



Credit Card Fraud Detection Using Machine Learning Algorithms and Labeled Data

Term Paper for IRFA Program “Data Science Software” Course

Sarbjit GILL, Saloniba RATHOD and Leticia MARRARA

Submitted: January 9, 2023

Table of Contents

Report

1. Project Topic Introduction & Related Literature	- 3 -
2. Dataset Description and Exploratory Analysis	- 4 -
3. Dataset Pre-Processing	- 14 -
4. Model Performance Evaluation Metrics	- 17 -
5. Champion Model Presentation, Tuning & Analysis	- 18 -
6. Challenger Models Presentation, Tuning & Analysis	- 21 -
7. Final Model Selection & Discussion	- 29 -
8. Conclusions & Recommendations	- 33 -

References	- 34 -
------------	--------

Appendix I: Project Jupyter Notebook File	- 37 -
---	--------

Credit Card Fraud Detection Using Machine Learning

Algorithms and Labeled Data

1. Project Topic Introduction & Related Literature

Worldwide losses from payment card fraud topped \$32 billion in 2021 per the latest Nilson Report (The Nilson Report, 2022), marking an increase of 14% over the previous year's amount. Fraud has and continues to be an important risk management issue for lenders (Bowling, 2022). McKinney (2019) shares that "a typical organization loses an estimated 5% of its yearly revenue to fraud." Thus, it is imperative that businesses continue to find, test, try, adopt and adapt diverse approaches to fraud analysis to minimize this key operational risk. One prominent area of research that has sprouted in growth over the past ten years is the use of machine learning (ML) algorithms in fraud detection (Le Borgne et. al, 2022).

As organizations continue to collect a myriad of data, it is ever-growing in size and complexity, making it harder to detect meaningful patterns by hand. It is especially true for credit card fraud detection (CCFD) as both the fraud tactics and consumer spending habits evolve overtime (Borgne et. al, 2022). In an attempt to meaningfully sift through this continuous stream of data to address the CCFD problem, several machine learning (ML) approaches have been considered in recent literature. As of the 2019 survey by Priscilla & Prabha, 26 methods have been employed by researchers to address the issue, with the top six most-frequent being: **Random Forest** (20), **Support Vector Machine** (16), Bayesian Network Classifiers (15), **Logistic Regression** (14), Decision Tree (12), and **Neural Network** (10) (Priscilla & Prabha, 2022). Our data science project will test the efficacy of four algorithms, in **bold** above, for addressing our problematic: credit card transaction fraud detection.

First, we'll begin by describing our dataset in Section 2, the unique challenges it poses and how we plan to tackle them. Section 3 will address data pre-processing and Section 4 the model evaluation metrics. Section 5 will present our champion model: random forest, and the associated analysis. Section 6 will introduce 3 challenger models to address the problem: support vector machine, logistic regression, and neural network. Section 7 will entail *Final Model Selection*, where all models will be compared—strengths and limitations—and the most appropriate selected. Finally, in Section 8, we'll conclude the project from a business point-of-view: fraud reduction, impact on revenue, recommendations, and other observations.

2. Dataset Description and Exploratory Analysis

Due to the sensitive nature of credit card transactions data, public or complete datasets are hard to find on this subject. Thus, the dataset for this project, titled “Credit Card Transactions Fraud Detection Dataset,” comes from the Kaggle.com website (Shenoy, 2020). It is a simulated dataset, using Sparkov_Data_Generation (MIT License), containing “legitimate and fraud transactions from the duration” 01-Jan-2019 to 31-Dec-2020, covering “credit cards of 1000 customers doing transactions with a pool of 800 merchants” across all customer profiles that the simulator has to offer. Given the nature of our dataset, the results derived in this project are applicable to this simulated dataset only and for an arbitrary entity we’ll name “Iron Bank” that has charged us with recommending a machine learning model specific to their organization’s data and the issue of fraud detection at hand.

The dataset comes pre-separated in two sets, containing 1,296,675 transactions in the training set (70% total) and 555,719 (30% total) in the test set. There are no missing values and the datasets contain 23 variables. A sample of our data is shared in Figure 1.

	transaction_time	credit_card_number	merchant	category	amount(usd)	first_name	last_name	gender	street_apt_number	city	...	latitude_
0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	Moravian Falls	...	
1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	Orient	...	
2	2019-01-01 00:00:51	38859492057661	fraud_Lind- Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	Malad City	...	
3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	9443 Cynthia Court Apt. 038	Boulder	...	
4	2019-01-01 00:03:06	375534208663984	fraud_Keeling- Crist	misc_pos	41.96	Tyler	Garcia	M	408 Bradley Rest	Doe Hill	...	
5	2019-01-01 00:04:08	4767265376804500	fraud_Stroman, Hudson and Erdman	gas_transport	94.63	Jennifer	Conner	F	4655 David Island	Dublin	...	
6	2019-01-01 00:04:42	30074693890476	fraud_Rowe- Vandervort	grocery_net	44.54	Kelsey	Richards	F	889 Sarah Station Suite 624	Holcomb	...	
7	2019-01-01 00:05:08	6011360759745864	fraud_Corwin- Collins	gas_transport	71.65	Steven	Williams	M	231 Flores Pass Suite 720	Edinburg	...	
8	2019-01-01 00:05:18	4922710831011201	fraud_Herzog Ltd	misc_pos	4.27	Heather	Chase	F	6888 Hicks Stream Suite 954	Manor	...	
9	2019-01-01 00:06:01	2720830304681674	fraud_Schoen, Kuphal and Nitzsche	grocery_pos	198.39	Melissa	Aguilar	F	21326 Taylor Squares Suite 708	Clarksville	...	

Figure 1: Sample preview of the dataset

After importing the data in the Jupyter Notebook, we drop the first column (row identifier) and create 7 new variables listed in Table 1 below. During our exploratory analysis, we’ll use these variables to understand their relationship with our target variable. If these new predictors are deemed useful, we may add the appropriate ones as features in our model.

Table 1: Names of and justification for new variables created

Variable Name	Datatype	Justification for Creation
hour_of_day	integer	This variable tracks the hour of the day in which a credit card transaction took place. We'll investigate this variable further during exploratory analysis to see how the fraud distribution fits over its histogram.
hour_of_day_cat	categorical	Though the ordinal nature of the 'hour of day' variable does provide a sense of time-passed relationship, however, hour 23 is also close to hour 0 and hour 1 and we lose that detail in ordinal encoding. Thus, we have transformed this integer variable into categorical by using letters of the alphabet. If the integer variable is deemed valuable after exploratory analysis, we plan to use target categorical encoding on the categorical one to retransform it to numeric and use it as a feature.
month	integer	Fraudulent transactions tend to increase during the holiday season per TransUnion's "Digital Holiday Fraud in 2022" report (TransUnion, 2022). Thus, we deem this variable is worth creating and investigating.
month_cat	categorical	Same reason as "hour_of_day_cat" above. November and December are also close to January and February and vice versa.
velocity	float	This variable tracks time elapsed (in seconds) since the previous transaction for each unique cardholder account. High velocity of payments can be a signal of fraud per Ravelin Insights (2022). We'll investigate this variable further during exploratory analysis.
age	integer	Age may play a role in predicting credit card fraud. According to the 2022 Federal Trade Commission (FTC) Consumer Sentinel Network report, the percentage of reported frauds and losses vary by age (Federal Trade Commission, 2022).
distance_from_home	float	Per Ravelin Insights, an unusual purchase location can be a signal of fraud taking place (Ravelin Insights, 2022). Thus, our new variable utilizes the latitude and longitudinal information supplied in the dataset to assess the distance between the cardholder's residence and merchant's location where the card was used. This simple transformation allows us to drop 4 columns from our dataset while preserving all the vital information in one. This results in dataset dimension reduction which will help speed up our analysis.

Before moving forward with exploratory analysis, we drop 9 variables that we do not deem of use in both the exploratory analysis and our model. 2 of these variables are datetime variables, "date_of_birth" and "unix_time", which the machine learning models cannot process but we've utilized them in creating some of the variables stated above. We also forgo the "transaction_id" variable which is the unique identifier for each transaction and is not deterministic of fraud. We also forgo 4 geographical variables: "latitude_card_holder", "longitude_card_holder", "latitude_merchant", and "longitude_merchant," as we've already used the valuable information contained in them to create the "distance_from_home" variable. Finally, we also forgo "first_name" and "last_name" columns and preserve this information in a single "full_name" categorical variable. After completing the initial transformation and cleaning, our final number of variables for exploratory analysis stands at 21, with 13 nominal categorical variables, 4 integer variables (discrete), 3 float variables (continuous), and 1 datetime variable (to be removed later). Please refer to Figure 2 for the list of variables and their data types.

#	Column	Dtype
0	transaction_time	datetime64[ns]
1	credit_card_number	category
2	merchant	category
3	category	category
4	amount(usd)	float64
5	gender	category
6	street_apt_number	category
7	city	category
8	state	category
9	zip_code	category
10	city_population	int64
11	job	category
12	is_fraud	category
13	hour_of_day	int64
14	hour_of_day_cat	category
15	month	int64
16	month_cat	category
17	velocity	float64
18	age	int64
19	distance_from_home	float64
20	full_name	category
		dtypes: category(13), datetime64[ns](1), float64(3), int64(4)

Figure 2: Variable names and data types

Upon conducting the exploratory analysis of the full dataset using ProfileReport command from pandas_profiling library, we discover the following:

- With the exception of five variables, all other nominal categorical variables in our dataset have high cardinality (see table 2 below). This poses a problem. We know that machine learning algorithms handle numeric data only but by using "One Hot

Encoding” (OHE) to encode these nominal categorical variables will vastly expand the size of our features, thus the dimensionality of our dataset, resulting in increased processing times. Also, OHE, while beneficial for algorithms such as logistic regression, affects the performance of tree-based algorithms such as random forest. Thus, to encode these high cardinality categorical variables into numeric values for machine learning algorithms to train on, we’ll employ the “regularized target encoding” approach recommended by Pargent et. al (2022). Target encoding “uses predictions of the target variable as numeric feature values” for the encoded variable, and regularization through cross-validation and parameter tuning tackles the problem of overfitting. This approach outperformed or performed at the same level as other categorical encoders in their research on a variety of machine learning algorithms.

Table 2: Categorical variables by cardinality, from the highest to lowest

Categorical Variable	Cardinality
credit_card_number	999 (high)
street_apt_number	999 (high)
full_name	989 (high)
zip_code	985 (high)
city	906 (high)
merchant	693 (high)
job	497 (high)
state	51 (high)
hour_of_day_cat	24 (moderate)
category (shopping/purchase type)	14 (slightly moderate given # of observations)
month_cat	12 (slightly moderate given # of observations)
gender	2 (low)
is_fraud (target variable)	2 (low)

- b) Our dataset is highly imbalanced (see Figure 3). Less than 1% of the credit card transactions are labeled fraudulent, precisely 0.521%, which is 9,651 transactions out of 1,852,394. Thus, we’ll need to address the issue of class imbalance before training our machine learning algorithms. However, given that our dataset has high cardinality

categorical features, we'll only be able to use oversampling, undersampling or a combination technique after categorical encoding as even the SMOTEN command only handles categorical data with 15 categories maximum.

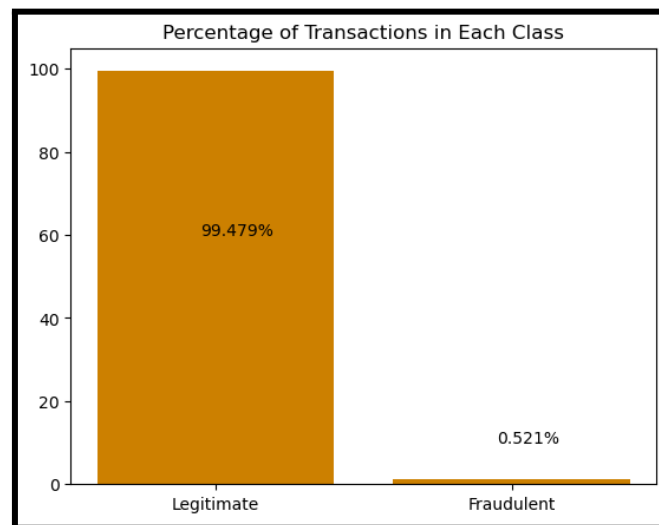


Figure 3: Percentage of transactions in each class

- c) A number of our variables are highly correlated, such as “credit_card_number” with “street_apartment_number”. Multicollinearity poses an issue for linear models such as logistic regression and support vector machines (Raj, 2020). However, models such as random forest and neural networks aren’t affected as much (De Veaux & Ungar, 1994). In order to compare the chosen models fairly, we’ll address the issue of multicollinearity after categorical encoding and re-running the correlation matrix.
- d) The variable “amount(usd)” is highly skewed. See density plot on the left below. Most transactions are small. We performed a log transformation to this variable (see plot on the right below) to make it appear more normally distributed, however our Shapiro test result indicates that the skewness will persist. Thus our models may be affected.

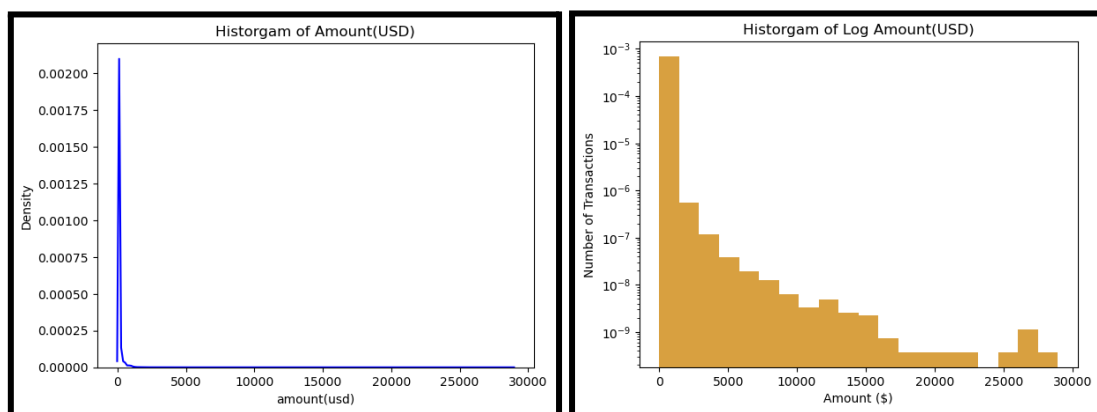


Figure 4: Amount(usd) density plot and log transformation plot

Further analyzing our dataset through the lens of our target variable: “is_fraud,” we discover the following:

- a) Fraud transactions typically range between US \$200 and \$900, with an average spend of \$531 per transaction (\$390 median), which is substantially more than the average spend per legitimate credit card transaction in our dataset: US \$68 (\$47 median).

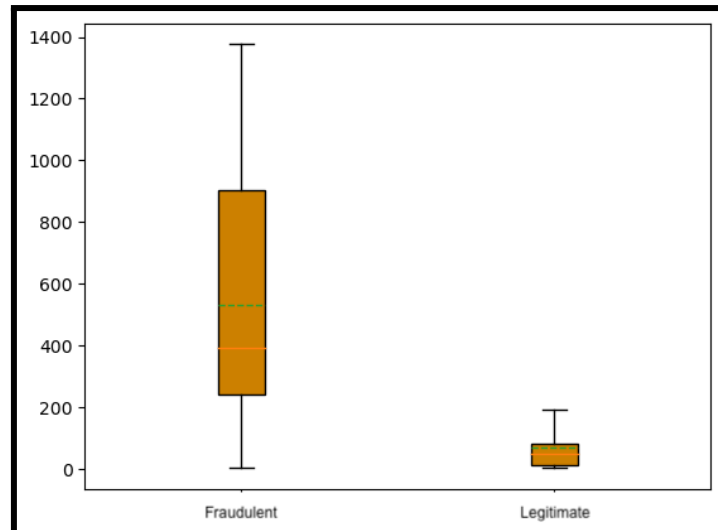


Figure 5: Average & median amounts of transactions class type with outliers removed

- b) The fraudulent transactions in our dataset mostly take place between 8 PM and 3 AM, whereas legitimate transactions follow a somewhat different pattern. Thus we'll use “hour_of_day_cat” as a feature.

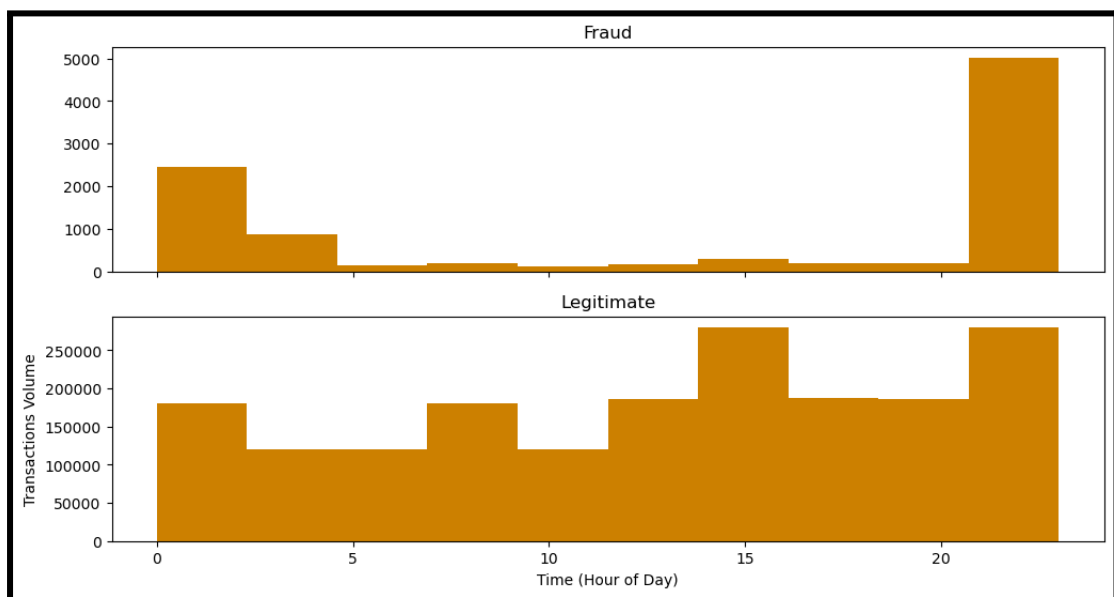


Figure 6: Legitimate and fraud transaction volume over time of day

- c) As anticipated, the volume of fraud transactions also varies by month in our dataset, with the December holiday season and early months of the year being the most popular. We'll use this variable as a feature.

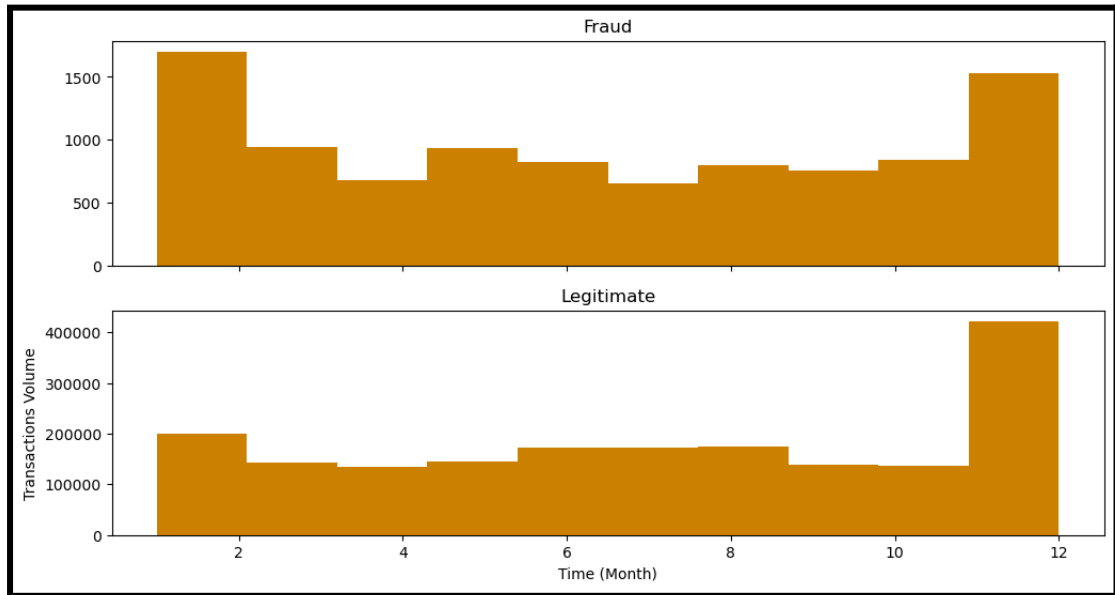


Figure 7: Legitimate and fraud transaction volume over time of year

- d) Transaction velocity also appears to be a good indicator of fraud as it can be seen below. We'll use this variable as a feature also.

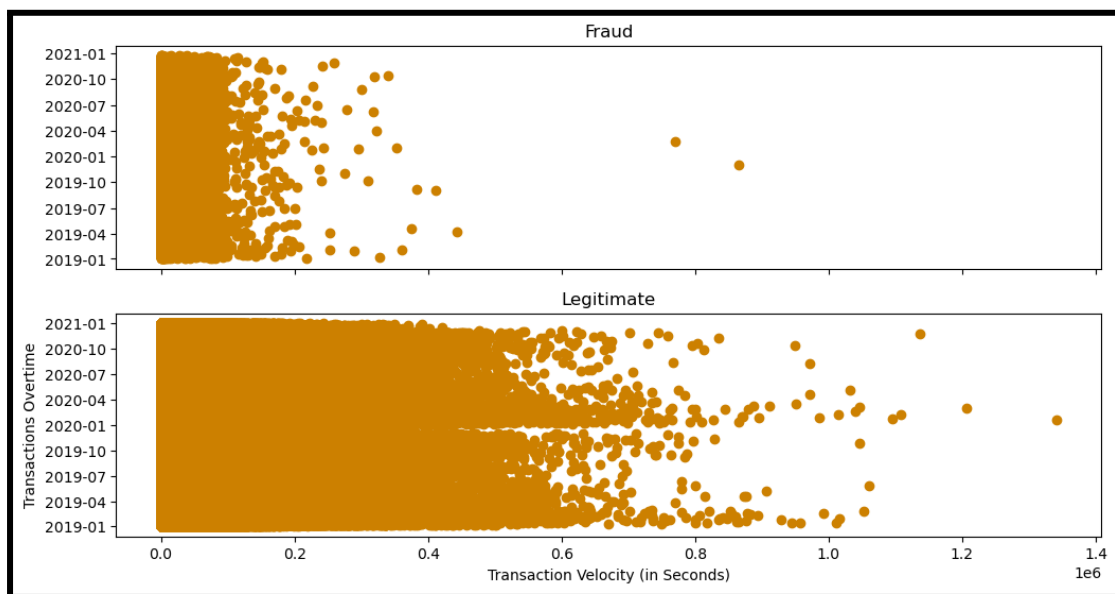


Figure 8: Legitimate and fraud transaction velocity spread

- e) Transaction distance from cardholders' residence does not appear to be a strong indicator in differentiating between fraud and legitimate transactions. We may drop this transformed variable during the feature selection stage.

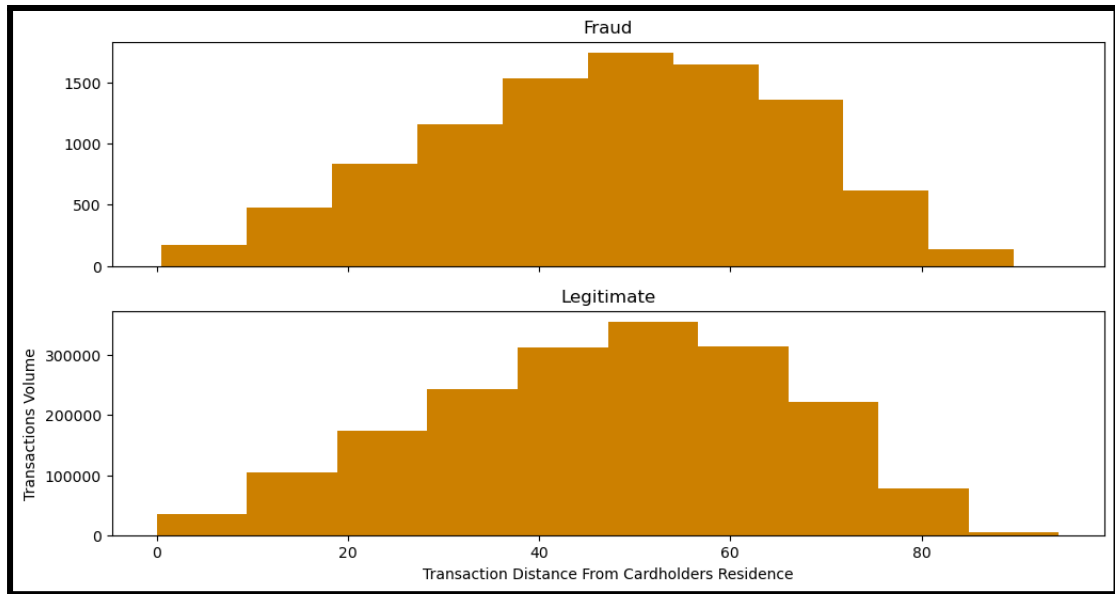


Figure 9: Legitimate and fraud transaction transaction distance spread

- f) Age may play a role in detecting fraud as the spread between fraud and legitimate transactions is different. We'll add this variable as a feature for now.

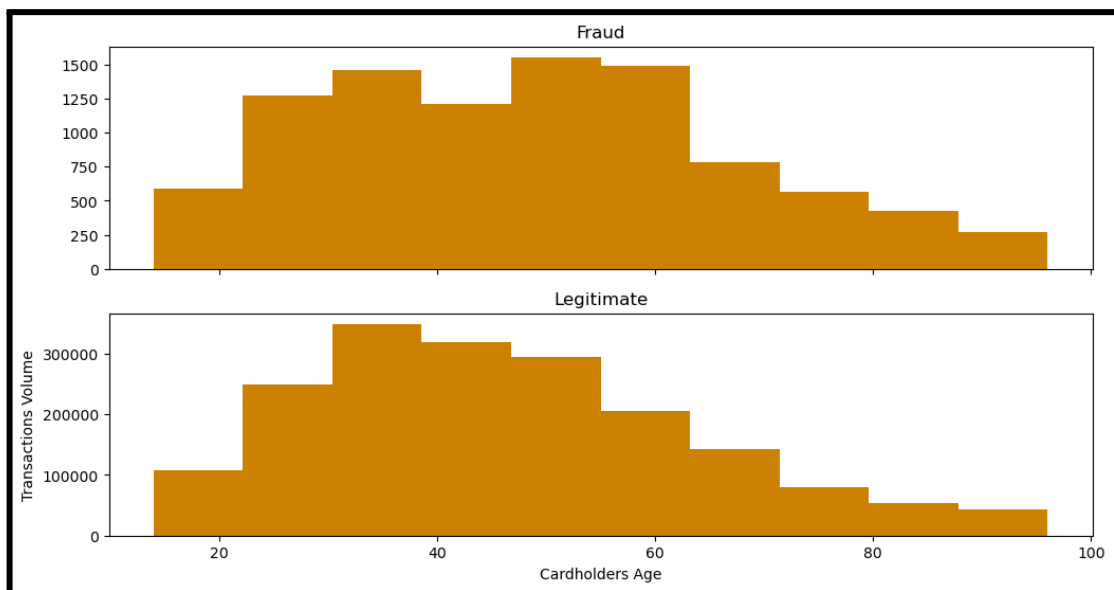


Figure 9: Legitimate and fraud transaction transaction distance spread

- g) Upon exploring the average legitimate and fraudulent expenditure per week given various shopping categories in the dataset, we found the following expenditure methods to be the most favored by fraudsters, especially internet methods: “shopping_net” (~\$1000 average spend (as)), “misc_net” (~\$800 as), “entertainment” (~\$500 as), “grocery_pos” (~\$325 as), “home” (~\$250 as), “food_dining” (~\$120 as). This variable may prove highly valuable in understanding fraud patterns.

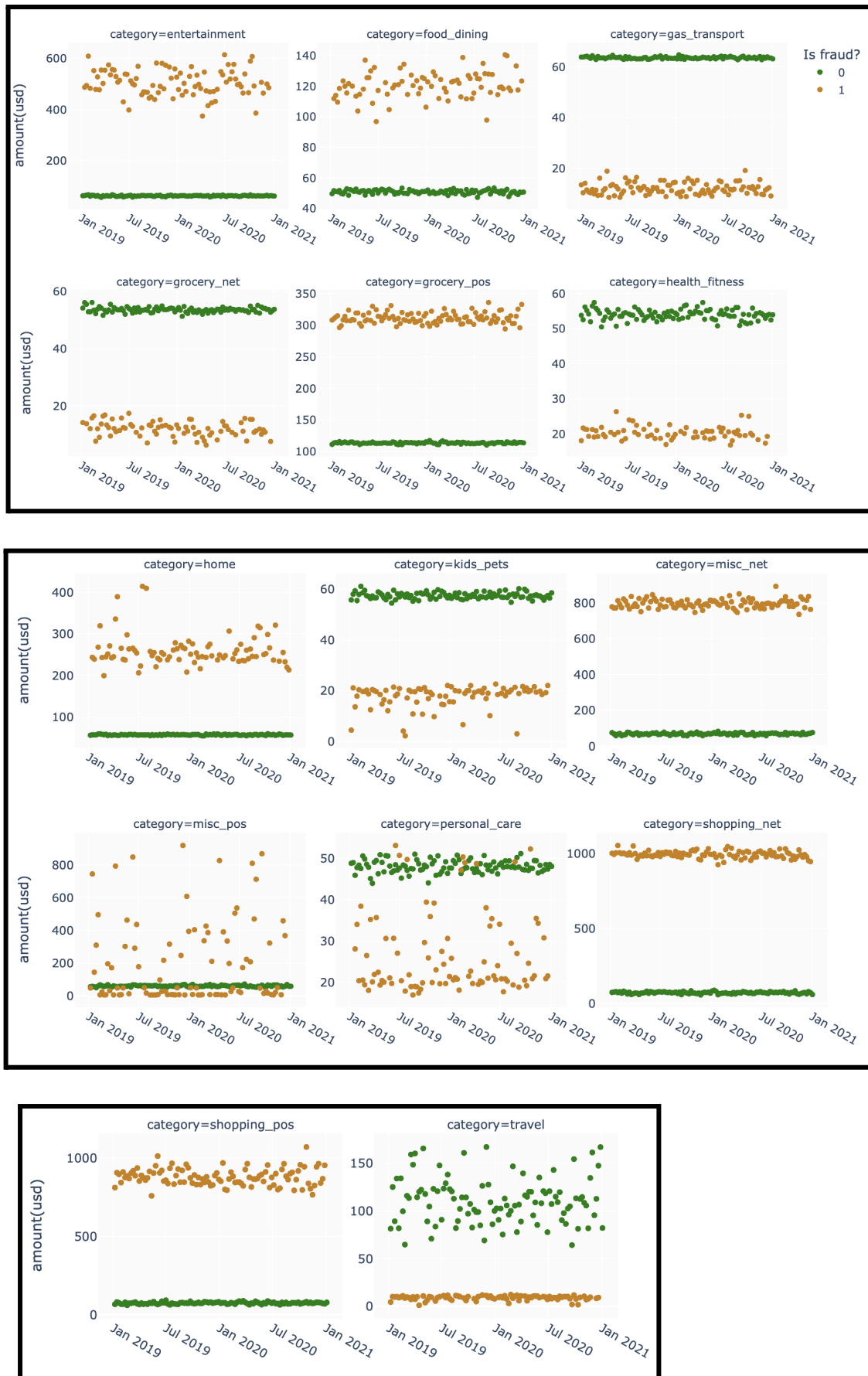


Figure 10: Weekly average legitimate and fraud expenditure by shopping categories

- h) Upon exploring our data from the viewpoint of the “merchant” variable, it appears it can be valuable in detecting fraud transactions. However, since the decision boundary data appears convoluted, it may lend to misclassification of transactions in certain cases.

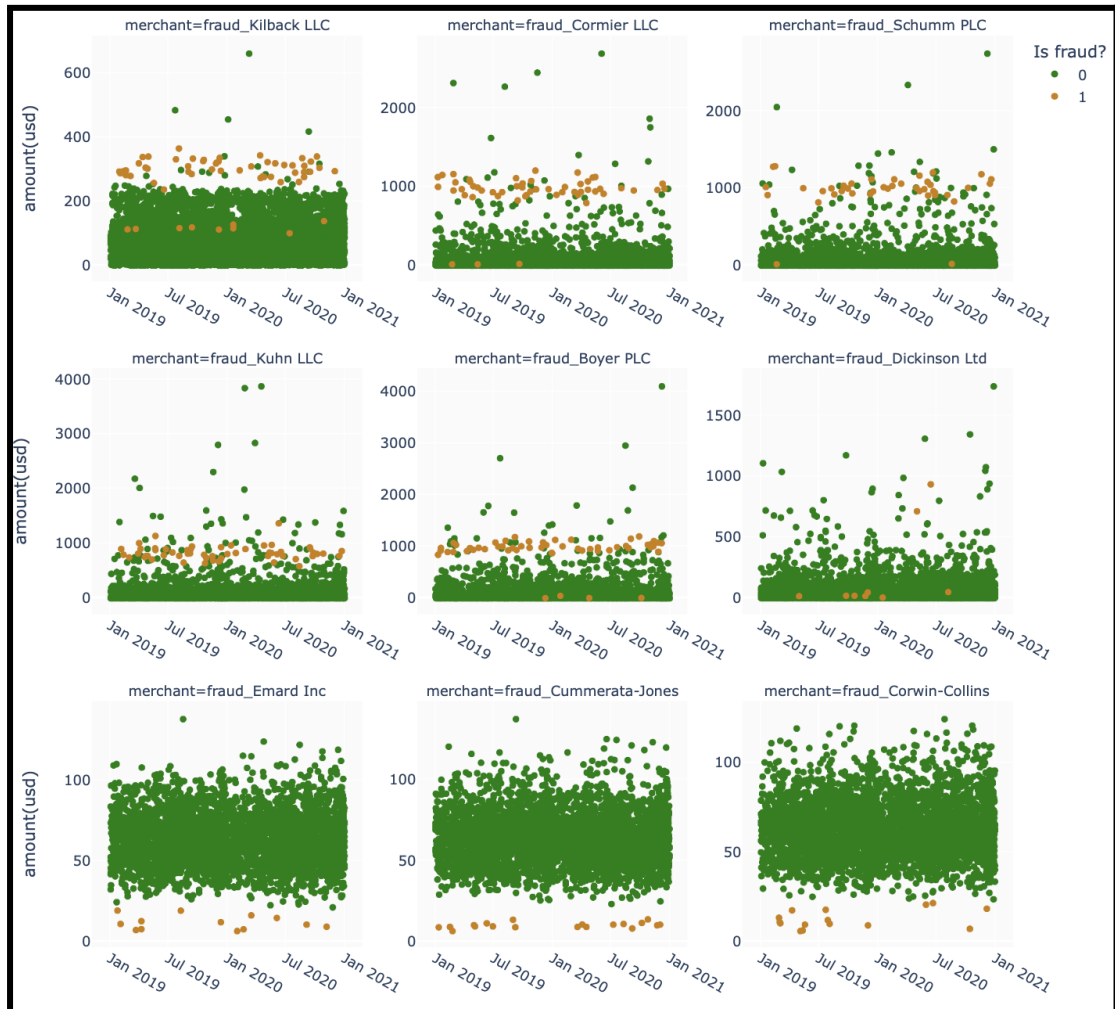


Figure 11: Top 9 merchants with the highest number of transactions per week

Having completed the exploratory analysis, we’ll now move to the data pre-processing stage where we’ll address encoding, data normality, data imbalance, and more. However, we’ll first drop the following variables: “transaction_time” (datetime variable; we’ve used it already to derive several new variables), “month” (ordinal variable; we’re keeping the nominal categorical version of this variable), and “hour_of_day” (same reasoning as “month”). This brings us down to 17 features, 1 binomial target variable, and 1.8 million observations.

3. Dataset Pre-Processing

In the pre-processing stage, we'll handle topics such as encoding, data imbalance, data normalization and scaling, and data splitting.

Encoding: First we encoded our low-cardinality nominal categorical variables. Given the number of observations in our dataset: 1.8 million, increasing features results in a drastic increase in dimensionality and affects computation and processing times. Initially our plan was to encode four variables with one-hot-encoding: “gender”, “category”, “month_cat”, and “hour_of_day_cat”, however our dataset quickly ballooned in size with some processes taking more than 2 hours to execute. Therefore, we had to make a computational/appropriate encoding tradeoff and we encoded only “gender” with one-hot-encoding. We have left a note in recommendations at the end for “the Iron Bank” to further investigate the impact of this variation in encoding as future research. Lastly, we did not drop one of the new gender variables at the risk of introducing bias. We'll let the models perform best feature selection and apply a regularization penalty to derive results.

Per the high-cardinality categorical variables, as stated above, we used the “regularized target encoding” approach, where we transformed the categorical variables into float variables based on the average target value of all data points belonging to the nominal categorical variable's category. Since this method typically results in overfitting, we've used regularization parameters to minimize this side-effect and make our results more robust. Specifically, our target encoder code is wrapped with a wrapper that performs a stratified 5-fold cross-validation test before returning values. Moreover, our encoder enclosed within also has two parameters tuned (min_samples_leaf=20, smoothing=10) to the latest recommendations by the creator of the encoder (different from the default settings) to minimize overfitting. A sample of our encoded data is visible in Figure 12.

	credit_card_number	merchant	category	amount(usd)	gender_1	gender_2	street_apt_number	city	state	zip_code	city_population	job	hou
0	0.004215	0.008804	0.006762	8.16	0.0	1.0	0.004215	0.004215	0.005202	0.004215	24840.0	0.002975	
1	0.000000	0.005435	0.003233	4.91	1.0	0.0	0.000000	0.000000	0.005025	0.000000	247.0	0.000000	
2	0.002212	0.005515	0.002677	50.65	1.0	0.0	0.002212	0.002212	0.005462	0.002212	4074.0	0.004208	
3	0.000000	0.002232	0.006762	17.48	1.0	0.0	0.000000	0.000000	0.005347	0.000000	1228.0	0.001780	
4	0.011828	0.001736	0.002548	9.29	0.0	1.0	0.011828	0.011828	0.005935	0.011828	663.0	0.007651	
5	0.000000	0.014628	0.014781	4.48	0.0	1.0	0.000000	0.000000	0.005202	0.000000	3096.0	0.000000	
6	0.019481	0.003506	0.002411	62.28	0.0	1.0	0.019481	0.019481	0.004858	0.019481	1480.0	0.010764	
7	0.003297	0.018493	0.014166	180.81	1.0	0.0	0.003297	0.003297	0.006620	0.003297	26551.0	0.003009	
8	0.001595	0.000961	0.001633	24.09	1.0	0.0	0.001595	0.001595	0.005047	0.001595	1760.0	0.004045	
9	0.000000	0.004919	0.006762	9.83	0.0	1.0	0.000000	0.000000	0.005195	0.000000	95015.0	0.002600	

Figure 12: A sample of the encoded data

Data Check - Correlation: After transforming all variables to numeric, we checked the correlation among them. A few of our predictors are highly correlated. We will not remove any variables at the risk of introducing bias, but only one column that is 100% correlated with another, thus essentially telling the same information. “credit_card_number” and “street_apr_number” fully explain each other. Thus, we dropped “street_apr_number”. We’ll keep the rest as is, and drop predictors during the feature selection stage. Now, we’ll address data imbalance.

	credit_card_number	merchant	category	amount(usd)	gender_1	gender_2	street_apr_number	city	state	zip_code	city_po
credit_card_number	1.000000	0.017613	0.018888	0.059259	-0.064782	0.064782	1.000000	0.933587	0.159877	0.990958	-0.007650
merchant	0.017613	1.000000	0.914390	0.090876	0.008801	-0.008801	0.017613	0.015538	0.000947	0.017325	-0.000248
category	0.018888	0.914390	1.000000	0.099159	0.009797	-0.009797	0.018888	0.016359	0.001562	0.018616	-0.000386
amount(usd)	0.059259	0.090876	0.099159	1.000000	-0.001631	0.001631	0.059259	0.052086	0.002811	0.058504	0.028887
gender_1	-0.064782	0.008801	0.009797	-0.001631	1.000000	-1.000000	-0.064782	-0.064943	-0.033454	-0.068264	-0.053428
gender_2	0.064782	-0.008801	-0.009797	0.001631	-1.000000	1.000000	0.064782	0.064943	0.033454	0.068264	0.053428
street_apr_number	1.000000	0.017613	0.018888	0.059259	-0.064782	0.064782	1.000000	0.933587	0.159877	0.990958	-0.007650
city	0.933587	0.015538	0.016359	0.052086	-0.064943	0.064943	0.933587	1.000000	0.146409	0.942113	-0.027408
state	0.159877	0.000947	0.001562	0.002811	-0.033454	0.033454	0.159877	0.146409	1.000000	0.161011	-0.000096
zip_code	0.990958	0.017325	0.018616	0.058504	-0.068264	0.068264	0.990958	0.942113	0.161011	1.000000	-0.000449
city_population	-0.007650	-0.000248	-0.000386	0.005676	0.028765	-0.028765	-0.007650	0.017500	-0.027408	-0.007533	1.000000
job	0.596634	0.007546	0.007453	0.028887	-0.053428	0.053428	0.596634	0.555543	0.135294	0.583511	0.000000
hour_of_day_cat	0.030844	0.007324	0.007996	0.030927	-0.000494	0.000494	0.030844	0.026867	0.002197	0.030376	0.000000
month_cat	0.007380	0.000594	0.000807	0.002617	-0.000063	0.000063	0.007380	0.006648	-0.000096	0.007323	-0.000000
velocity	0.179944	0.001445	0.002380	-0.003265	-0.056019	0.056019	0.179944	0.177658	0.045134	0.177779	0.000000
age	0.082115	-0.000295	-0.000616	-0.010020	-0.004976	0.004976	0.082115	0.075753	0.052585	0.081069	-0.000000
distance_from_home	-0.000199	0.001843	0.001841	-0.001123	0.002118	-0.002118	-0.000199	-0.000458	-0.010871	-0.000449	0.000000
full_name	0.992743	0.017449	0.018578	0.058325	-0.063190	0.063190	0.992743	0.928764	0.145732	0.983659	-0.000000

Figure 13: A sample of correlation among all variables post encoding

Data Imbalance: Most machine learning classification algorithms and neural networks rely on the assumption of balanced distribution of class labels (Abd Elrahman & Abraham, 2013). Thus, it is recommended to balance your dataset via oversampling, undersampling or a combination of the two techniques and record model performance with and without class imbalance correction. The technique we are using is called SMOTE: Synthetic Minority Over-sampling. It oversamples the minority class to produce synthetic minority class samples by first choosing a minority class data point, then using the k-nearest neighbors (kNN) method it finds other minority class data-points nearby, resulting in samples lying on the lines connecting the chosen minority point to some or all of its neighbors. We had also looked into using Tomek links to lightly undersample our majority class so that the decision boundary that SMOTETomek would produce would be much clearer as it removes observations that may convolute machine learning (Johnson, 2021). However, our dataset is too large and the process requires hefty computational power. Thus, we’ve attempted to

balance our class labels via SMOTE only and left the other method as a recommendation for the Iron Bank for future research. After the transformation, we now have an equal 966,915 class members in the training set. We'll compare the results of both SMOTE-transformed data and imbalanced data.

Data Normalization and Scaling: Given that some of our variables, such as “amount(usd)” were heavily skewed, to pre-process this data to reduce impact of outliers, we've opted for SciKit's Robust Scaler. As the documentation states: “Outliers can often influence the sample mean / variance (Standard Scaler approach) in a negative way. In such cases, the median and the interquartile range often give better results.” (Scikit, 2023)

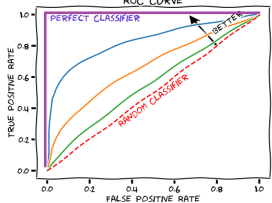
Data Splitting: Our data is split into three sets: training, validation, and test. Our test set is the same as the one we've imported and we have applied the same encoding transformation to it as our other sets. We have further split our initial training set into a new training set and a validation set to fit, predict performance and tune our models. We used scikit-learn's train_test_split command (70/30 split) to accomplish this objective. We will use our test set (30% of full dataset) only at the end to perform model selection.

Now that we have pre-processed our data, we are ready for testing it on our chosen models to understand their performance. First, however, we will quickly introduce the metrics we'll use to assess our models.

4. Model Evaluation Metrics

Given that we are addressing the issue of fraud detection, we will consider the following metrics when evaluating our machine learning models.

Table 3: Model evaluation metrics in relevance to our problem

Metric	Formula	Interpretation for Our Problem
Recall	$\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$	$\frac{\text{\# of Correctly Identified Fraudulent Transactions}}{\text{Total \# of Fraudulent Transactions in the Validation/Test Set}}$ <p>This metric evaluates the rate of fraud detection by our model. It will be our primary metric that we will always aim to maximize (thus aim to reduce false negatives), since, on average, each marginal loss in recall percentage is associated with US \$51,247 (\$37,639 median)* in fraud transactions in our dataset.</p>
Accuracy	$\frac{\text{True Positives} + \text{True Negatives}}{\text{All Cases}}$	$\frac{\text{Correctly Identified Transactions (Fraud and Legitimate)}}{\text{Total \# of Transactions in the Validation/Test Set}}$ <p>On average we can achieve 99.48% accuracy by simply classifying all our transactions as legitimate. Therefore, this metric is beneficial in understanding how accurately our model is classifying the two classes. The more the legitimate transactions are labeled fraudulent the more they will need to call the Iron Bank to unfreeze their accounts.</p>
AUROC		Area under the receiver operating characteristic tells us about the power of our model in discriminating between two classes: the probability that our model is classifying a random fraudulent transaction as fraudulent versus a random legitimate transaction as fraudulent. This metric is of value but we want to primarily maximize recall and minimize dollars lost.
Precision	$\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$	$\frac{\text{Correctly Identified Fraudulent Transactions}}{\text{Total \# of Transactions Identified as Fraudulent by Model}}$ <p>This metric is of less value to us as our primary goal is fraud detection. We want to minimize false negatives over false positives while trying to be accurate. Given that less than 1% of our overall transactions are fraudulent, a low precision score isn't daunting.</p>
F1	$\frac{2 * \text{Recall} * \text{Precision}}{\text{Precision} + \text{Recall}}$	This metric looks at the relationship between recall and precision. A high score means our model can both identify fraud and do so precisely so that legitimate transactions are not labeled as fraudulent. This metric is of value, however, we may have to sacrifice precision to maximize our recall rate.

* Calculation: 9,651 fraudulent transactions in full dataset x 0.01 miss rate x \$531 avg/ \$390 median fraud value

5. Champion Model Presentation, Tuning & Analysis

In their 2019 survey, Priscilla et al. (2019) found that “**random forest**” was the most popular machine learning model employed by researchers to address the issue of credit card fraud detection. Thus, we have chosen this tree-based learning method as our champion model.

How does it work? Random forest is built upon the decision tree model which takes one down the classification path through a series of questions or attribute information. Where random forest proves valuable is that it randomly samples features for individual decision trees to be built in the digital forest and bootstraps from the observations. Averaging out these results from a large number of uncorrelated trees, it builds a more bias-free and accurate model that does not overfit (TIBCO, 2023). However, model interpretability can prove difficult due to the way the model is constructed by the algorithm.

Random Forest Models For Detecting Credit Card Fraud At The Iron Bank - Baseline Results: After running our preprocessed data through scikit-learn’s RandomForestClassifier algorithm, we found that our model with synthetically balanced data performed poorly (see table 4 and figures below), and the unbalanced-data model shows promise but requires further tuning. It currently predicts 7 out of every 10 fraudulent transactions when tested on our validation set. However, given that the average expenditure of a fraudulent transaction in our dataset is USD \$531 (median \$390), with this model over the span of the time in our dataframe (2 years), we would’ve missed \$1.5 million dollars worth of fraudulent transactions on average (\$1.1 million using median value).^{*} Thus, we must attempt to improve this baseline fraud prediction score through feature selection and hyperparameter tuning of our models.

Table 4: Baseline random forest model results

	Recall	Accuracy	F1	Precision	AUROC	RP Curve
Without SMOTE	71%	100%	82%	96%	86%	69%
With SMOTE	48%	35%	1%	0%	42%	1%

^{*} Calculation: 9,651 fraudulent transactions in full dataset x 0.29 miss rate x \$531 avg/ \$390 median fraud value

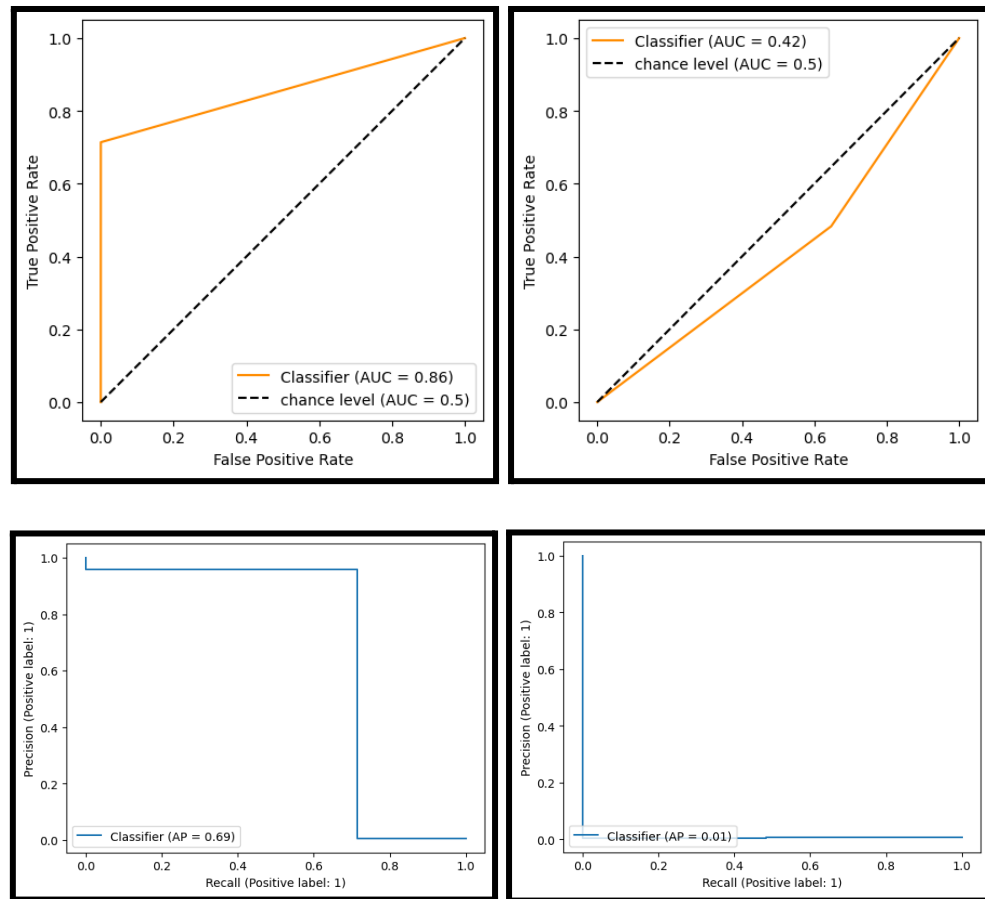


Figure 14: Baseline random forest model results (Left: Unbalanced, Right: SMOTE)

Results after features selection and hyperparameter tuning: We employed the Boruta algorithm to help us perform feature selection for random forest models. It rejected 6 features from our unbalanced-data random forest model, which we then dropped: "gender_1", "gender_2", "state", "city_population", "month_cat" and "distance_from_home". And the algorithm suggested no variables be dropped from the synthetically-balanced data model. Next, we performed hyperparameter tuning with cross-validation using scikit's GridSearchCV command, again with the aim to maximize the "recall" score. We struck gold. Our refined model with unbalanced data now detects 19 out of every 20 fraudulent transactions (see the table and figures below), thus minimizing the average fraud stated above to US \$252,234 (\$188,195 median) only over 2 years, while classifying almost the same number of transactions (19 out of 20) correctly. The refined model is not perfect, but it is an incredible improvement over baseline results:

```
RandomForestClassifier(class_weight='balanced', max_depth=4, max_features='sqrt',
min_samples_split=5, n_estimators=60, n_jobs=-1, random_state=4)
```

On the other hand, despite the hypertuning of the parameters of the synthetically-balanced data model to maximize recall through cross-validation, the recall score of 99% on the fitted model failed to generalize to our validation set, resulting in no improvement in the model results.

Table 5: Latest random forest model results

	Recall	Accuracy	F1	Precision	AUROC	RP Curve
Without SMOTE	95%	94%	15%	8%	94%	8%
With SMOTE	48%	35%	1%	0%	42%	1%

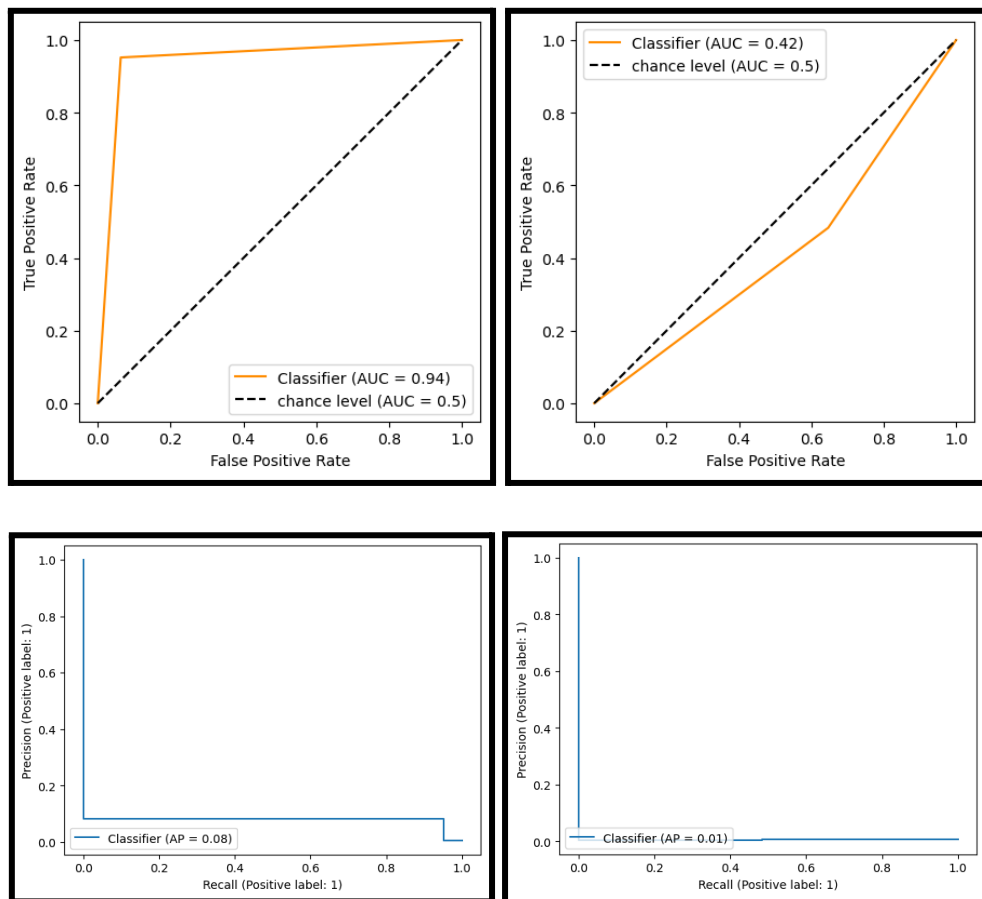


Figure 15: Latest random forest model results (Left: Unbalanced, Right: SMOTE)

Discussion: The refined unbalanced-data model inclusive of feature selection and hyperparameter tuning shows promising results in detecting credit card fraud transactions for the Iron Bank while maintaining a high level of accurate classification. Moreover, it is a near-perfect classifier from a data science perspective. We will evaluate this model on the held-out test set when performing final model selection.

6. Challenger Models Presentation, Tuning & Analysis

Three other methods among the top six in Priscilla et al.'s (2019) survey were: support vector machine, logistic regression and neural network. These three will serve as our challenger models to the champion model. We'll present each and discuss the results generated.

6.1 Support Vector Machine Machine Learning Model

How does it work? Support Vector Machine (SVM) is a supervised machine learning algorithm that performs classification, regression and detection of outliers by simply drawing a straight line between two classes (Maklin, 2019). All the given data points that fall on one side of the line will be labeled as one class and the rest of points labeled as the second. Given that there are infinite such lines, the "support" in "support vector machine" are position vectors that decide the decision boundaries. For ease of use with our large dataset, we've opted for scikit-learn's SGDClassifier, which is a linear SVM classifier with stochastic gradient descent training.

SVM Models For Detecting Credit Card Fraud At The Iron Bank - Baseline Results: Baseline results for one of our models trained on balanced data is rather impressive. On average this model detects 97 out of every 100 fraudulent transactions. However, the model suffers when it comes to accuracy, classifying only 46% of all of the transactions appropriately. Thus, customers will likely get agitated if their accounts keep getting frozen and they are unable to use their cards. Agitated clients may leave the bank for a competitor if the misclassification issue persists. Thus, we will aim to refine and find a better model that classifies transactions more accurately while maintaining a high rate of fraud detection.

Table 6: Baseline SVM model results

	Recall	Accuracy	F1	Precision	AUROC	RP Curve
Without SMOTE	56%	90%	6%	3%	73%	2%
With SMOTE	97%	46%	2%	1%	71%	1%

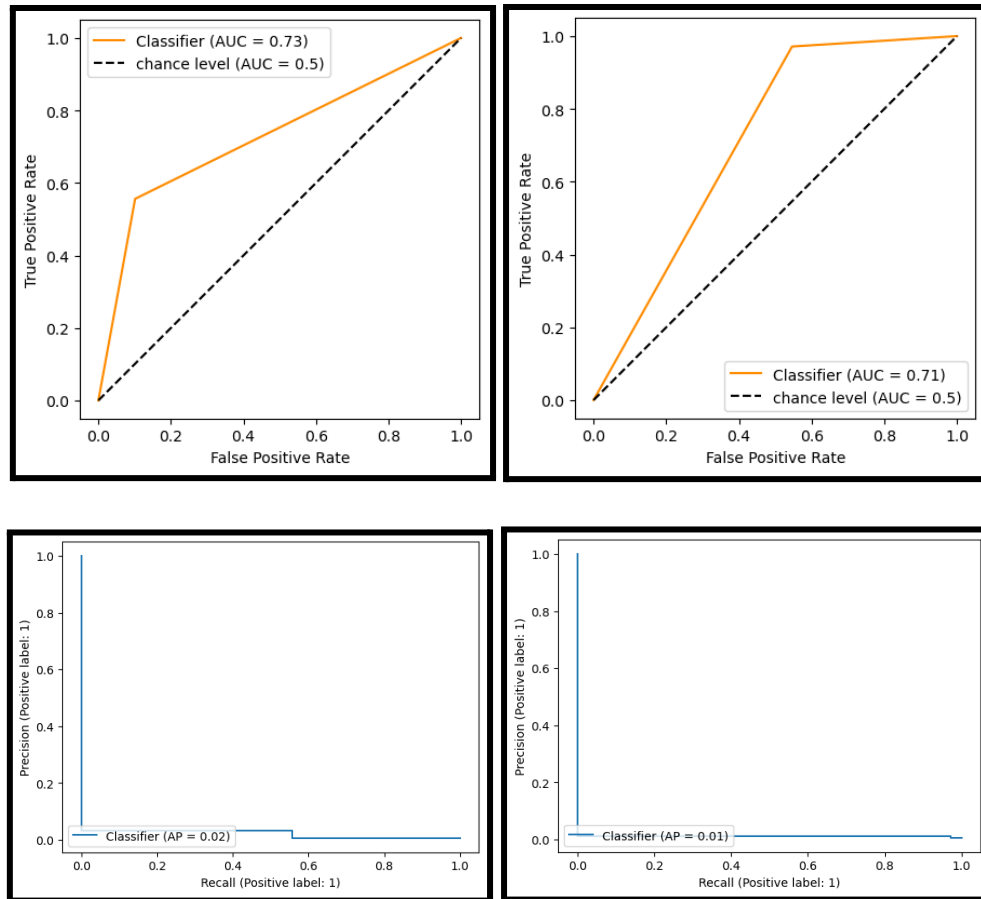


Figure 16: Baseline SVM model results (Left: Unbalanced, Right: SMOTE)

Results after features selection and tuning the hyperparameters: For feature selection for each of the SVM models, we first performed recursive feature elimination with cross-validation using scikit_learn's RFECV command. We coded the command to maximize the "recall" score. Only the optimal features selected by the recursive algorithm were kept. For the unbalanced-data SVM model, we dropped three features: 'state,' 'city_population,' and 'distance from home', and from the balanced-data SVM model we dropped five: "gender_1", "gender_2", "state", "zip_code", and "city_population". After hypertuning individual model parameters using the same technique as the champion model, we ran the refined models. The results improved, but at the expense of accurate classification, indicating the results are overfitting. The best improvement was in our unbalanced-data model whose recall improved to 93% from 56%, but with a drop of accuracy to 76% from 90%. And though our balanced data model detects fraud at a very high rate, at 98%, it suffers from poor accurate classification, classifying only 45% of the transactions accurately.

Table 7: Latest SVM model results

	Recall	Accuracy	F1	Precision	AUROC	RP Curve
Without SMOTE	93%	76%	4%	2%	85%	2%
With SMOTE	98%	45%	2%	1%	71%	1%

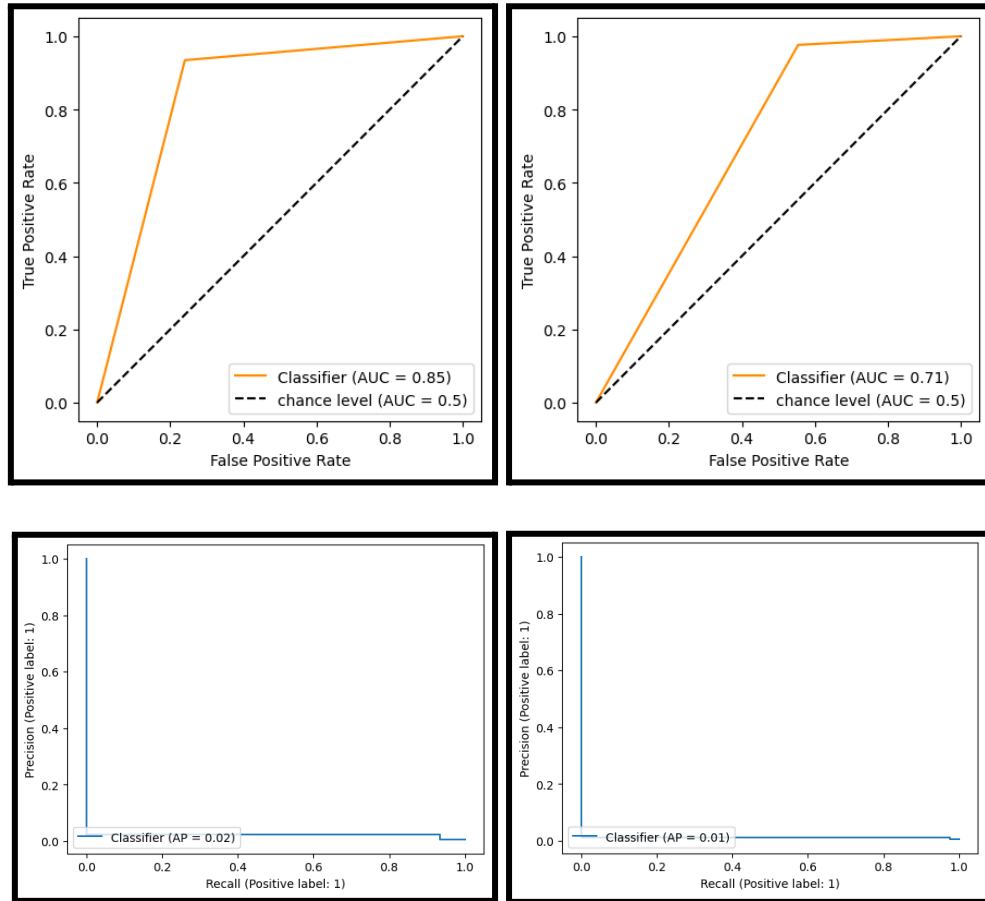


Figure 17: Latest SVM model results (Left: Unbalanced, Right: SMOTE)

Discussion: Given the tradeoff between recall and accuracy, we will choose the refined unbalanced-data SVM model with 14 features for the final model selection stage. It is also a better classifier from a data science model perspective with an AUC score of 85%.

6.2 Logistic Regression Machine Learning Model

How does it work? Logistic Regression is a machine learning algorithm that can be used for classification and in predictive analytics. When the dependent variable is categorical, such as ‘is_fraud’ in our dataset, the algorithm takes in the predictor inputs to regress upon

and supplies us a final answer between two or more possible outcomes (Swaminathan, 2018). Akin to linear regression, logit relies on the assumptions of: a) independence of errors, b) linearity for continuous variables, c) absence of multicollinearity, and d) lack of influential outliers. We'll aim to address these issues in the refinement of our baseline models

Logistic Regression Models For Detecting Credit Card Fraud At The Iron Bank - Baseline Results: Alike SVM, our baseline model with balanced data shows remarkable fraud detection performance, detecting 97 out of every 100 fraudulent transactions (see tables and figures below), when we run it through scikit-learn's LogisticRegressionClassifier. However, it is at the expense of accuracy as the model is misclassifying more than 50% of legitimate transactions as fraudulent. We'll further refine the models presented below.

Table 8: Baseline logistic regression model results

	Recall	Accuracy	F1	Precision	AUROC	RP Curve
Without SMOTE	32%	100%	45%	74%	66%	24%
With SMOTE	97%	48%	2%	1%	72%	1%

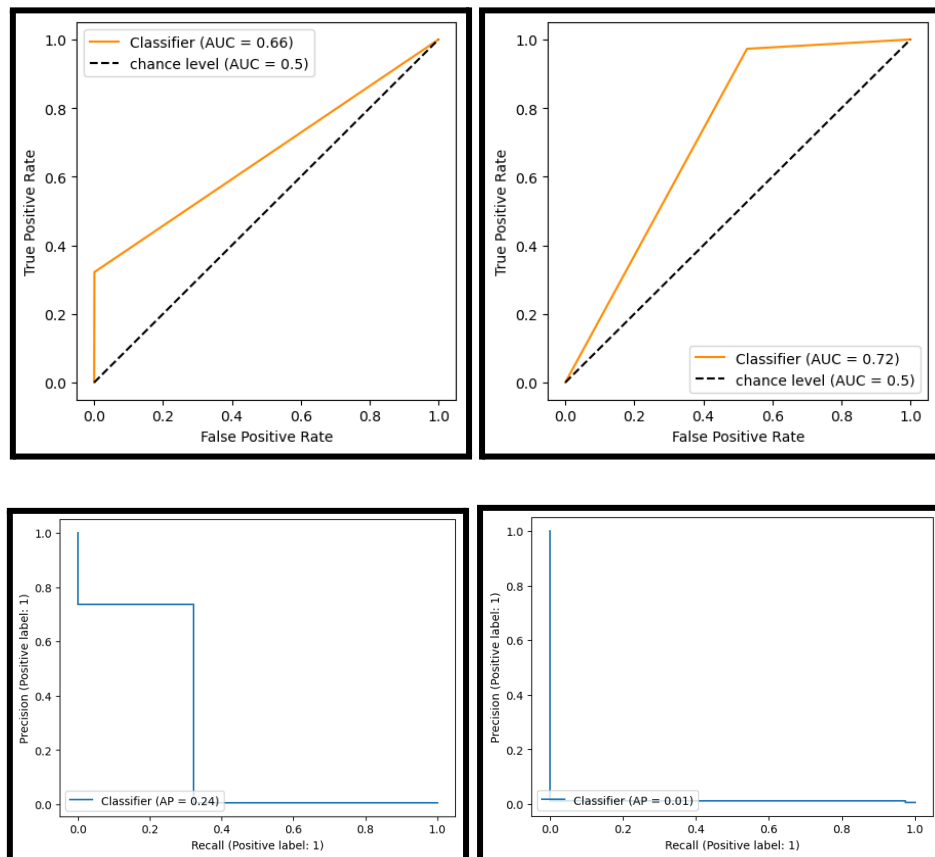


Figure 18: Baseline logistic regression model results (L: Unbalanced, R: SMOTE)

Results after features selection and hyperparameter tuning: After performing recursive feature elimination with cross-validation, both of our models improved from their baseline results. Our final model for unbalanced-data contains 14 features (3 eliminated) and the balanced-data model contains only 5 features. The former detects nearly 9 out of every 10 fraudulent transactions and correctly classifies 85% of the transactions. The latter has high recall, detecting 97 out of every 100 fraudulent transactions but it wrongly classifies data 51% of the time.

Table 9: Latest logistic regression model results

	Recall	Accuracy	F1	Precision	AUROC	RP Curve
Without SMOTE	88%	85%	6%	3%	87%	3%
With SMOTE	97%	49%	2%	1%	73%	1%

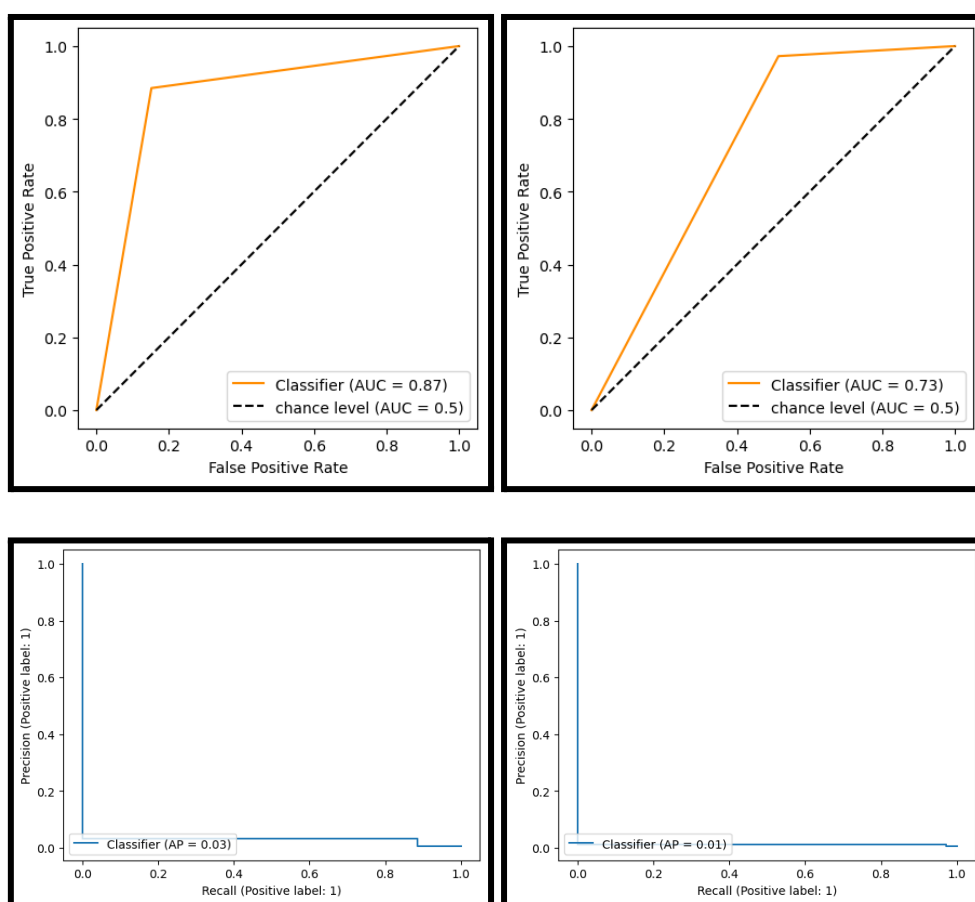


Figure 19: Latest logistic regression model results (Left: Unbalanced, Right: SMOTE)

Discussion: We will select the refined unbalanced-data logistic regression model with 14 features for the final model selection stage given the recall-accuracy tradeoff. It is also a better classifier from a data science model perspective with an AUC score of 87%.

6.3 Neural Network Multi-Layer Perceptron Machine Learning Model

How does it work? Multi-layer perceptron is a neural network model that categorizes input data into one of two separate states based on a training procedure carried out on prior input data (QuantStart, 2022). It is similar to how SVM forms a linear boundary. The perceptron model takes a set of scalar input features, along with a constant 'bias' term and allots weights to these inputs. This linear combination of weights and inputs is then fed through an activation function that determines which state (class) the set of inputs belongs to.

Multi-Layer Perceptron Models For Detecting Credit Card Fraud At The Iron Bank - Baseline Results: Baseline unbalanced-data multi-layer perceptron model performed at a promising level pre-tuning. It currently detects almost 3 out of every 4 fraudulent transactions. However, the neural network model failed to perform when fitted on synthetically-balanced data. We'll now tune these models to see if we can get better results.

Table 10: Baseline multi-layer perceptron model results

	Recall	Accuracy	F1	Precision	AUROC	RP Curve
Without SMOTE	74%	100%	70%	66%	87%	49%
With SMOTE	1%	81%	0%	0%	41%	1%

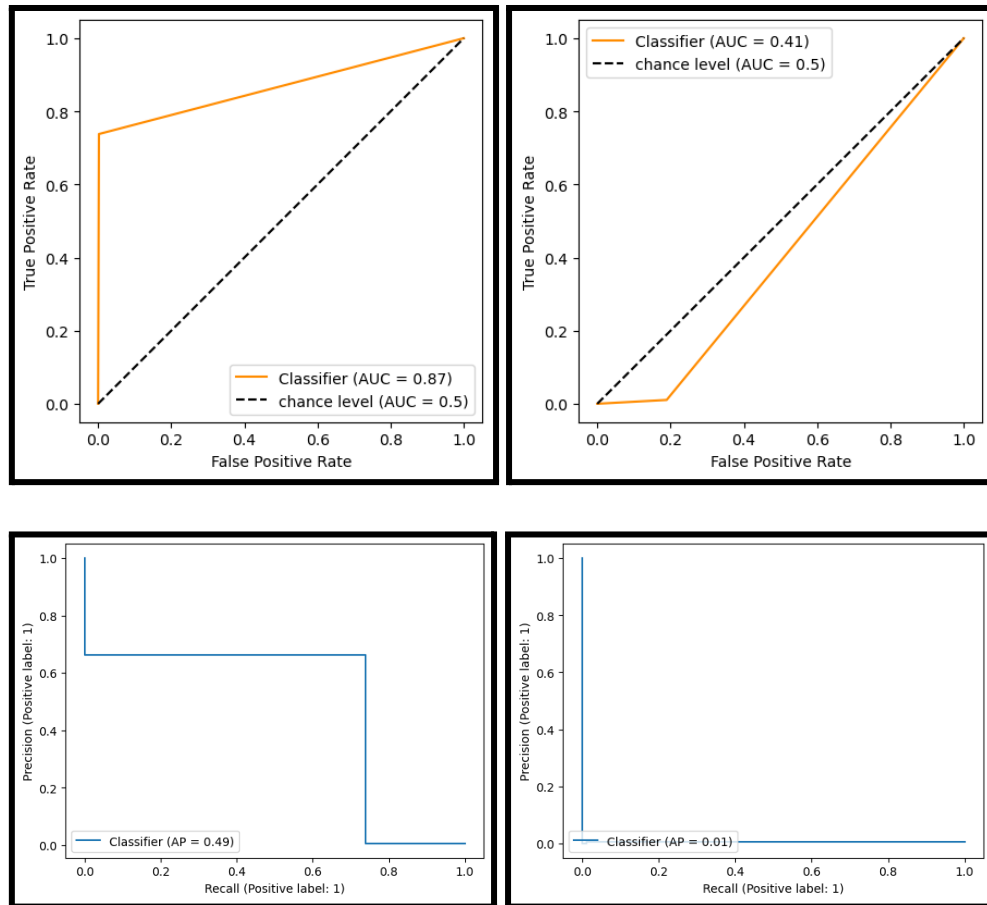


Figure 20: Baseline multi-layer perceptron model results (L: Unbalanced, R: SMOTE)

Results after tuning the hyperparameters: Since the neural network models perform feature selection themselves in their learning process by assigning weights to inputs, we proceeded directly to the tuning of hyperparameters. Due to the limitations imposed by the training time of our model, we were only able to attempt a few rounds of tweaks and they resulted in unsuccessful results. Our unbalanced-data model now detects less fraud than before. Thus, the default settings are better for now, and we'll leave it as a recommendation for the Iron Bank to explore hyperparameter tuning for multi-layer perceptron models as future research. Hypertuning settings of the model with balanced data produced high recall at the expense of accuracy.

Table 11: Latest multi-layer perceptron model results

	Recall	Accuracy	F1	Precision	AUROC	RP Curve
Without SMOTE	66%	100%	73%	81%	83%	53%
With SMOTE	96%	57%	3%	1%	76%	1%

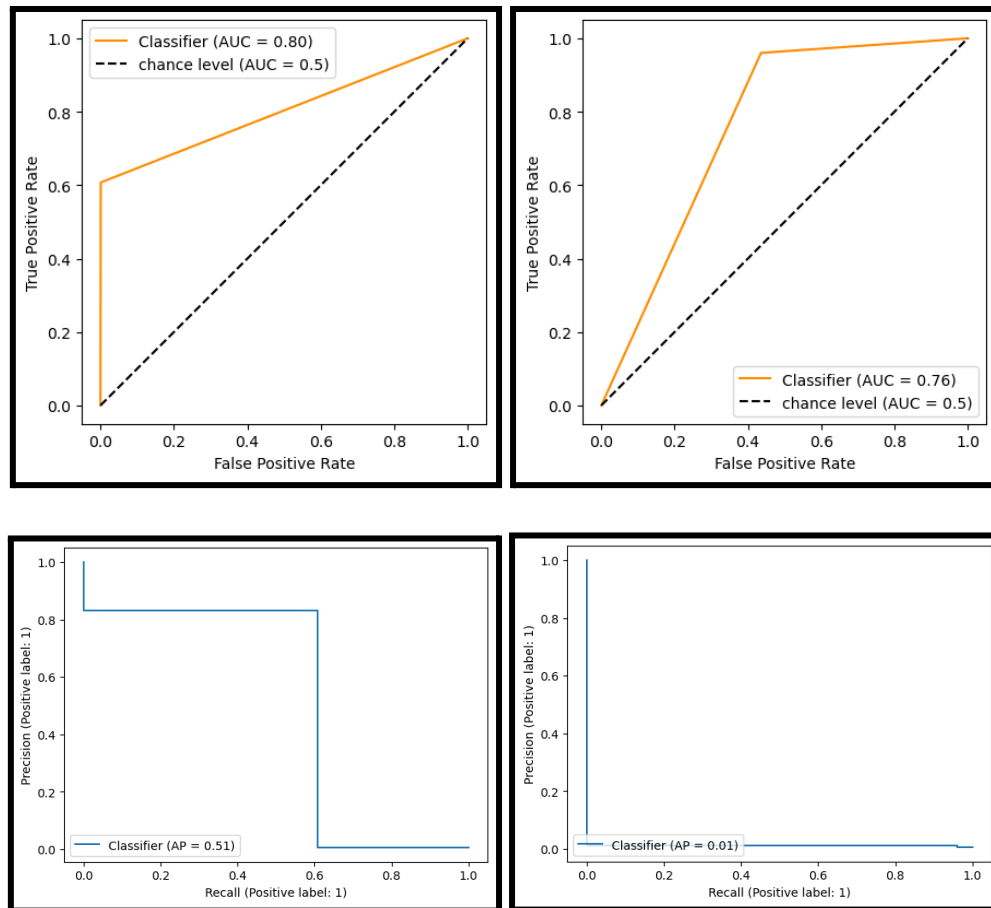


Figure 21: Latest multi-layer perceptron model results (L: Unbalanced, R: SMOTE)

Discussion: Our multi-layer perceptron models failed to perform on par with other models we've investigated thus far. For the final model selection stage, given the recall-accuracy tradeoff, we will select the baseline unbalanced-data model. It is also the better classifier from a data science model perspective with an AUC score of 83%.

7. Final Model Selection & Discussion

After running our chosen models on the held-out test set (30% of the dataset), two of our models failed to perform as expected: Random Forest and Multilayer Perceptron (see table and figures below). However, the other two, Support Vector Machine and Logistic Regression, held up as expected. We discuss these outcomes after you digest the results below.

Table 12: Model performance on the held-out test set (30% of the dataset)

	Recall	Accuracy	F1	Precision	AUROC	RP Curve	Fraud \$ Missed
SVM	92%	76%	3%	1%	84%	1%	~ \$410 thousand
Logistic Regression	81%	85%	4%	2%	83%	2%	~ \$974 thousand
Multi-layer Perceptron	10%	99%	8%	7%	55%	1%	~ \$4.6 million
Random Forest	10%	93%	1%	1%	52%	0%	~ \$4.6 million

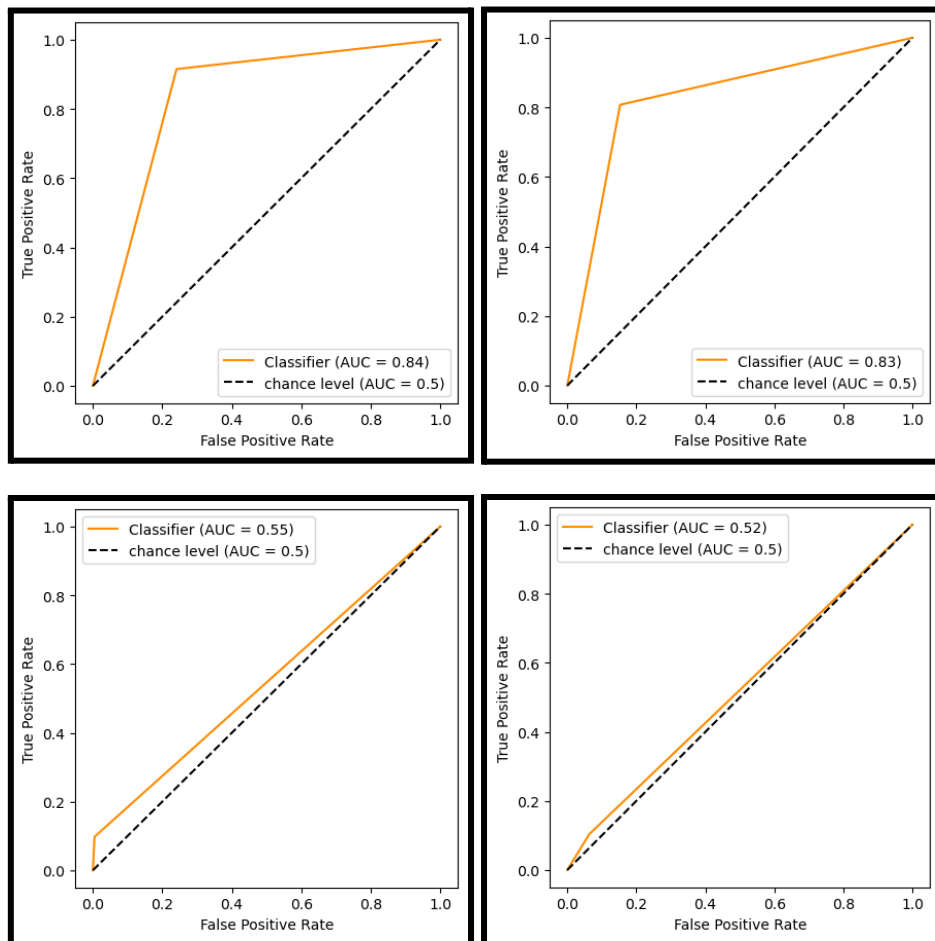


Figure 22: Final results (TL: SVM, TR: Logistic, BL: M-Perceptron, BR: Random Forest)

Our objective for conducting this research was to recommend credit card fraud detection machine learning models to the Iron Bank in order to assist the organization with minimizing loss incurred due to fraudulent transactions. Currently, that amount is \$5.1 million dollars biennially based on our dataset, with an average fraudulent transaction amounting to US \$531 (\$390 median). To this end, we recommend two models to our client: Support Vector Machine and Logistic Regression, with caveats presented below and suggestions for future research in the following section.

The Support Vector Machine model we propose has 14 features (listed in table 13 below) and hyperparameter settings as follows (a scikit-learn classifier):

```
SGDClassifier(alpha=0.5, class_weight='balanced', penalty='elasticnet',  
max_iter=5000, n_jobs=-1, random_state=0)
```

The SVM model detects fraud at the rate of 92-93% and inaccurately classifies an estimated 1 out of every 4 transactions (26%). In terms of fraud minimization, our proposed model consistently performs well, reducing the \$5.1 million biennial loss to \$410,000. However, this comes at the expense of accuracy as a quarter of the transactions are estimated to be misclassified which will likely result in upset customers. Since less than 0.6% of the transactions are fraudulent in our dataset, almost all of the misclassified transactions are likely to be legitimate. Thus, we propose that the Iron Bank first conduct a cost-benefit analysis study to estimate the dollar value of misclassifying a legitimate transaction as fraudulent and then comparing that amount to fraud reduction before making the decision to accept or reject the model in its current form. On the other hand, in the next section, we propose a range of future research topics to further refine the models so that the detection rate stays high while the accuracy score improves.

The second model that we propose to address fraud detection at the Iron Bank is the Logistic Regression model with 14 features also (1 different than the SVM model; see table 13), which has the following hyperparameter tuning composition (a scikit-learn classifier):

```
LogisticRegression(C=1000, class_weight='balanced', max_iter=1000, n_jobs=-1,  
random_state=2)
```

This model detects fraud at a lower rate (81-88% in our estimations), however, accuracy is vastly improved as almost 44 out of 50 transactions are now correctly classified. Thus, fewer customers would have to contact the Iron Bank to unfreeze their accounts. At the

fraud detection rate of 81%, this model reduces the \$5.1 million biennial loss amount to \$974,000. Again, we duly advise the Iron Bank to undertake further research to make a weighted and informed decision.

Table 13: Final model features for the proposed models

Proposed Support Vector Machine Model Features	Proposed Logistic Regression Model Features
1. credit_card_number 2. merchant 3. category 4. amount(usd) 5. gender_1 6. gender_2 7. city 8. zip_code 9. job 10. hour_of_day_cat 11. month_cat 12. velocity 13. age 14. full_name	1. credit_card_number 2. merchant 3. category 4. amount(usd) 5. gender_1 6. gender_2 7. zip_code 8. job 9. hour_of_day_cat 10. month_cat 11. velocity 12. age 13. distance_from_home 14. full_name

Finally, we reject the “Multi-layer Perceptron” and “Random Forest” models in their current form as they fail to fit our objective. The models showed promising preliminary results, especially the random forest model, however, they failed to generalize and produce valuable results. We conducted an investigation to attempt to understand their failure by revisiting our code, however, we were unable to find markedly better results than the ones above after performing the following tweaks and re-running the models:

- a) Re-split the training set into training and validation sets employing the “shuffle=False” setting when using scikit-learn’s “train_test_split” command.
- b) Re-split the training set into training and validation sets employing the “shuffle=True” and “stratify=(by Y values)” settings using scikit-learn’s “train_test_split” command.
- c) Used SMOTE on new training sets derived using the “train_test_split” command.

- d) Used MinMaxScaler which changes all values in the dataset to between 0 and 1 while preserving original distributions.

One cause of such poor performance could be that several similar observations from the two classes lie on or close to the decision boundary, thus convoluting machine learning. Therefore, we've left it as a recommendation for the Iron Bank to explore undersampling techniques along with synthetic-minority class oversampling to improve the results.

Our final model selection consists of the SVM and the Logistic Regression models only with current results.

8. Conclusions & Recommendations

From the vantage point of credit card fraudulent transaction detection, our work produced two models that we are recommending to the Iron Bank which detect fraud at the rates of 92% and 81%, the SVM and Logistic Regression models respectively. However, as expressed, the accuracy score requires improvement to foolproof the models and more research is suggested before making the final model selection. To that end, we recommend the following research topics for the Iron Bank to undertake in the future:

- a) Explore the use of Tomek links and ‘keep and delete’ undersampling methods, such as One-Sided Selection and Neighborhood Cleaning Rule, along with Synthetic Minority Over-sampling (SMOTE) when addressing the class imbalance problem at the data pre-processing stage.
- b) Explore different target encoding approaches to encoding high-cardinality nominal categorical variables, such as “Leave One Out Encoder,” “Generalized Linear Mixed Model Encoder,” “CatBoost Encoder,” and more.
- c) Explore the use of “One-Hot-Encoder” for encoding all low-to-moderate-cardinality nominal categorical variables.
- d) Explore the hyperparameter tuning of the multi-layer perceptron model and other models as desired.
- e) Conduct a cost-benefit analysis study to weigh the marginal loss in model recall score to the marginal loss in model accuracy score.

We also recommend the Iron Bank consult a senior data science specialist to audit our proposed (and failed) models to further enhance the value of this work. We are glad to submit two strong candidate models which lower the loss experienced from fraud for the Iron Bank from \$5.1 million to ~\$410,000-974,000 biennially.

References

1. Abd Elrahman, S. M., & Abraham, A. *A review of class imbalance problem*. Journal of Network and Innovative Computing, 1(2013), 332–340.
2. Bowling, S. (2022) *8 of the biggest issues facing the banking industry*, *American Banker*. American Banker. Available at: <https://www.americanbanker.com/list/8-of-the-biggest-issues-facing-the-banking-industry> (Accessed: January 8, 2023).
3. Federal Trade Commission (2022) *Consumer Sentinel Network Data Book 2021*, *Federal Trade Commission*. Available at: <https://www.ftc.gov/reports/consumer-sentinel-network-data-book-2021> (Accessed: January 8, 2023).
4. De Veaux, R.D., Ungar, L.H. (1994). *Multicollinearity: A tale of two nonparametric regressions*. In: Cheeseman, P., Oldford, R.W. (eds) *Selecting Models from Data*. Lecture Notes in Statistics, vol 89. Springer, New York, NY. https://doi.org/10.1007/978-1-4612-2660-4_40
5. Johnson, E. (2021) *Tomek links, smote, and XGBoost for fraud detection*, *Medium*. Medium. Available at: <https://epjohnson13.medium.com/tomek-links-smote-and-xgboost-for-fraud-detection-1fc8b5208e0d#bce8> (Accessed: January 8, 2023).
6. Le Borgne, Y.-A. et al. (2022) *Reproducible machine learning for credit card fraud detection - practical handbook*, *GitHub*. Available at: <https://fraud-detection-handbook.github.io/fraud-detection-handbook/Foreword.html> (Accessed: January 8, 2023).
7. Maklin, C. (2022) *Support Vector Machine Python Example*, *Medium*. Towards Data Science. Available at: <https://towardsdatascience.com/support-vector-machine-python-example-d67d9b63f1c8> (Accessed: January 8, 2023).
8. McKinney, T. (2019) *DataCamp: Fraud Detection in Python*, *Fraud Detection with Python*. Available at:

https://trenton3983.github.io/files/projects/2019-07-19_fraud_detection_python/2019-07-19_fraud_detection_python.html (Accessed: January 8, 2023).

9. Pargent, F. *et al.* (2022) “Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features,” *Computational Statistics*, 37(5), pp. 2671–2692. Available at: <https://doi.org/10.1007/s00180-022-01207-6>.
10. Priscilla, C.V. and Prabha, D.P. (2020) “Credit Card Fraud Detection: A Systematic Review,” *Learning and Analytics in Intelligent Systems*, pp. 290–303. Available at: https://doi.org/10.1007/978-3-030-38501-9_29.
11. Raj, S. (2020) *Effects of multi-collinearity in logistic regression, SVM, RF*, Medium. Available at: <https://medium.com/@raj5287/effects-of-multi-collinearity-in-logistic-regression-svm-rf-af6766d91f1b> (Accessed: January 8, 2023).
12. Ravelin Insights (2022) *Machine learning for fraud detection*, Ravelin. Available at: <https://www.ravelin.com/insights/machine-learning-for-fraud-detection> (Accessed: January 8, 2023).
13. Scikit (2023) *RobustScaler (Sklearn.preprocessing)*, Scikit. Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html> (Accessed: January 8, 2023).
14. Shenoy, K. (2020) *Credit Card Transactions Fraud Detection Dataset*, Kaggle. Available at: <https://www.kaggle.com/datasets/kartik2112/fraud-detection?resource=download&select=fraudTrain.csv> (Accessed: January 8, 2023).
15. Swaminathan, S. (2019) *Logistic regression - detailed overview*, Medium. Towards Data Science. Available at: <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc> (Accessed: January 8, 2023).
16. QuantStart (2022) *Introduction to artificial neural networks and the Perceptron*, QuantStart. Available at:

<https://www.quantstart.com/articles/introduction-to-artificial-neural-networks-and-the-perceptron/> (Accessed: January 11, 2023).

17. TIBCO (2023) *What is a random forest?*, TIBCO Software. Available at:
<https://www.tibco.com/reference-center/what-is-a-random-forest> (Accessed: January 8, 2023).
18. The Nilson Report (2022) *Payment card fraud losses reach \$32.34 billion*, GlobeNewswire News Room. The Nilson Report. Available at:
<https://www.globenewswire.com/news-release/2022/12/22/2578877/0/en/Payment-Card-Fraud-Losses-Reach-32-34-Billion.html> (Accessed: January 8, 2023).
19. TransUnion (2022) *Suspected Digital Holiday shopping fraud in U.S. increases 127% compared to rest of 2022*. TransUnion. Available at:
<https://newsroom.transunion.com/holiday-fraud-2022/> (Accessed: January 8, 2023).

Appendix I: Project Jupyter Notebook File

Project Jupyter Notebook file can be accessed [at this link](#) (format .ipynb; Google Drive link).