DSCI445 Term Project - Bank Account Fraud Detection

Nick Brady, Jakob Wickham, Noah Sturgeon

December 8, 2024

1 Introduction

1.1 Motivation

One important use of statistical models is protecting customer and business interests by identifying potential fraudulent bank account 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. A high false positive rate can create a negative customer experience while putting additional work on support teams that handle these verifications. The cost of undetected fraud, or false negatives, has a high monetary cost to the organization. [1]

Considering these consequences, fraud detection models must accurately detect fraud while also minimizing the negative impacts of false positives.

This project aims to identify the best model via evaluating 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

Our metrics exclude accuracy, as accuracy overvalues false negatives.

This project aims to determine the best-performing model and sampling technique based on these metrics and assess the real-life implications of deploying such a model in financial institutions.

1.2 Dataset overview

For our modeling, we used 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 reason why the BAF is a resource for testing fraud detection models is that the data set is based on present-day fraud dectection data sets and is inherendtly imbalenced with a very small percentage of the data set is fradulent. BAF has 1 million bank account instances with 31 features and the fraud_bool response for each instance. [2]

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)
	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_*	Months at previous address	0 to 380
	current_address_*	Months at current address	0 to 429
	customer_age	Age in years (rounded to decade)	10 to 90
	days_since_request	Days since request	0 to 79
	intended_balcon_*	Initial amount transferred	0 to 114
	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
	dob__distinct_emails_	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
	proposed_credit_limit	Proposed credit limit	200 to 2000
	session_length_*	Session length on website	0 to 107
	device_distinct_*	Distinct emails for device (last 8 weeks)	0 to 2
	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.

2 Methodolgy

2.1 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. We inspected those columns to determine their data quality.

After inspection, a decision had to be made about how to deal with the missing data: either remove the column entirely from the model or impute 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 with over 50% missing values. Imputing data with more missing data than that can add bias to our model. The percentage of missing values in each applicable feature is given in the table below.

Imputation was applied to the columns with less than 50% missing values. Imputation is the process of filling the missing values with reasonable values[3]. In this data set, we used the median value. For our model to know the difference between values that were imputed or not, we created an additional column_is_imputed flag so our model can determine if they were imputed or not.

During the initial data-cleaning 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.

Percentage of Missing Data in Numeric Columns

[55]:

	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

2.2 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. To start, 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.

Proportions of Fraudulent in the Dataset

[57]:

	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 fraud. 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 (1.66%) and type of operating system the user was using Windows (2.46%) 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 used 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.

Table 1: Summary of statistics on numeric features for fraud and non-fraud cases

Non Fraud Cases

[59]:

	Feature	Count	Mean	Std
0	income	988971	0.561313	0.290309
1	name_email_similarity	988971	0.494815	0.288855
2	current_address_months_count	988971	86.504746	88.231333
3	customer_age	988971	33.609125	11.989302
4	days_since_request	988971	1.025383	5.378088
5	zip_count_4w	988971	1572.138693	1005.357780
6	velocity_6h	988971	5670.664988	3010.120768
7	velocity_24h	988971	4771.528849	1479.588964
8	velocity_4w	988971	4857.444566	919.140920
9	bank_branch_count_8w	988971	184.923747	460.054059
10	date_of_birth_distinct_emails_4w	988971	9.526521	5.031063
11	credit_risk_score	988971	130.469904	69.357052
12	bank_months_count	988971	14.879864	9.964202
13	proposed_credit_limit	988971	512.303162	484.365435
14	session_length_in_minutes	988971	7.549669	8.004030
15	device_distinct_emails_8w	988971	1.018348	0.174328
16	month	988971	3.285582	2.208634

Fraud Cases

[60]:

	Feature	Count	Mean	Std
0	income	11029	0.686635	0.265579
1	name_email_similarity	11029	0.393161	0.295607
2	current_address_months_count	11029	114.869707	85.252948
3	customer_age	11029	40.858645	13.086334
4	days_since_request	11029	1.054615	5.707977
5	zip_count_4w	11029	1622.311542	1005.687071
6	velocity_6h	11029	5183.913444	2902.298679
7	velocity_24h	11029	4613.138798	1436.521551
8	velocity_4w	11029	4755.844185	975.663156
9	bank_branch_count_8w	11029	133.976426	416.350611
10	date_of_birth_distinct_emails_4w	11029	7.443195	4.848911
11	credit_risk_score	11029	177.590353	81.910348
12	bank_months_count	11029	16.475564	9.382739
13	proposed_credit_limit	11029	833.986762	643.287556
14	session_length_in_minutes	11029	8.239513	9.674728
15	device_distinct_emails_8w	11029	1.080152	0.317993

16	month	11029	3.565962	2.312055

Table 2: Fraud Rate by numeric features binned

	Feature	Bin	Count	Fraud %
0	current_address_months_count	(100.0, 150.0]	111417	2.039186
1	customer_age	(50.0, 100.0]	42660	3.469292
2	credit_risk_score	(200.0, 400.0]	170593	2.636685
3	proposed_credit_limit	(1000.0, 1500.0]	145735	2.184101
4	proposed_credit_limit	(1500.0, 2100.0]	6545	13.414820
5	session_length_in_minutes	(50.0, 90.0]	6856	2.114936
6	device_distinct_emails_8w	(1.0, 2.0]	25302	4.090586

Table 3: Fraud Analysis by Categorical Features

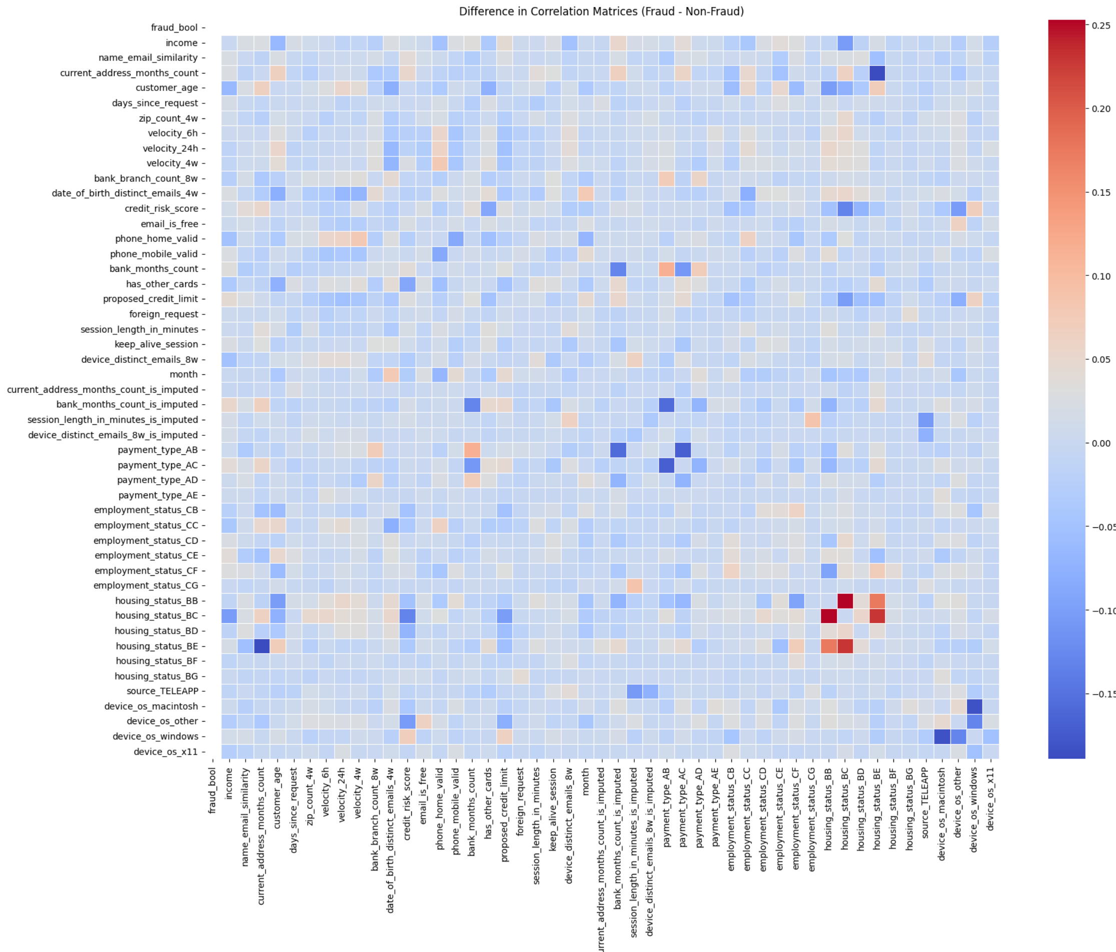
	Feature	Category	Count	Fraud %
0	payment_type	AC	252071	1.669768
1	employment_status	CC	37758	2.468351
2	employment_status	CG	453	1.545254
3	source	TELEAPP	7048	1.589103
4	device_os	windows	263506	2.469393

Table 4: Analysis of Binary Features

[63]:

	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

Table 5: Correlation Difference matrix



2.3 Naive Approach

After cleaning the dataset and conducting EDA to see if there were any correlations or predictors that may stand out to help us work with our models, we found no standout features that significantly contributed to improving our understanding of the data. As a result, we decided to proceed to the model building.

Our initial step was to establish baseline "control" models: if we did nothing to handle the imbalanced data, how would those models perform? Our naive approach served as a benchmark.

We selected four models for our control group: logistic regression, K-nearest neighbors, decision tree, and a random forest of 100 trees. We investigated making more complex models, such as support vector machines or possibly any unsupervised models. However, our million rows and 30+ features made such approaches computationally prohibitive—the random forest model was the most complex in 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).

Control Models

	Model	Precision	Recall	F1-Score	ROC-AUC	FPR	Recall @5% FPR
0	Logistic Regression	0.447368	0.007665	0.015071	0.872193	0.000106	0.504959
5	K-Nearest Neighbors	0.150000	0.001353	0.002681	0.660778	0.000086	0.294612
10	Decision Tree	0.083089	0.102344	0.091717	0.544839	0.012665	0.136288
15	Random Forest	0.428571	0.001353	0.002697	0.819742	0.000020	0.439317

As expected, these initial models performed poorly regarding recall rates, though the false positive rates were surprisingly low for all of them, some approaching 0%. The only notable trend to take away from this is that last column: a higher FPR resulted in a higher recall rate, and this trend has caused us quite a lot of trouble with our original 5% FPR restriction, so we scrapped it. Balancing recall and FPR became a recurring issue throughout the project, as we aimed to maximize fraud detection while minimizing incorrect fraud predictions.

The trade-off between recall and FPR comes from the imbalanced nature of this dataset. Increasing recall requires our model to be more aggressive in predicting fraud, which causes an increase in FPR. Fixing FPR at a low threshold forces our model to be more conservative, which creates an increase in missing actual fraudulent transactions, which is a failure in the model design.

Having a high FPR might seem problematic, but in the case of fraud detection, it's better than having a lower recall rate. A false positive, which is a legitimate transaction being flagged as fraud, results in a negative customer experience. However, a false negative, a fraudulent transaction being labeled as legitimate, can result in financial losses for both the customer and the organization, which must then go through dispute resolution. For these reasons, we prioritize models with higher recall even at a cost of false postive rates as this approach aligns with our goals for fraud detection.

2.4 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.

2.4.1 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 sample, calculating the difference between its nearest neighbors, multiplying the difference by a random number, and creating a new sample in feature space there. [4]
- ADASYN, or the Adaptive Synthetic sampling technique, is an upgrade on SMOTE that aims to synthesize data points that are deemed "harder to learn". [5]
- 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. [6]

We also used a technique of solely undersampling, called random undersampling: modifying the majority class and 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.

Next, we looked at additional models meant for imbalanced data and see how they compared.

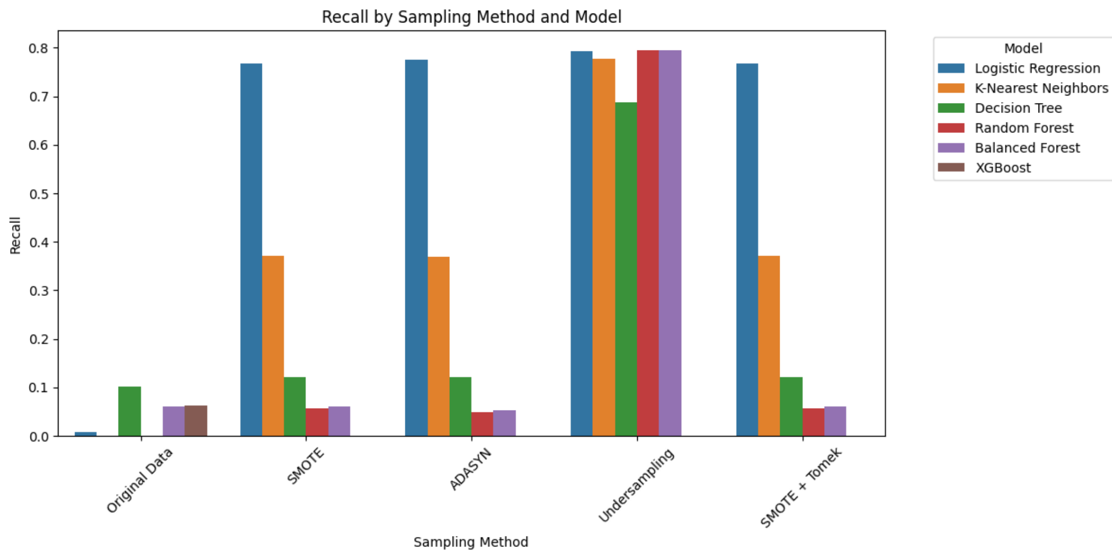
2.4.2 Imbalanced Models

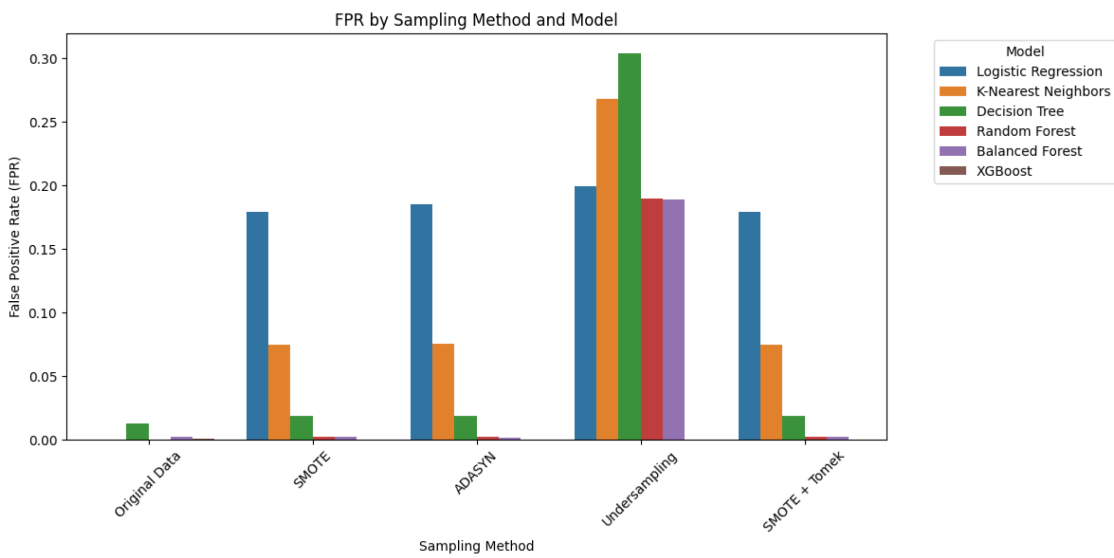
For our project, we looked at 2 models that were specifically designed for imbalanced data: Balanced Forests and XGBoost

- 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. [7]
- 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. [8]

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:

Models and their corresponding Recall and FPR by Sampling





3 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 for our model turned out to be under-sampling. We initially expected more advanced techniques, such as SMOTE (synthetic minority oversampling techniques) or ADASYN (adaptive synthetic sampling), to perform better than under-sampling. The results from the graph shown above show that these techniques generally performed worse except for logistic regressions, which remain consistent.

This outcome is tied to the way our advanced sampling techniques function. Both methods rely on strong predictors or a combination of predictors that correlate highly with the minority class, which in this case is fraudulent activity. The covariance matrix and other exploratory data analysis showed that no single predictor or a combination of predictors can determine fraudulent activity. The synthetic data sets created by those advanced sampling techniques failed to improve our model's performance and, in some cases, performed no better or even worse than random chance.

Randomly reducing the majority class worked better because it created a more balanced data set by simplifying our models' tasks. By limiting the majority's class dominance, the model could better identify patterns in the minority class without being overwhelmed by the sheer volume of majority class data. Though straightforward, this approach not only proved to be more effective given the lack of strong fraud predictors in our data set but also reduced the computational intensiveness for our project.

The best model performance was the random forest with the under-sampling 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 the balanced forest did better than the random forest when using the original dataset. Still, when using under sampling, the balanced forest barely did worse than the random forest, which makes sense. A balanced forest *is* a random forest with built-in under-sampling. If this model were used, we could determine 79.5% of all fraudulent activity with an FPR of 18.9% before tuning. Even with all these models not being tuned, it would only increase performance slightly and be computationally intensive.

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. Our vision of what the "best" model would be would ultimately not be the best model when comparing at low FPR rates.

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 used all the predictors into our models and looked at how models would perform if every feature was used. We attempted feature engineering to see if there were any bins within predictors that stood out as fraudulent, but they were miniscule compared to everything else. 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 these challenges in this dataset, random forest with under-sampling was the best model we could create.

4 References

[1] "What Are False Positives In Fraud? What Causes Them?", *SEON*, 10 Nov. 2023, https://seon.io/resources/dictionary/false-positives.

[2] "Bank Account Fraud Dataset Suite (Neurips 2022)." *Kaggle*, 29 Nov. 2023, https://www.kaggle.com/datasets/sgpjesus/bank-account-fraud-dataset-neurips-2022.

[3] "6.4. Imputation of Missing Values." *Scikit*, https://scikit-learn.org/1.5/modules/impute.html.

[4] Chawla, N. V., et al. "Smote: Synthetic minority over-sampling technique." *Journal of Artificial Intelligence Research*, vol. 16, 1 June 2002, pp. 321–357, https://doi.org/10.1613/jair.953.

[5] He, Haibo, et al. *"Adasyn: Adaptive Synthetic Sampling Approach for imbalanced learning." 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 26 Sept. 2008, pp. 1322–1328, https://doi.org/10.1109/ijcnn.2008.4633969.

[6] Viadinugroho, Raden Aurelius Andhika. "Imbalanced Classification in Python: Smote-Tomek Links Method." *Medium*, Towards Data Science, 18 Apr. 2021, https://www.towardsdatascience.com/imbalanced-classification-in-python-smote-tomek-links-method-6e48dfe69bbc.

[7] Chen, Chao, et al. "Using Random Forest to Learn Imbalanced Data." *Digital Collections*, Statistics Department at the University of California at Berkeley, Berkeley, California, Jul 2004, https://digicoll.lib.berkeley.edu/record/85556?v=pdf.

[8] "XGBoost." *GeeksforGeeks*, 6 Feb. 2023, https://www.geeksforgeeks.org/xgboost/.