IST707 – Applied Machine Learning

Fraudulent Credit Card Transaction Detection

Dixitha Kasturi
dkasturi@syr.edu

Introduction

In today's time we see a lot of fraud transactions being reported as people find different ways of accessing one's finances. The ecommerce world has been growing rapidly with people preferring the ease of online shopping. With this, any purchase done, transactions are being done online which in some cases is a great source for information(card/payment source) leak. Monetary theft is thereby much easier if one has the information. Fraudulent transactions are usually done from different time zones and areas. Consumers, banks, and merchants are all at danger when a data breach leads in monetary theft and, as a result, a loss of client loyalty and reputation. Fraudsters continue to explore for new ways and tricks to commit fraud, even after various measures to prevent fraud have been deployed. So it is important to analyze the fraudulent transactions to mine patterns and predict if a transaction is fraudulent without being manually reported.

Credit Card Fraudulent Transactions

The goal with this project is to find fraudulent transactions, the factors heavily contributing to it and if any of the contributing attributes have a targeted biasing using machine learning models.

Using Data mining, segmentation of fraudulent transactions and pattern recognition, we can easily identify which transactions are faulty and we can thereby flag the source. The dataset has credit card fraud data from Jan 2019 to Dec 2020. I did exploratory data analysis and time series analysis to identify patterns that contribute to the fraudulent transactions with appropriate visualizations and transformations wherever needed. I used classification and regression techniques to predict if the incoming transaction is fraudulent or not. It is a binary classification problem.

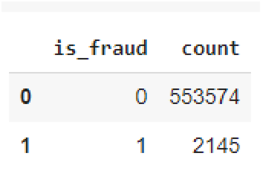
Dataset:

The geographical, demographic, and transaction attributes dataset is a simulated credit card transaction dataset, that was posted on Kaggle encompassing fraudulent transactions from January 1, 2019 to December 31, 2020. It was created with the Sparkov Data Generation tool and includes credit cards from 1000 customers who transacted with 800 retailers.

Train set : 1296675 Rows/transactions, 23 Attributes , January 1, 2019 to June 21, 2020

Test set : 555719 Rows/transactions, 23 attributes, June 21, 2020 to December 31, 2020

The dataset was ***heavily imbalanced.*** To handle this imbalance Undersampling and Oversampling techniques were used, more in detail as the paper progresses.



	is_fraud	count
0	0	553574
1	1	2145

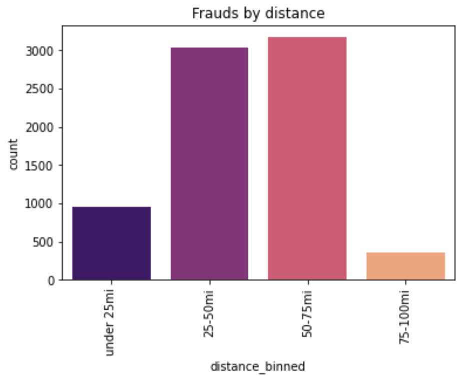
Approach :

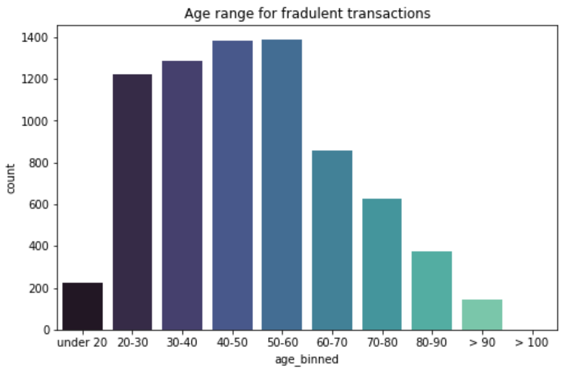
1. EDA :

The dataset was clean with no missing values. There were a lot of columns that were categorical but in string format(city, state, job ,category) , numeric columns(like latitude and longitude, population density) and date- timestamp(transaction timestamp and date of birth). I derived columns from the existing columns and binned them into categories for ease of understanding.

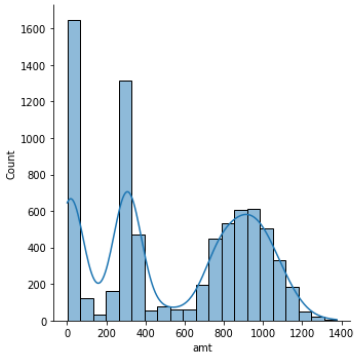
- **Date-time :** Hour, Day, Month, Age of CC owner at the time of transaction
- **Distance :** Between CC owner, merchant(LAT,LONG) – Haversine distance
- **Binning :** Distance, Population Density
- **Recent Stamp :** Difference between current and last transaction
    - Replaced NA's with -1
- Percentage of fraudulent transactions on a card

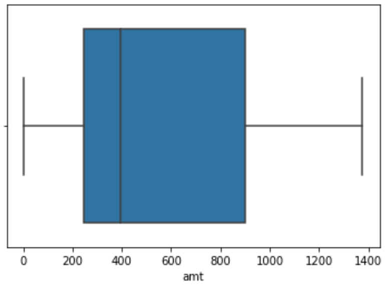
Most of the fraudulent transactions happened in the 25-75 mile distance range. Most of the fraudulent transactions came from 40-60 year old age range of credit card owners, the age represents the age of the credit card owner at the time of transaction.

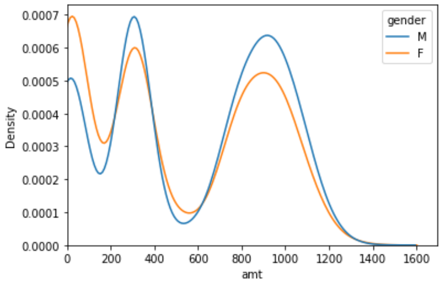




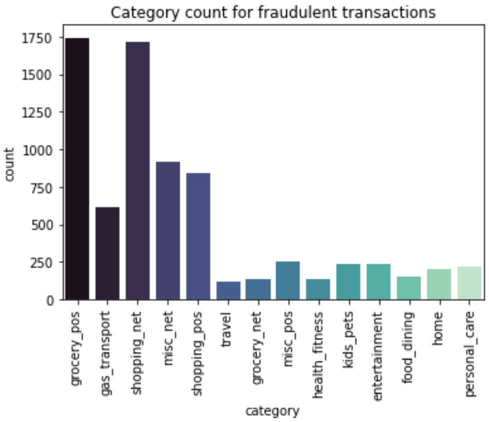
The fraudulent transaction amount maxed at $1400, very small amounts(< $50) were fraudulent and accounted to the highest number of fraudulent transactions. Big amounts of fraudulent transactions were not made. We usually donot tend to notice very small fraudulent transactions(< $10) unless reported by the creditcard user. Bigger transactions are easy to identify if they are fraudulent(usually through telephonic or email verification if the big amount of transaction was performed by the credit card user or no). Most of the lower amount($ 0 - $200) fraudulent transactions came from a female credit card owner.







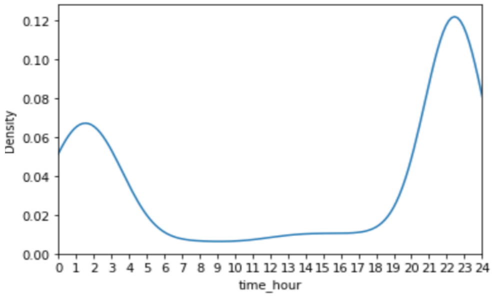
Most of the fraudulent transactions came from Instore grocery shopping and internet shopping. The grocery pos can be due to credit card skimmers as stated in article[1]

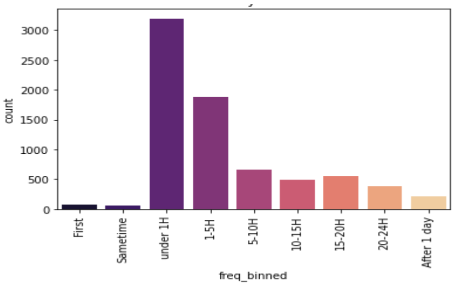


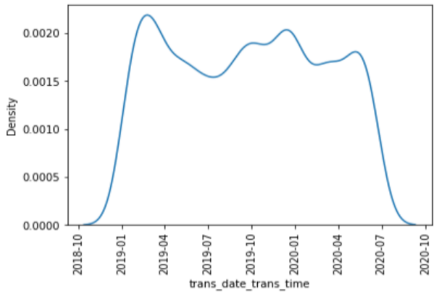
New York accounted to the highest number of fraudulent transactions, followed by Texas

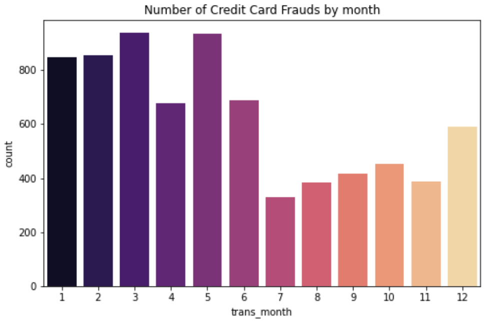
2. Time Series EDA:

Using the time stamp of the transaction, it was easy to understand when most of the fraudulent transactions happened. Most of the fraudulent transactions happened from 6pm in the evening, peaking at midnight till 1 am and then till 4am. By months, the first half of the year recorded highest. Most of the fraudulent transactions happened within an hour of the previous transaction followed by the 1-5 hours period.









3. Transformations :

I used encoding techniques to transform variables, previously converted to categories by binning Usually when we give categorical data to machine learning models, they tend to take more time to get trained on the data. Giving numeric(ranked or unranked) variables would make the training process easier. I used the following encoding to transform the respective variables.

- **Label Encoder** (merchant,distance_binned,frequency_binned_,population_binned, age_binned)
- **One-hot encoder** (Category, gender)

4. **Unbalanced Data :** Before using the transformed data for model training, it was essential to address the unbalanced data problem. There are several techniques to handle data imbalance. I used 3 of them

    1) Under sampling
    2) SMOTE – over sampling
    3) Cross validation

5. Models :

Because the target variable was binary, I used the following models to make predictions.

For each of the models, I compared AUC of ROC for unsampled, over sampled and under sampled data to address the data imbalance, while performing cross-validation and hyper parameter tuning.

a) Logistic Regression:

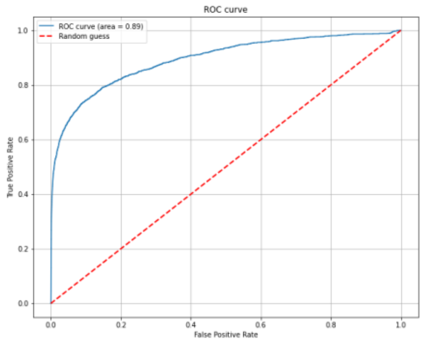
Sampling	Cross-validation score	AUC for train-test split	Best Model paramater
No sampling	0.889	0.8936	C =1000
Under Sampling	***0.9435***	***0.9438***	C = 1000
SMOTE	0.9330	0.9326	C =100

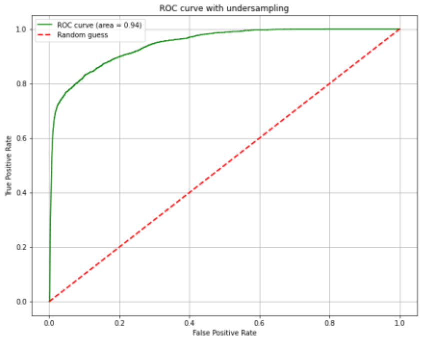
For 70:30 train test split of training set:

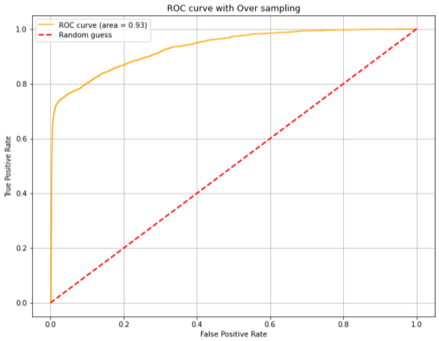
Sampling		Precision	Recall	F1-Score	Accuracy score
No sampling	0	0.99	1	1	0.994
	1	0.39	0.04	0.07	
Under Sampling	0	1	0.88	0.94	0.8837
	1	0.04	0.84	0.08	
SMOTE	0	1	0.98	0.99	***0.9833***
	1	0.22	0.72	0.33	

For unseen Test set

Sampling	Precision	Recall	F1-Score	Accuracy score
No sampling 0	1	1	1	0.9959
1	0.40	0.08	0.14	
Under Sampling 0	1	0.91	0.95	0.9130
1	0.04	0.91	0.07	
SMOTE 0	1	0.98	0.99	***0.9839***
1	0.16	0.73	0.26	







b) Decision trees:

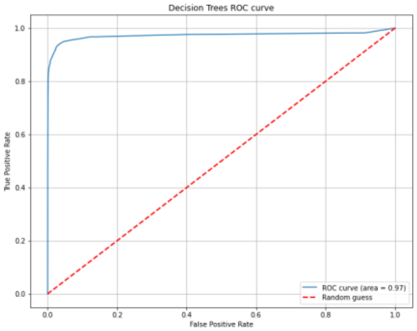
Sampling	Cross-validation score	AUC for train-test split	Best Model paramater
No sampling	0.9579	0.9727	Max depth = 10
Under Sampling	**0.9718**	**0.9786**	Max depth = 10
SMOTE	0.9697	0.9734	Max depth = 10

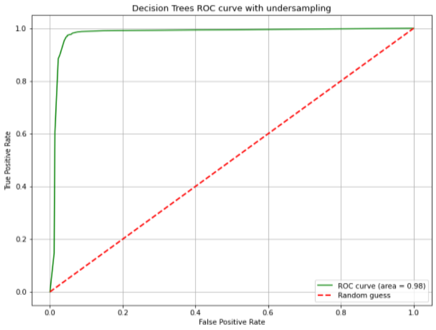
For 70:30 train test split of training set:

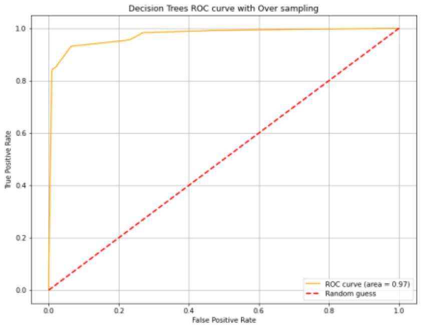
Sampling		Precision	Recall	F1-Score	Accuracy score
No sampling	0	1	1	1	0.997
	1	0.84	0.73	0.78	
Under Sampling	0	1	0.95	0.97	0.9489
	1	0.10	0.97	0.18	
SMOTE	0	1	0.98	0.99	***0.9762***
	1	0.18	0.86	0.29	

For unseen Test set

Sampling		Precision	Recall	F1-Score	Accuracy score
No sampling	0	1	0.92	0.96	0.9172
	1	0.40	0.77	0.07	
Under Sampling	0	1	0.97	0.99	***0.9731***
	1	0.12	0.98	0.22	
SMOTE	0	1	0.94	0.97	0.9388
	1	0.05	0.90	0.10	







c) ***Naïve Bayes :*** For this, I only ran the experiment for different sampling techniques on train-test split of training data and unseen test data

For 70:30 train test split of training set:

Sampling		Precision	Recall	F1-Score	Accuracy score
No sampling	0	0.99	1	1	0.9942
	1	0	0	0	
Under Sampling	0	0.99	0.09	0.17	0.9614
	1	0.01	0.91	0.01	
SMOTE	0	0.99	0.09	0.17	0.9614
	1	0.01	0.91	0.01	

For unseen Test set

Sampling		Precision	Recall	F1-Score	Accuracy score
No sampling	0	1	0.92	0.96	0.996
	1	0	0	0	
Under Sampling	0	1	0.09	0.17	0.9459
	1	0	0.92	0.01	
SMOTE	0	1	0.09	0.17	0.9495
	1	0	0.92	0.01	

Models Conclusion :

Overall decision trees performed the best with SMOTE but for unseen test-data the model that was trained on undersampling gave higher AUC. For both logistic regression and Naïve Bayes, SMOTE worked best on both training and unseen test data.

Conclusion and Future Scope

Because of the data being unbalanced, multiple tests had to be run to get the displayed results, this made the process of running models more time consuming than expected. Could have performed Sensitive Hyper Parameter tuning and Proper Time series trend analysis.

Multiple models can be run with more ensemble techniques and unsupervised learning.

References

[1] Guerrero, M. (n.d.). McKinney Woman Finds Credit Card Skimmer Hidden in Plain Sight Inside Gas Station. Card Skimmer. Retrieved from https://www.nbcdfw.com/news/local/mckinney-woman-finds-credit-card-skimmer-hidden-in-plain-sight-inside-gas-station/2924773/

[2] https://machinelearningknowledge.ai/categorical-data-encoding-with-sklearn-labelencoder-and-onehotencoder/

[3] https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection

[4] https://TowardsDataScience.com

[5] https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/