DSCI445 Term Project - Bank Account Fraud Detection

Nick Brady, Jakob Wickham, Noah Sturgeon

December 7, 2024

0.1 Introduction

0.1.1 Motivation

One important use of statistical models is protecting customer and business interests by identifying potential fraudulent applications. In these models, there must be a balance between correctly identifying the fraud (high recall) and reducing false positives.

False positives are when legitimate applications are incorrectly flagged as fraudulent. The cost of undetected fraud has a high monetary cost to the organization. A high false positive rate can create a negative customer experience while putting additional work on support teams that must handle these verifications.

With these consequences in mind, fraud detection models must be able to detect the difference while also minimizing the negative impacts accurately.

Our goal in this project is to work with various machine learning methods to find the best model based on the following metrics:

- **Precision**: Correctly identified fraudulent cases across all *classified* fraudulent cases
- **Recall**: Correctly identified fraudulent cases across all *truly* fraudulent cases
- **F1-Score**: A metric providing a balanced measure of the harmonic mean of precision and recall
- **ROC-AUC**: Model's ability to distinguish between classes, where a higher score indicates better classification performance

We are not including accuracy as part of the metrics as while it's important to see if models can correctly identify non-fraudulent data, it does not correctly reflect the project's goals.

Reference for Text: https://seon.io/resources/dictionary/false-positives/

0.2 Dataset overview

For our modeling, we will be using the Bank Account Fraud NeurIPS 2022 datasets (called BAF for short), which are a suite of synthetic datasets meant to evaluate machine learning methods. The dataset has 1 million bank account instances with 31 features and the fraud_bool response for each instance.

Variable Type	Feature Name	Description	Values/Range
Categorical	`payment_type`	Credit payment plan type	AA to AE (5 types)
	`employment_status`	Employment status	CA to CG (7 types)

Variable Type	Feature Name	Description	Values/Range
	`housing_status`	Residential status	BA to BG (7 types)
	`source`	Application source	INTERNET, TELEAPP
	`device_os`	Device operating system	Windows, macOS, Linux, X11, other
Numeric	`income`	Annual income in quantiles	0.1 to 0.9
	`name_email_similarity`	Email and name similarity	0 to 1
	`prev_address_months_count*`	Months at previous address	0 to 380, -1 = missing
	`current_address_months_count*`	Months at current address	0 to 429, -1 = missing
	`customer_age`	Age in years (rounded to decade)	10 to 90
	`days_since_request`	Days since request	0 to 79
	`intended_balcon_amount*`	Initial amount transferred	0 to 114, -1 to -16 = missing
	`zip_count_4w`	Applications in same zip (last 4 weeks)	1 to 6830
	`velocity_6h`	Apps per hour (last 6 hrs)	-175 to 16818
	`velocity_24h`	Apps per hour (last 24 hrs)	1297 to 9586
	`velocity_4w`	Apps per hour (last 4 weeks)	2825 to 7020
	`bank_branch_count_8w`	Branch apps (last 8 weeks)	0 to 2404
	`date_of_birth_distinct_emails_4w`	Distinct emails with same DOB (last 4 weeks)	0 to 39
	`credit_risk_score`	Internal risk score	-191 to 389
	`bank_months_count*`	Months of previous account	0 to 32, -1 = missing
	`proposed_credit_limit`	Proposed credit limit	200 to 2000
	`session_length_in_minutes*`	Session length on website	0 to 107, -1 = missing
	`device_distinct_emails_8w*`	Distinct emails for device (last 8 weeks)	0 to 2, -1 = missing
	`device_fraud_count`	Fraud count for device	All values = 0
	`month`	Application month	0 to 7
Binary	`email_is_free`	Free email domain	0 or 1
	`phone_home_valid`	Home phone validity	0 or 1
	`phone_mobile_valid`	Mobile phone validity	0 or 1
	`has_other_cards`	Other cards with bank	0 or 1
	`foreign_request`	Request from foreign country	0 or 1
	`keep_alive_session`	"Remember me" enabled	0 or 1

* Negative values indicate missing data for these variables.

0.3 Methodolgy

0.3.1 Data Preprocessing

Our first step was the data-cleaning process. At first glance, no columns had missing data. However, a unique attribute of the BAF is that certain numeric columns had negative values that signify missing values. Inspection of those columns was required to determine their data quality.

After that inspection, a decision had to be made about dealing with the missing data: either removing the column entirely from the model or imputing the data.

The decision on what columns had to be removed was determined by calculating the percentage of each feature with missing values and removing the columns over a threshold of 50%. As shown by the table below. Imputing data over that threshold can add bias to our model.

Imputation was applied to the columns below the threshold. Imputation is the process of filling the missing values with reasonable values. In this data set, the median value was used. For our model to know the difference between values that were imputed or not, we created an additional column _is_imputed so our model can determine if they were imputed or not.

During the initial process, we found that device_fraud_count has only 0 in its column, which did not provide any meaning, so we dropped that column from the dataset.

Reference to Text: https://scikit-learn.org/1.5/modules/impute.html

[17]:

	Percentage Missing
intended_balcon_amount	74.2523
prev_address_months_count	71.2920
bank_months_count	25.3635
current_address_months_count	0.4254
session_length_in_minutes	0.2015
device_distinct_emails_8w	0.0359

0.3.2 Exploratory Data Analysis

Since the data cleaning process was completed, we looked at the remaining columns to see if there were any clues or indicators of what could determine fraudulent applications. Before, we looked at the categorical, numeric, and binary Predictors separately. We first looked at how many fraud cases we had in the data set and what percentage, as shown below. With the number of fraud cases being so imbalanced, we will need to understand the effect of the imbalance and how to address it.

[19]:

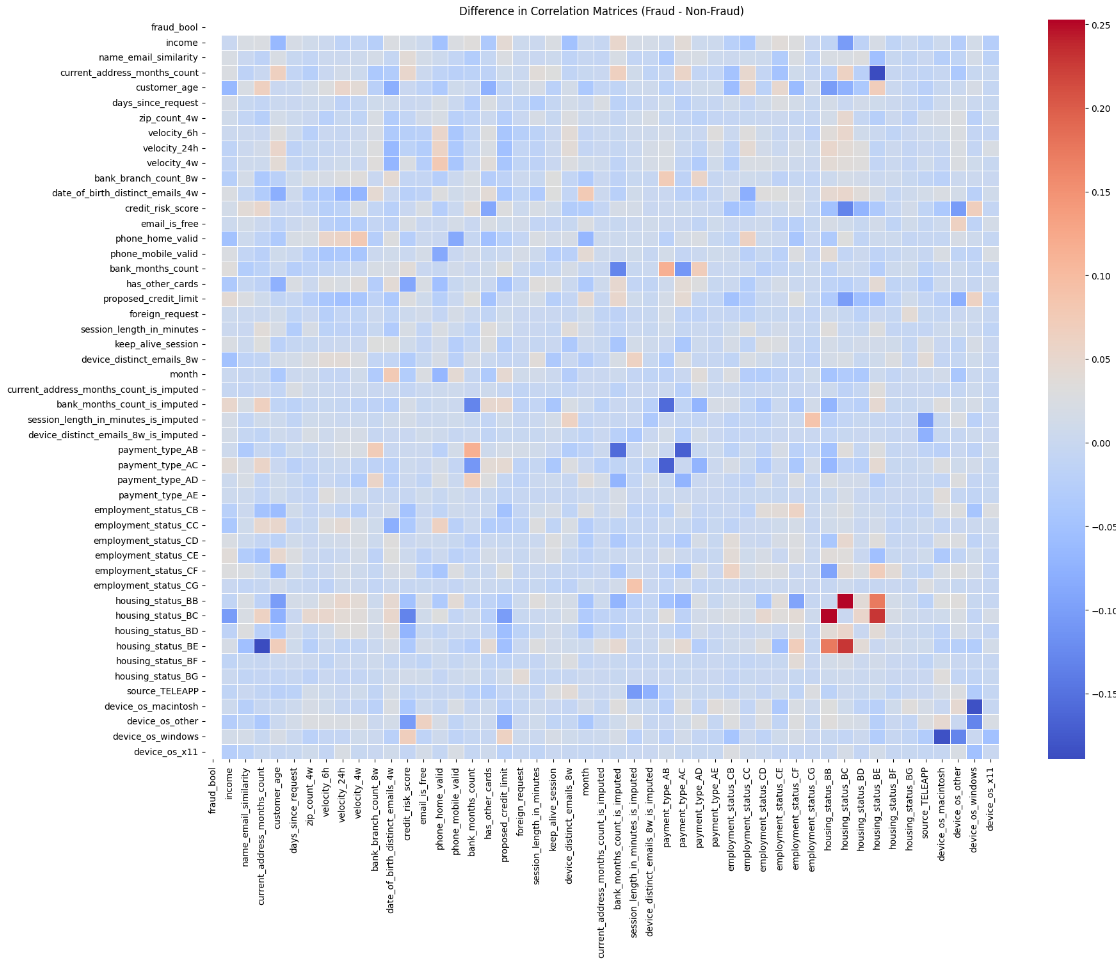
	Fraudulent Status	Count	Percentage
0	Non-Fraudulent	988971	98.8971
1	Fraudulent	11029	1.1029

Our exploratory data analysis and correlation matrix showed insights into key predictors that can determine fraud. Those insights allowed us to perform feature engineering before model selection to help our model accurately classify our objective. When we performed a summary of statistics on numeric features for fraud and non-fraud cases (See Table #1), it showed us higher fraud rates for income, proposed_credit_limit, and customer age. When we performed a Fraud Rate by numeric features binned (See Table #2), it confirmed those claims with 13% fraud rates for people requesting credit limits above $1500 and 4.2% for people over 60. When completing a Fraud

Analysis by Categorical Features (See Table #3), payment types like AC (1.75%) and BA (3.75%) were indicators of fraudulent activity. Our Analysis for Binary Features (See table #4) showed us that identity and foreign_request (2.2%) were also fraud indicators. The correlation matrix (See Table #5) confirmed our finding when looking at the predictors separately. Still, there is high collinearity due to the time series' relation to each other, and some, like days_since_request, showed weaker correlations for limited predictive potential. Initially, we created feature engineering to create high-risk flags and risk categories based on this known information, but they failed to improve model performance which will be discussed further in the analysis.

[23]:

	Feature	Count_True	Fraud_Percentage_True	Count_False	Fraud_Percentage_False
0	email_is_free	529886	1.375956	470114	0.795126
1	phone_home_valid	417077	0.669181	582923	1.413223
2	phone_mobile_valid	889676	1.054429	110324	1.493782
3	has_other_cards	222988	0.417511	777012	1.299594
4	foreign_request	25242	2.198716	974758	1.074523
5	keep_alive_session	576947	0.653093	423053	1.716333



0.4 Naive Approach

After cleaning up our dataset and doing EDA to check if there were any correlations or predictors that may stand out to help us work with our models, ultimately nothing of much use came out from it, so we decided to jump straight to model creating. The first idea we decided to check out was our "control" models: if we did nothing to handle the imbalanced data and instead throw it into some models, how would those models perform? This is the naive approach that we took a look at.

We selected 4 models for our control group: logistic regression, K-nearest neighbors, decision tree, and a random forest of 100 trees. We looked into doing more complex models such as support vector machines or possibly any unsupervised models, but with our million row and 30-something feature dataset, these models would take a long time to calculate, so we decided to have the random forest be our most complex model for our control group.

Below is the results of the naive approach alongside an interpolated recall score if the model had a 5% false positive rate (FPR).

```
                 Model  Precision    Recall  F1-Score   ROC-AUC       FPR  \
```

```
0   Logistic Regression   0.447368  0.007665  0.015071  0.872193  0.000106
5   K-Nearest Neighbors   0.150000  0.001353  0.002681  0.660778  0.000086
10        Decision Tree   0.083089  0.102344  0.091717  0.544839  0.012665
15        Random Forest   0.428571  0.001353  0.002697  0.819742  0.000020

    Recall @5% FPR
0         0.504959
5         0.294612
10        0.136288
15        0.439317
```

As expected, these models did not have good recall rates, but the false positive rates were pretty low for all of them–some of them could even be considered 0%. The only useful thing to take away from this is that last column: a higher FPR somehow results in a higher recall rate, and seeing this trend later on has caused us some trouble with our original 5% FPR restriction, leading us to scrap it. We had to deal with this recall rate / false positive rate balance throughout our project to ensure that our model was predicting fraud data as best as it could while also not incorrectly predicting fraud data frequently, which was a pain to deal with for hyperparameter tuning.

The reason behind the recall rate and false positive rate having some kind of influence on each other is simple: because our data is imbalanced, we're trying to make our models have a better chance of predicting a fraudulent transaction–which'll increase recall–but in turn, it'll increase the chance the model incorrectly predicts a non-fraudulent transaction as fraudulent, increasing the false positive rate. If we fix the FPR, the model will be more conservative and predict less fraudulent transactions, resulting in the possibility of fraudulent transactions being predicted as non-fraudulent, which is very bad.

Having a high FPR may seem scary, but in the case of fraud detection, it's actually better to have that than a lower recall rate. If a fraud detector predicts that a legitimate transaction was fraudulent (a false positive), the worst case scenario is the company loses a customer. If a fraud detector predicts that a fraudulent transaction was legitimate (a false negative), the worst case scenario is the person lost money and has to file a dispute to try and get their money back. So in our project, which deals with fraud detection, we decided that it was better risking our models having a high recall rate and FPR than trying to stay below our 5% FPR restriction and having a low recall rate because of it.

0.5 Imbalanced Data Handling

Going into this project, we knew that the only way to improve performance was to handle the imbalanced data. Fortunately, there's some techniques that we could incorporate into our models, as well as some specialized models meant for imbalanced data that we could use to try and improve performance.

0.5.1 Sampling Techniques

For our project, we used 4 sampling techniques to attempt to improve performance on our imbalanced dataset: SMOTE, ADASYN, Random Undersampling, and a combination of SMOTE and Tomek's Links.

Three of the techniques we used involved oversampling: modifying the minority class and inflating the amount of data in the minority to match that of the majority.

- SMOTE, or Synthetic Minority Oversampling Technique, attempts to improve the performance of a model by generating new observations of the minority class. The new observations are generated by selecting a point, finding its nearest neighbor, and placing a point in between the two.
- ADASYN, or the Adaptive Synthetic sampling technique, is an upgrade on SMOTE that aims to synthesize data points that are deemed "harder to learn".
- The combination of SMOTE + Tomek's Links involves first using SMOTE to inflate the dataset and then the use of the Tomek's Links algorithm to undersample the dataset by selecting pairs of observations that are the nearest neighbors to each other but of differing classes.

One of our techniques solely relied on undersampling: modifying the majority class and pruning the amount of data in the majority to match that of the minority.

- Random undersampling involves randomly taking a subset of the majority equal to the size of the minority.

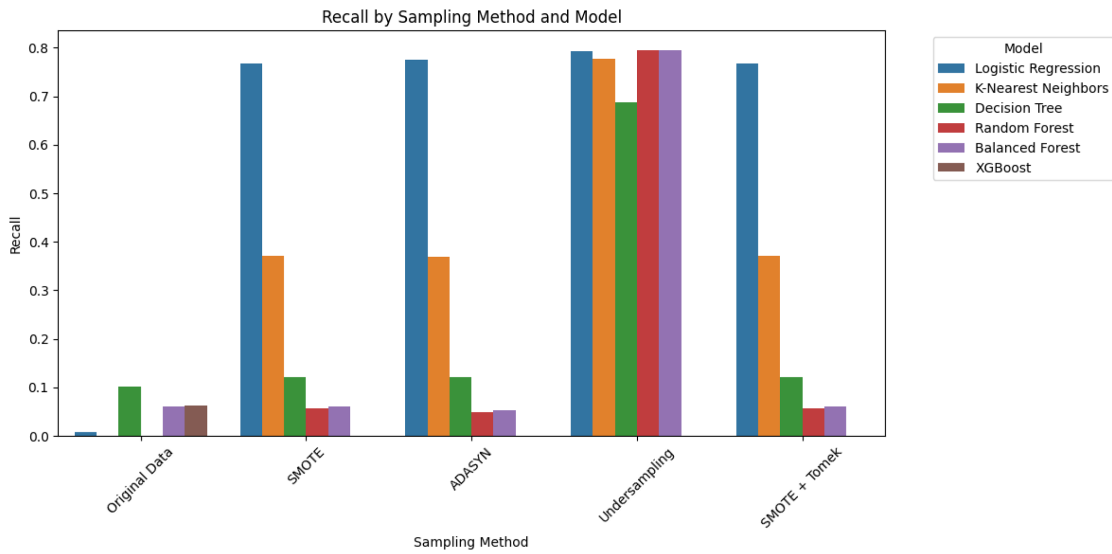
Applying these various techniques to our four starter models achieved better results compared to just using the original dataset, but before we stopped the project there, we decided to take a look at some models meant for imbalanced data and see how they compared.

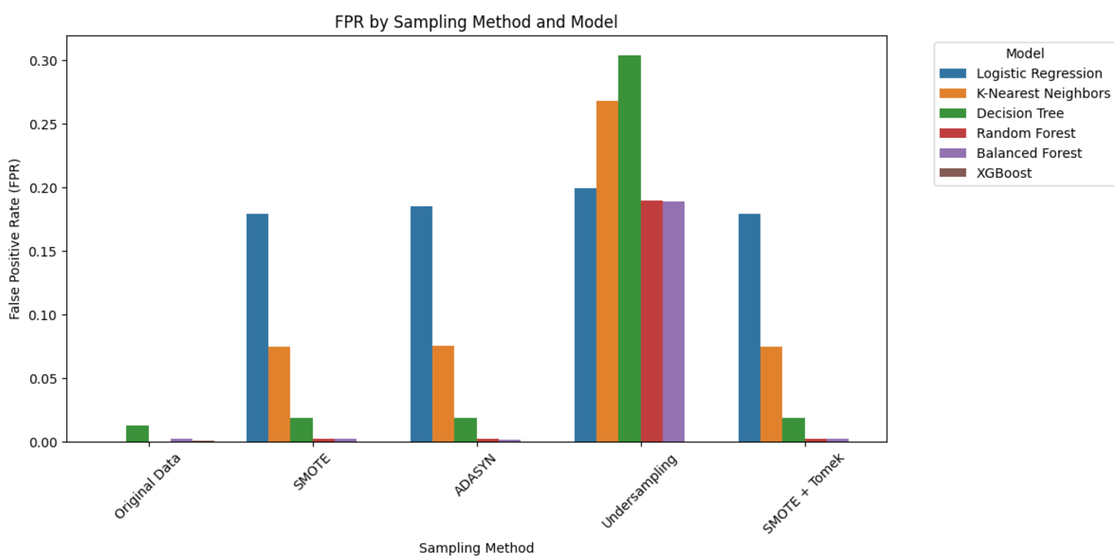
0.5.2 Imbalanced Models

For our project, we looked at 2 models that were specifically designed for imbalanced data: Balanced Forests and XGBoost

- Balanced Forests are similar to random forests, but instead of giving the whole dataset to each tree, a balanced forest undersamples the dataset and gives the result to each tree in the forest.
- XGBoost, or the Extreme Gradient Boost model, is a special kind of decision tree that uses gradient boosting and elastic regression to adjust itself after each iteration.

We ran each model (except XGBoost) with each imbalanced sampling technique to see how those models faired with a specific technique to see if a specific model performed better or worse with a specific technique. Included is a barplot of the recall rates and the FPR of each technique and the models they were performed on:





0.6 Conclusion

Compiling all of the models we used and the best sampling technique used on them, we get this final table of results:

```
                  Model Sampling Method  Accuracy  Precision    Recall  \
18        Random Forest   Undersampling  0.810425   0.044971  0.795311
23      Balanced Forest   Undersampling  0.810560   0.044956  0.794409
3   Logistic Regression   Undersampling  0.800505   0.042717  0.793508
8   K-Nearest Neighbors   Undersampling  0.732790   0.031534  0.777277
13        Decision Tree   Undersampling  0.695915   0.024752  0.688007
```

```
25              XGBoost   Original Data  0.988480   0.383152  0.063571

    F1-Score   ROC-AUC       FPR  Recall @5% FPR
18  0.085129  0.879264  0.189406        0.486192
23  0.085096  0.878432  0.189259        0.478887
3   0.081070  0.872973  0.199417        0.504058
8   0.060608  0.816793  0.267709        0.358584
13  0.047785  0.692005  0.303996        0.113160
25  0.109049  0.884227  0.001148        0.523895
```

The best sampling technique on the models turned out to be undersampling, which took us by surprise as we expected something like SMOTE or ADASYN to perform better or at least marginally better, but the graphs above showed they all did relatively worse (except logistic regression, which was pretty consistent). We believe that this may be the case due to SMOTE and ADASYN relying on specific predictors that correlate highly toward the minority class to help make the modified dataset more balanced, but due to our dataset not having much, if any at all, information that specifically related to fraudulent data, SMOTE and ADASYN didn't perform as well as expected. If anything, they performed about as good, if not worse, than random chance.

The best model performance came out to be the random forest with the undersampling technique. This, again, took us by surprise as we expected the balanced forest and XGBoost models to perform better due to them being made specifically to deal with imbalanced data. Looking at the results, XGBoost and balanced forest did better than random forest when using the original dataset, but when using undersampling, the balanced forest just barely did worse than the random forest, which makes sense. A balanced forest *is* a random forest, just with undersampling built-in.

But why does this all matter? Why did we choose the model with such high recall rate yet high FPR? As stated previously in the paper, in fraud detection, it's safer to go with more false positives that'll become inconveniences than to go with more false negatives that'll become money stolen. Our original idea was to try and find the best model and sampling technique while having the FPR less than 5%, but seeing that our best model and sampling technique, the random forest with undersampling, would have a recall rate of 48% if we kept the FPR at 5%, we decided to forgo this restriction as our vision of what the "best" model would be would ultimately not be the best model in the eyes of fraud detection.

Fraud detection is hard. There are many factors that play a role, and many of them could easily be legitimate in certain circumstances. In our project, we noted that there were no predictors that stood out as being more fraudulent than legitimate, so we threw all the predictors into our models and looked at how models would perform if every feature was used. We tried using feature engineering to see if there were any bins within predictors that stood out as fraudulent, but they were miniscule compared to everything else, so we scrapped that. We tried using correlation matrices to see if there were any relationships between predictors that stood out as being more fraudulent, but an overwhelming majority of them had little to no difference between fraudulence or legitimacy. We tried multiple models and sampling techniques with every factor, and with our current knowledge and expertise as well as with this odd dataset, this is the best we can get.