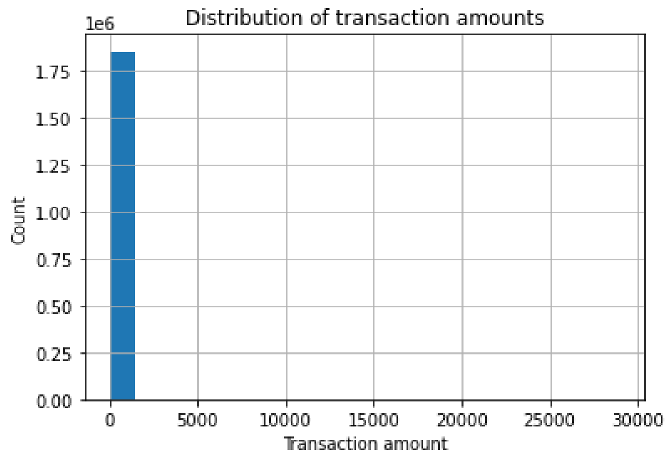

Figure 1: Distribution of transaction amounts

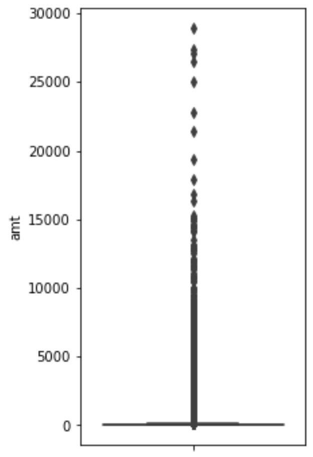

Figure 2: Boxplot of transaction amounts

We identify our outliers by using the standard 1.5*IQR method, where we set the lower bound and upper bound as Q1 - 1.5*IQR and Q3 + 1.5*IQR. Before removing outliers, we wanted to understand how it would affect our target variable, ‘is_fraud’. As a result, we created two boxplots of transaction amounts, one for valid transactions and the other for fraud transactions, side-by-side for comparison, as shown in figure 3.

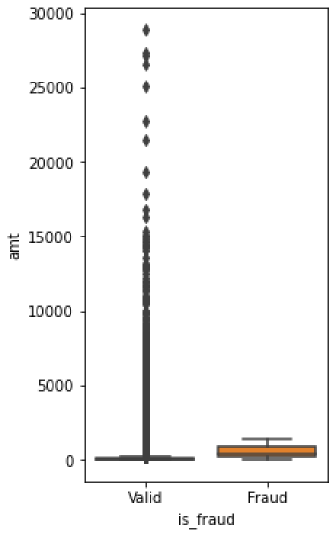

Figure 3: Transaction amounts for valid and fraud transactions

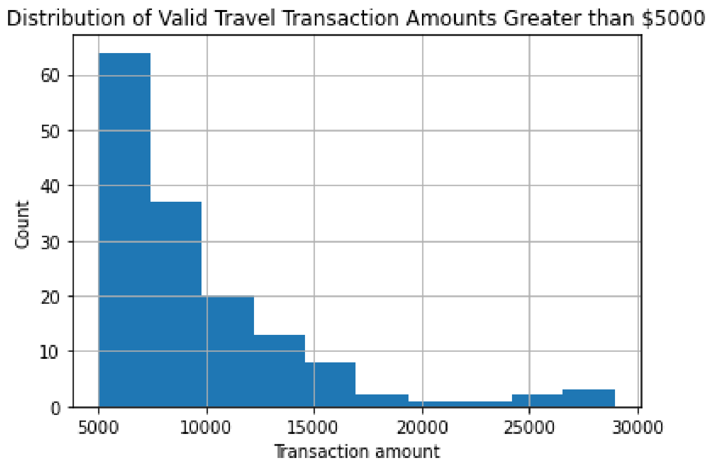

Figure 4: Distribution of valid transactions amounts in ‘travel’ category greater than $5000

Upon further inspection, we discovered that most of the amounts that were greater than $5000 were in the travel category, as shown in figure 4 and the table below:

Number of Valid Transactions Greater than $5000 by Category	
travel	151
shopping_pos	25
shopping_net	19

As a result, we decided to use $5000 as the threshold for removing outliers, provided that the transaction is in the 'travel' category.

In a real business scenario, we would first have a conversation with our client to discuss the potential cost of removing these outliers. Here, we will assume that this conversation occurred and that the client informed us that it is OK to remove these outliers.

Exploratory Data Analysis and Initial Findings

Before diving into any of the other variables, we first explored the target variable, 'is_fraud', to see how many of each class we are dealing with. The results are shown in the table below:

is_fraud	Count	Percentage
0	1,842,592	99.479%
1	9651	0.521%

As expected, the classes are highly imbalanced, which we will need to address once we get to the modeling stage.

From the 'trans_date_trans_time' variable, we extracted a few additional variables, including the hour of day of the transaction and the day of week of the transaction. Based on the comparison of the graphs of number of transactions by hour of day for valid and fraud transactions, it's clear that most fraudulent transactions take place from 10 PM to 4 AM, as shown in figure 5.

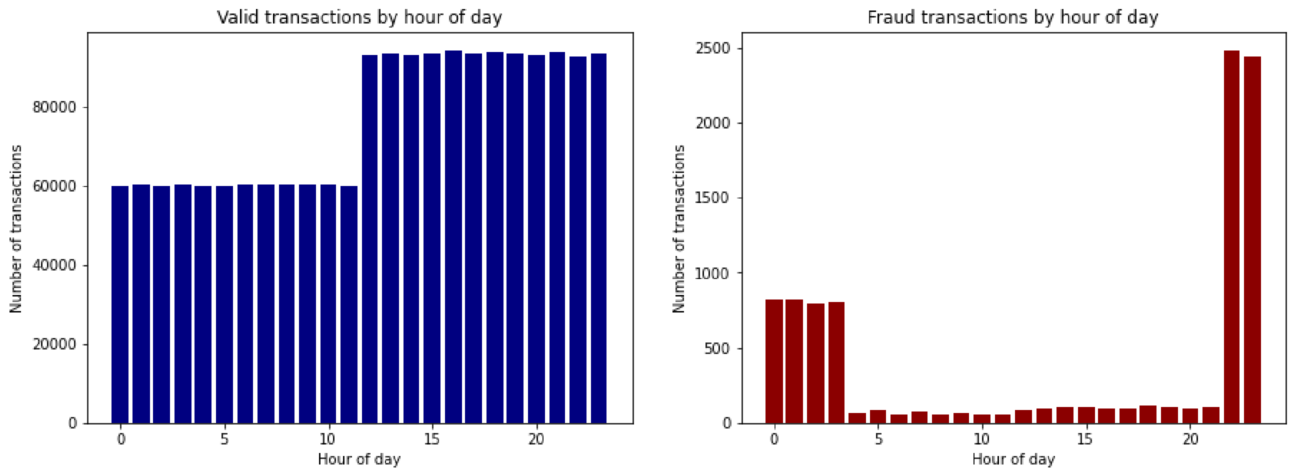

Figure 5: Number of transactions by hour of day for valid and fraud transactions

In order to utilize this valuable feature in a machine learning model, we encoded this information into a binary variable, called ‘abnormal_hours’. If ‘abnormal_hours’ is 1 for a transaction, this means that it occurred sometime between 10 pm and 4 am, and 0 for otherwise.

In figure 6 below, we graphed the bar plots of the number of transactions by each day of week for valid and fraud transactions.

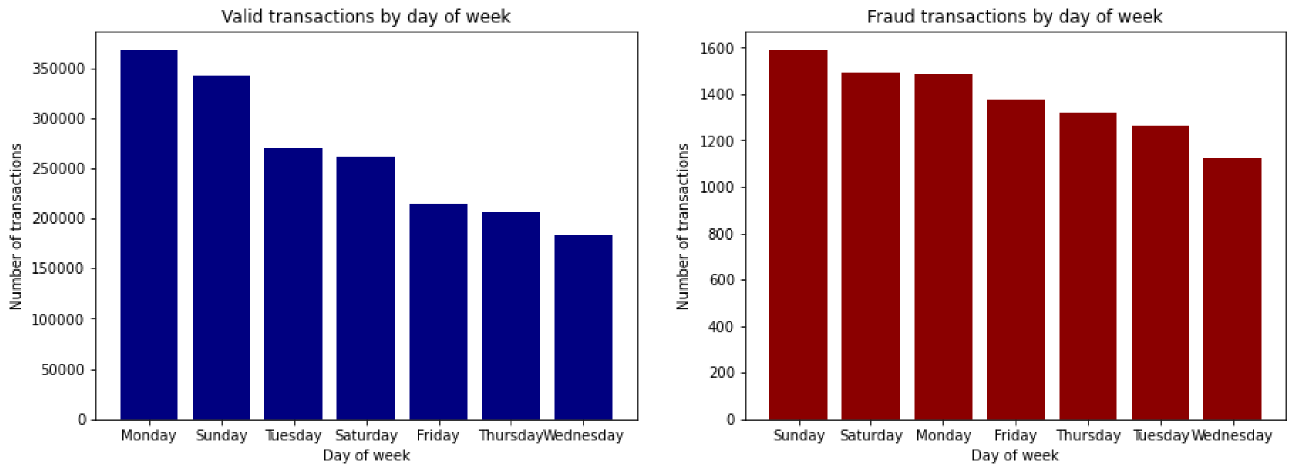

Figure 6: Number of transactions by day of week for valid and fraud transactions

We can conclude that most valid transactions take place on Mondays and Sundays, and most fraud transactions take place on Sundays, Saturdays, and Mondays. In order to prepare this variable for our algorithms, we one-hot encoded this variable and generated dummy variables for each day of the week.

We also extracted the number of seconds passed since the last transaction on that same credit card, and visualized this information as shown in figure 7 below.

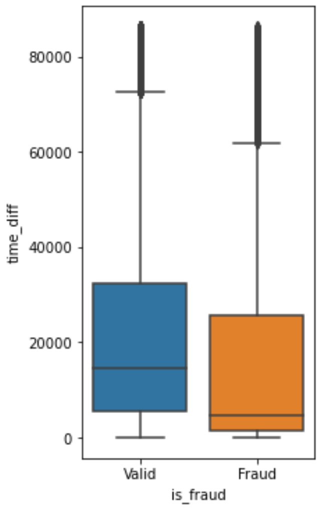

Figure 7: Number of seconds since the last transaction from that specific credit card

We can conclude that, on average, fraud transactions have less time in between transactions than valid transactions. It is possible that once an offender steals someone's credit card information, they will attempt to make many transactions as quickly as possible.

To further explore this, we extracted the number of transactions in the last 1/7/30 days for each credit card and transaction. The data is visualized in figure 8 below:

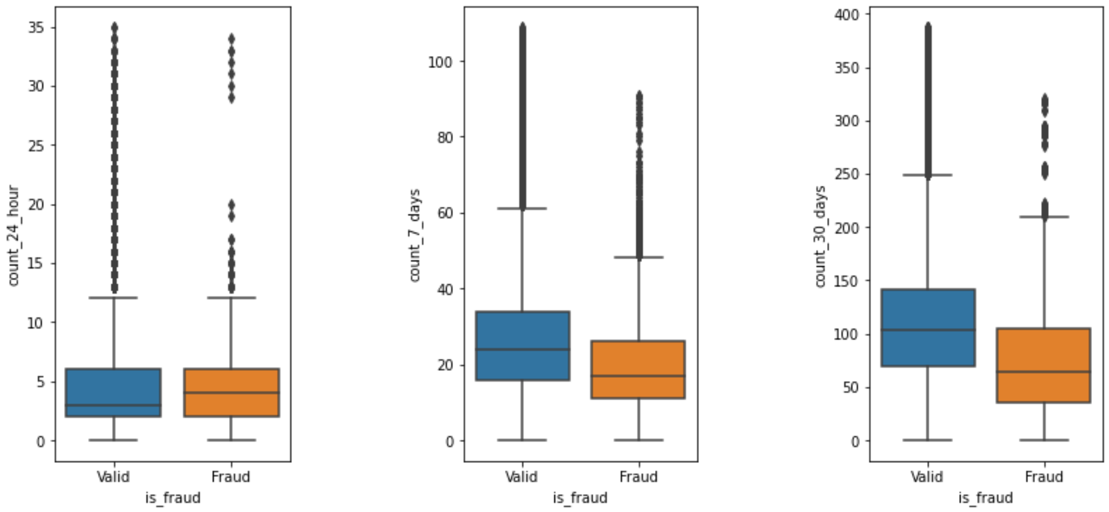

Figure 8: Boxplots of number of transactions in the last 24 hours, 7 days, and 30 days

While, on average the number of fraud transactions in the last 24 hours is greater than that of valid transactions in the last 24 hours, the *opposite* is true when we look at the last 7 days and the last 30 days. It is likely that when someone's credit card information is stolen, the offender will use the credit card as much as possible in the first 24 hours and then stop using it after a certain point in time. This halt could be due to the offender discarding the stolen card information or the card owner discovering fraud transactions on their own and reporting it to the issuing bank.

We were also able to draw valuable insights from the ‘amt’ column, identifying the amount of each transaction. After removing a fair number of severe outliers from the data wrangling stage, we can construct the boxplots below.

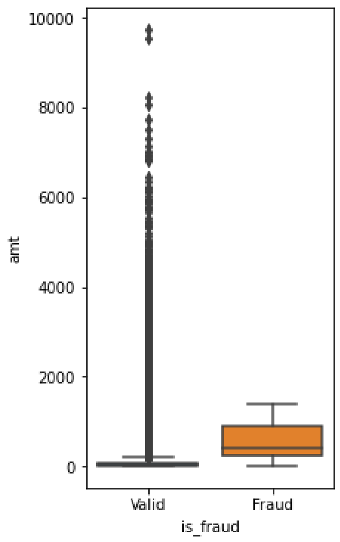

Figure 9: Boxplots of transaction amounts for valid and fraud

Aside from the outliers, we can see that valid transactions typically have significantly lower amounts than fraud transactions. Furthermore, when we calculate the mean amount of each class, we find that valid transactions have a mean amount of $66.88 while fraud transactions have a mean amount of $530.66. This is particularly significant because even though the valid transactions have outliers on the extreme positive side, it’s mean is still notably lower than that of fraud transactions.

Our dataset also included a ‘category’ variable, which indicates the category of the transaction (i.e. entertainment, travel, grocery, etc). We plotted a bar graph of the transaction count of each category for valid and fraud transactions, shown by figure 10 below.

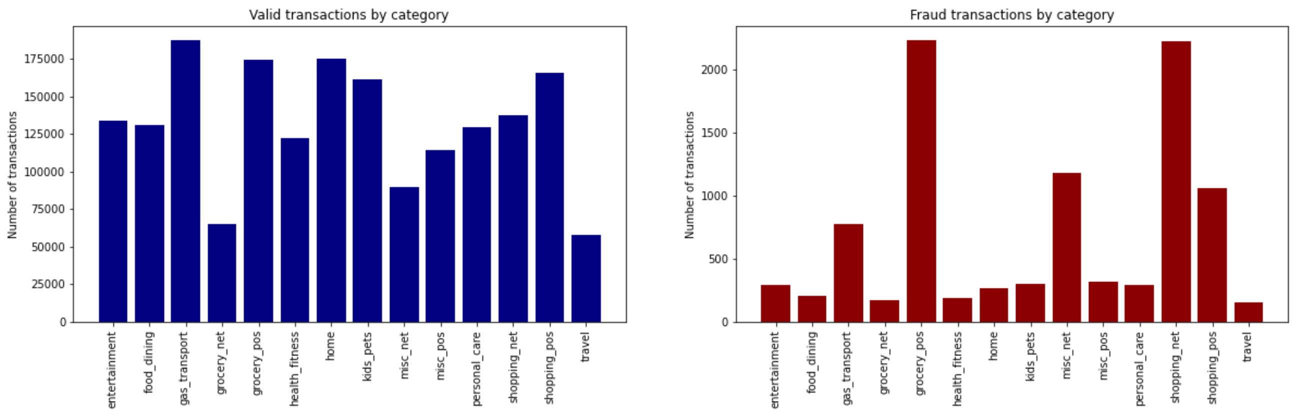

Figure 10: Bar graphs of number of transactions by category for valid and fraud transactions.

We can see that most fraud transactions fell in the ‘grocery_pos’ and ‘shopping_net’ categories, followed by the ‘misc_net’, ‘shopping_pos’, and ‘gas_transport’ categories. To utilize this information, we also one-hot encoded this variable with the prefix ‘cat’ (i.e. ‘cat_grocery_pos’).

We also produced a correlation heatmap (found in the 03_preprocessing notebook), which calculates the correlation between each and every numeric variable in our dataset. We conclude that the following variables have higher correlations with our target variable, ‘is_fraud’, and may be good predictors of fraud:

- ‘amt’
- ‘abnormal_hours’
- ‘count_30_days’
- ‘cat_shopping_net’
- ‘count_7_days’
- ‘cat_grocery_pos’
- ‘cat_misc_net’

Baseline Modeling

To get a baseline model with performance metrics, we split the data into training and testing sets using the 80/20 ratio. To guarantee that the ratio of valid/fraud transactions was preserved in both the training and testing set, we passed our target variable ‘is_fraud’ into the ‘stratify’ parameter of sklearn’s train_test_split function.

We chose Logistic Regression as the algorithm for our baseline model because of its simplicity to use and quick results for binary classification. At this stage, we trained the model two separate times, one with unscaled data, the second with standardized data. Our performance was slightly

After training the model on the training set, the classification report for the test set is shown in the table below:

Scaled	**Valid Precision**	**Valid Recall**	**Valid f1-score**	**Fraud Precision**	**Fraud Recall**	**Fraud f1-score**
Unscaled	0.99	1.00	1.00	0.19	0.03	0.06
Scaled (with StandardScaler)	1.00	1.00	1.00	0.47	0.11	0.17

Given that our metrics on the majority class are unsurprisingly high, we will focus our attention on the metrics on the minority class, which is fraud, because that is the class that we are trying to accurately predict. By running a quick comparison of baseline models with unscaled data vs scaled data, we can see that performance is slightly improved when using scaled data.

According to the precision score, our baseline model's predictions for fraud transactions were 47% accurate. According to the recall score, our baseline model captures only 11% of all fraud transactions. The f1-score is calculated as the harmonic mean of precision and recall.

Needless to say, this is still not an acceptable result and we can attribute the poor performance of this model to the heavily imbalanced classes in this dataset. It is unreasonable to expect any significant improvement in performance without dealing with the class imbalance, which brings us to extended modeling.

Extended Modeling

Having seen the poor performance of our Logistic Regression baseline model on our imbalanced dataset, we will try additional algorithms with four class balancing techniques and generate twelve models for comparison as follows:

Logistic Regression with: ● Random oversampling ● Random undersampling ● SMOTE ● Set class_weight parameter as 'balanced'	**Random Forest** with: ● Random oversampling ● Random undersampling ● SMOTE ● Set class_weight parameter as 'balanced'	**XGBoost** with: ● Random oversampling ● Random undersampling ● SMOTE ● Set scale_pos_weight parameter as the ratio of valid to fraud transactions

The random oversampling, random undersampling, and SMOTE techniques were used on the train set and we confirmed that the counts of each class were appropriately balanced based on each technique, as shown in the number of observations of each class in the table below.

technique	**X_train**	**y_train**	**Valid transactions**	**Fraud transactions**
Original Data	1481794	1481794	1474073	7721
Random Under Sampling	15442	15442	7721	7721
Random Over Sampling	2948146	2948146	1474073	1474073
SMOTE - minority	2948146	2948146	1474073	1474073

The intention behind running these 12 "out-of-the-box" models is to identify any models that stand out in terms of performance based on f1-score, precision, and recall. After selecting a smaller subset of models, our plan is to perform hyperparameter tuning to see if we can further improve performance.

To determine the efficacy of each of our 12 models, we compare their precision, recall, and f1-scores on the test set for both the valid transaction class and the fraud transaction class. These metrics are shown in the table below.

Algorithm	**Class Balancing Technique**	**Valid Class**			**Fraud Class**		
		Precision	Recall	f1-score	Precision	Recall	f1-score
Logistic Regression	Random Under Sampling	1.00	0.90	0.95	0.04	0.86	0.08
	Random Over Sampling	1.00	0.90	0.95	0.04	0.85	0.08
	SMOTE	1.00	0.91	0.95	0.05	0.85	0.09
	Set class_weight	1.00	0.90	0.95	0.04	0.85	0.08
Random Forest	Random Under Sampling	1.00	0.98	0.99	0.20	0.95	0.33
	Random Over Sampling	1.00	1.00	1.00	0.93	0.71	0.80
	SMOTE	1.00	1.00	1.00	0.78	0.80	0.79
	Set class_weight	1.00	1.00	1.00	0.96	0.64	0.77
XGBoost	Random Under Sampling	1.00	0.98	0.99	0.19	0.98	0.32
	Random Over Sampling	1.00	0.99	1.00	0.42	0.95	0.58
	SMOTE	1.00	1.00	1.00	0.74	0.83	0.79
	Set scale_pos_ weight	1.00	0.99	1.00	0.42	0.95	0.58

As we had hoped, the performance of our models generally improved compared to our initial baseline model once we applied certain class balancing techniques. A few models (XGBoost with random undersampling, random forest with random undersampling, XGBoost with random oversampling) had impressively high recall scores of 95% or greater, which means that these models were able to capture at least 95% of all the fraudulent transactions. However, these models also came with especially low precision scores (19%, 20%, 42%), which means that out of all the transactions that our model predicted as fraud, less than half were correct predictions.

Here are the three models that stood out based on their precision, recall, and f1-scores:

- Random Forest with random oversampling
- Random Forest with SMOTE
- XGBoost with SMOTE

In the next stage of our modeling, we applied RandomSearchCV to tune the hyperparameters of each model. For the random forest models, we tuned the following parameters:

- 'criterion'
- 'max_depth'
- 'max_features'
- 'min_samples_leaf'
- 'min_samples_split'
- 'n_estimators'

For the XGBoost model, we tuned the following parameters:

- 'max_depth'
- 'learning_rate'
- 'subsample'
- 'colsample_bytree'
- 'colsample_bylevel'
- 'n_estimators'

The implementation details of this step can be found in the 04_modeling_v1 notebook in the project repository. In the next section, we will present a summary of our final results, discuss the merits of our models, and identify a "best" model.

3. Findings

The table below shows the metrics of our top three models after applying hyperparameter tuning to determine if we can improve performance.

Algorithm	**Class Balancing Technique**	**Valid Class**			**Fraud Class**		
		Precision	Recall	f1-score	Precision	Recall	f1-score
Random Forest	Random oversampling	1.00	1.00	1.00	0.93	0.71	0.80
Random Forest	SMOTE	1.00	1.00	1.00	0.77	0.80	0.79
XGBoost	SMOTE	1.00	1.00	1.00	0.85	0.81	0.83

The first model we tuned (random forest with random oversampling) did not improve the metrics for the fraud class. It does have a high precision score, which indicates that 93% of the observations that the model labeled as fraud were correct. However, its recall score is approximately 10% lower than that of the other two models. Ultimately, we are looking for a balanced performance between precision and recall, which is why we are choosing to optimize the f1-score.

With that in mind, the XGBoost with SMOTE model performed the best. Based on its metrics, this model captures 81% of all fraud transactions. Additionally, 85% of the transactions that this model labeled as fraud were correct. While this precision value is not as high as the first model we tuned, we believe it offers the best balance of minimizing both false positives and false negatives, as well as giving us the best f1-score thus far.

The confusion matrix for this model is shown below in figure 11 to give us a numerical sense as to how our model is performing. Note that the valid transaction class is indicated by the label 0 and the fraud class is indicated by the label 1.

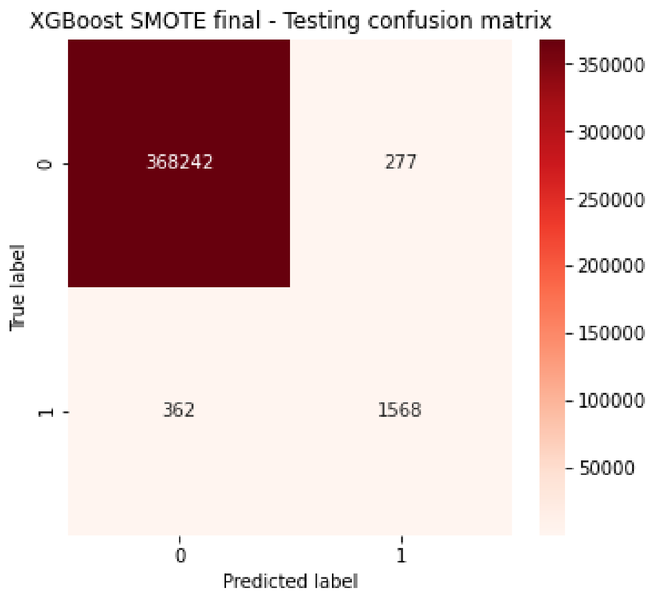

Figure 11: Confusion matrix for XGBoost with SMOTE

As we can see, out of all valid transactions, only 277 out of 368,519 actual valid transactions are incorrectly labeled as fraud. For the fraud side of the coin, only 362 out of 1,930 actual fraud transactions are incorrectly labeled as fraud.

We can shift our attention to the correct predictions of our model to get a different perspective of how our model performs. Out of the 368,519 actual valid transactions, our model accurately predicted 368,242 of them. Out of 1930 actual fraud transactions, our model accurately predicted 1568 of them.

4. Conclusions and Future Work

Needless to say, credit card fraud is a serious issue that affects the consumer, the merchant, and the issuing bank. We could very well have selected a model that can capture as many fraudulent transactions as possible, but it would have come at the cost of a significantly high number of valid transactions that are also flagged as fraud by the model.

Recognizing that false positives and false negatives are equally costly in this scenario, we took a balanced approach of trying to maximize both precision and recall in this optimization problem. By extracting and engineering various additional features from our dataset and applying a variety of models, we were able to produce a precision score of 85% and a recall score of 81%, which gives us an f1-score of 83%. We can be satisfied with these results, considering the degree to which the classes are imbalanced in this dataset.

Furthermore, by incorporating the SHAP package into our work, we were able to determine the impact of our most significant features on the likelihood of fraud. According to our XGBoost model, the following adjustments in features will *increase* the likelihood that a transaction may be fraudulent:

- Transaction amount is high
- Number of transactions in the last 24 hours is high
- Number of transactions in the last 7 days is low
- Transaction that takes place during abnormal hours (10 pm - 4 am)
- Transaction is in the gas_transport category

For future work, here is a brief list of additional work that can be done to improve the performance and the business impact of our model:

- We could have explored the ‘jobs’ column of the dataset and potentially engineer a feature based on type of job and the likelihood that a particular job may be more susceptible to credit card fraud.
- We could have explored the ‘state’ column of the dataset and potentially engineer a feature that counts the number of unique states from the last 3, 5, 10 transactions.
- We could try alternative resampling methods to balance the classes during training, such as Near Miss Undersampling and Borderline-SMOTE.
- Assuming we had additional data that provides information about the *cost* of a false negative and false positive, we can come up with a new metric that captures precision, recall, and this cost to the client (the issuing bank).
- We can use the model to generate the probability of fraud for any given transaction, and explore different thresholds for the model’s predictions.

5. Recommendations for the Clients

We would recommend that the executives of this bank utilize this model to generate the probability of fraud for future transactions. For any given transaction, it can be placed in one of three categories depending on the probability: most likely valid, most likely fraud, and requires human inspection. The thresholds for the probabilities can be analyzed in future work against the number of false positives and false negatives.

We also determined that 95% of all valid transactions are less than $190, while 75% of all fraud transactions are greater than $190. With that in mind, if a credit card charge is greater than $190, we recommend that it be reviewed by a human element for the possibility of fraud.

Additionally, if a transaction takes place during abnormal hours (10 pm - 4 am), we recommend that it be reviewed by a human element for the possibility of fraud.

6. Consulted Resources

https://www.self.inc/info/credit-card-fraud-statistics/

https://fcase.io/a-major-challenge-false-positives/

https://www.forbes.com/sites/jordanmckee/2018/11/19/three-digital-commerce-growth-opportunities/?sh=f4ffa0c38223

https://www.signifyd.com/resources/fraud-101/why-liable/