FRAUD DETECTION IN CREDIT CARD TRANSACTIONS

EECS 6412 DATA MINING PROJECT REPORT

Kamil Amin
York University
kamil988@yorku.ca

Mithila Sivakumar
York University
msivakum@yorku.ca

1 Introduction

With e-commerce websites growing in abundance, people tend to rely on online services to perform their transactions which has led to the increase in fraudulent activities. The importance of having a good fraud detection system is necessary in order to avoid incurring losses. The Credit Card Fraud Detection is an example of a binary classification problem. Here, we train the model using past transactions to find the fraudulent transactions on a future data set. In fraud detection problems, usually there are around 99 percent of non-fraudulent data and only 1 percent contains fraudulent data. If a model predicts all data points as a normal class then the accuracy of the model will be 99 percent but this is not right. This results in accuracy paradox. The main objective of our project is to find the fraudulent transactions in a given synthetic transaction data set using 5 different data mining algorithms and compare the performance of each algorithm to determine the best fit.

The rest of the paper is organized as follows. Section 2 contains Related work, Section 3 talks about Data visualization, Section 4 talks about the dataset and environment used, Section 5 describes our methodology, Section 6 discusses the results and finally Section 7 talks about final thoughts and Conclusion followed by Appendices 7.

2 Related Work

There is a total of 6 solutions posted in Kaggle for this particular dataset. In **Solution 1**, they have used LSTM and CNN with very less pre-processing compared to our approach and they have 100 percent accuracy resulting in accuracy paradox. In **Solution 2**, they have implemented CATBoost algorithm and achieved an F1 score of 85 percent. **Solution 3 and Solution 4** has implemented LightGBM. **Solution 5** performs only statistical analysis of the data and **Solution 6** has implemented Logistic regression but the results are not clearly stated.

3 Data Visualization

We performed a few statistical analysis by taking inspiration from one of the solutions [2] which focused specifically on performing only analysis. We created a map of the USA depicting the number of transactions in each state in Figure 1 followed by a bar chart in Figure 2, which shows the number of transactions by every month.

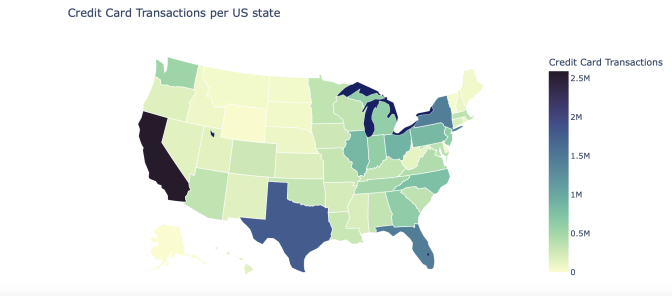

Figure 1: USA MAP showing the distribution of transactions by state

4 Dataset and Environment

We were able to acquire a dataset with more than 20 million transactions from Kaggle. They were generated from a multi-agent virtual world simulation performed

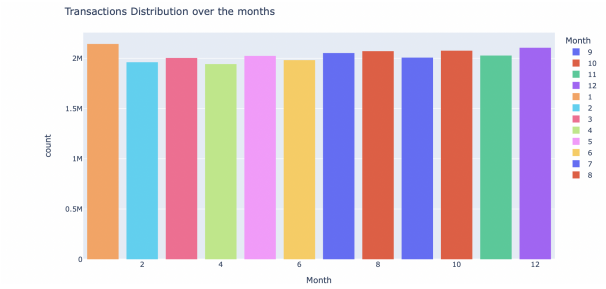

Figure 2: Bar chart showing transactions by Month

by IBM. The data covers 2000 (synthetic) consumers resident in the United States, but who travel the world. The data also covers decades of purchases, and includes multiple cards from many of the consumers [2].

We have used Python [3] and scikit-learn [5] as our environment and we used Google colab as our IDE.

5 Methodology

Our main methodology is to focus on Data pre-processing and Dimensionality Reduction. The full list of all the pre-processed columns is given in Figures 7, 8 and 9 under Appendices. The main takeaway from this section is that we have tried to use most or all of the pre-processing techniques explained in class to us. Our main goal here was to put in action what we learnt in the course.

Data Cleaning

	Missing count	Missing ratio
Merchant State	2720821	0.111569
Zip	2878135	0.118020
Errors?	23998469	0.984072

Figure 3: List of columns with missing values in the dataset along with their count and ratio

Data cleaning is the process of filling missing values in the dataset. In our data, few columns had missing values. These missing values were not present as a result of incomplete data. For example, the column ***Merchant State*** had empty values for online transactions. This is because for the case of online transactions, there is no merchant state. Hence, we filled in *Online* for missing values in this column, since it was empty only for online transactions. We identified few columns, listed in Figure 3 which had missing values in this sense and filled in with meaningful values.

- ***Merchant State*** : Filled "Online" for empty values.
- ***Zip*** - Filled "0" for online and "99999" for international.
- ***Errors?*** - Filled "No errors" for empty values (Empty values were present only when there is no error).

Feature Engineering Feature Engineering is the process of creating a new column based on a single or a combination of 2 or more columns. We have created approximately 40 new columns based by performing some operation on the existing columns. For example, we created a new column ***frequent pin change cardm***, which was obtained by subtracting values in column ***Acct open date*** from ***Expires***. The purpose of creating this column is that it helps to approximately find how often the user has requested for a new card.

Feature Reduction After performing feature engineering, we removed the columns based on which the engineering was performed. We also removed columns which are not relevant to the prediction (that is, no relationship with the class attribute) by performing manual statistical analysis. For example, we removed the ***Card on Dark Web*** column because it had only one unique value 0 for all the transactions which doesn't have relationship to the class attribute in any way.

Data Integration The dataset which we acquired consists of three different csv files. The first table contains all the 24 million transactions. The second table contains all 2000 Users and their demographic information. The third and final table contains information about the user's different cards (FICO score, Card type etc). After pre-processing all the columns in every table, we finally merged all the columns into one single file and ended up with approximately 55 attributes. We wanted to merge the relevant columns from the other two tables with the transaction table because we believe that these features will help to get more information about the fraudulent transactions and hence help with the model training.

Normalization After we completed the pre-processing steps explained above, before passing this dataset to the

algorithms as input, we performed normalization so as to scale the attribute values to fall within a small, specified range. We chose min-max normalization and performed the scaling so that all values were within 0 and 1 which will help the algorithms converge faster.

Case Reduction Since there are 24M transactions in this dataset, it will not be feasible to pass around 20 million records as training set to any of the algorithms. So in order to avoid this, we need to perform sampling (selecting limited number of records from the dataset and pass it as training set to the algorithms). We have tried three different types of sampling.

The first one is ***Random Under Sampling***. This is nothing but the sample will retain the total number of minority classes as in the original dataset and reduce the number of majority class (equal to minority class by default or we can specify the percentage).

The second method which we used for sampling is called ***Random Sampling***, which is nothing but randomly picking the specified number of samples from the original dataset.

The third and final sampling method which we selected was a combination of ***Random Over and Under sampling***. In this method, we will first over sample the minority class and then under sample the majority class based on the number of minority class.

We used packages from sklearn ***(SMOTE and Imbalanced learn to be specific)*** to implement all the sampling techniques.

Dimensionality Reduction Since we ended up with approximately 55 columns after all the pre-processing, we needed to perform dimensionality reduction, that is, reduce the number of features/attributes so that we can send only the richest of features to the data mining algorithms to achieve better performance. ***We wanted to select a methodology which has been rarely used or relatively new to show novelty in our project***. Fortunately, while researching in this regard, we came across two such algorithms.

The first one is called **UMAP** [4]. UMAP is a dimensionality reduction algorithm that is based on manifold learning techniques and topological data analysis. However, there is a known bug in UMAP where it gets stuck at some point. We tried running it for over a day but it ended up getting stuck at the same point every time. So we decided to search for a better algorithm. We came across ***IVIS*** [6], which is another dimensionality reduction algorithm.

IVIS is relatively new (published in 2019) and is based on Siamese Neural Networks. It performs supervised as well as unsupervised dimensionality reduction. In supervised, it makes use of available class labels to perform dimenisonality reduction. it supports both classification and regression and makes uses of the loss included in Keras (neural networks) [1]. ***To our knowledge, none of the solutions in Kaggle have used IVIS or there is not many examples that make use of this algorithm. So we decided to use IVIS to perform the reduction of dimensions.***

Based on the guidelines presented in the official IVIS website [6], through the process of trial and error we ended up with the optimal hyper-parameters for IVIS which yielding close to perfect results. The main hyper-parameters in IVIS are as follows:

- ***embedding dims***: Defines the number of dimensions we want. We tried 2, 3 and 7 and ended up with choosing 7 for feeding to the algorithms.
- ***n epochs without progress***: Number of epochs to run when there is no improvement in result. The document guides us to use less value than default so we have used 3.
- ***supervision weight***: To control the relative importance IVIS places on the labels during training. This can be between 0 and 1. In the docs, they have specified that using 0.8 will clearly define the classes. So we have used 0.8.
- ***k***: The number of nearest neighbours to retrieve for each point. Low k values will result in prioritisation of local dataset features Conversely, high k values will force IVIS to look at broader aspects of the data, losing desired granularity. We tried both lower (k=15) and higher (k=85) but achieved better results with k=15. This is can be seen in Figure 6 vs k=15 in Figure 4

Transforming the dataset with IVIS resulted in a total of 7 features. We had to perform min-max normalization once again on the IVIS transformed dataset as it applied PCA and the values were no longer between 0 and 1. A snapshot of the final dataset is given in Figure 5, which is ready to be passed as input to the data mining algorithms.

We trained five different data mining algorithms to find which one performed the best. The classifiers which

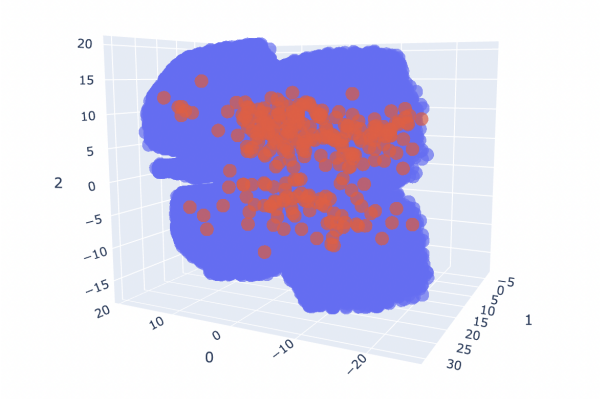

Figure 4: A 3-D graph showing the not so clear partition of both classes (Red-Fraud, Blue- Not Fraud) when k=85 in IVIS

	0	1	2	3	4	5	6	label
0	0.755608	0.485490	0.495076	0.629360	0.469401	0.611696	0.739901	0.0
1	0.668953	0.537702	0.480973	0.670242	0.366366	0.416830	0.726169	0.0
2	0.651498	0.521750	0.506433	0.701506	0.413261	0.613676	0.782894	0.0
3	0.758055	0.478949	0.504500	0.619930	0.477581	0.611527	0.748637	0.0
4	0.755918	0.494885	0.381224	0.568077	0.504888	0.427512	0.509551	0.0

Figure 5: Snapshot of the final IVIS transformed, min-max normalized dataset

we chose were ***XGBoost, Random Forest, Decision Trees, K- Nearest Neighbours and Bayesian Classifiers.***. In the results discussion section, we have explained which amongst the three sampling methods specified here yielded the best result.

First we split the entire dataset into ***training and test set (80/20).*** We took the training set and ran the grid search for different parameters of the classifiers and cross validation using ***GridSearchCV*** from scikit learn. After finding out the optimal parameters, we sampled the training data using the sampling methods explained above and used it to train the five algorithms. Finally, we tested the algorithms using the test set. The results of these tests are discussed in the next section.

6 Results Discussion

Upon testing the three different sampling techniques, we were able to infer that the combination of over and under sampling resulted in the best scores. For example, for the ***Random Sampling*** (2.5 million transactions randomly sampled), the highest ***ROC-AUC score was 84 percent***,

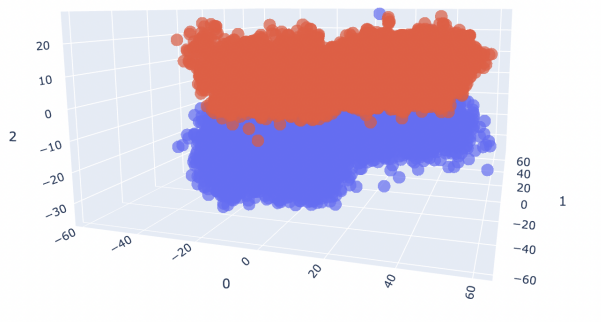

Figure 6: A 3-D graph showing the clear partition of both classes (Red-Fraud, Blue- Not Fraud) when k=15 in IVIS

Classifier	ROC-AUC
XGBoost	97.5
Random Forest	97.4
Decision Tree	97.1
Bayesian	96
KNN	96

Table 1: Algorithms ranked according to the ROC-AUC score when Combined Sampling was used.

for the ***Random Under Sampling***, it was ***96 percent*** and finally for the ***combined sampling, it was 97 percent***. It can also be seen that both Random Under Sampling and Combined sampling resulted in a better performance.

From the table 1, it can be inferred that ***XGBoost had the higest ROC-AUC score with 97.5 percent***, followed by Random Forest and Decision Trees. Our F-1 score was about 99 percent, beating one of the solutions posted in Kaggle as well. Also, due to the heavy pre-processing and use of IVIS, it can be seen that all 5 algorithms perform very well.

7 Conclusion

Fraud detection is a tricky problem where the number of fraud transactions is very rare compared to non-fraud. It is a classic example of imbalanced classification and care needs to be taken while feeding such imbalanced datasets to data mining algorithms. In this project, we have taken a synthetic transaction dataset from Kaggle, created by IBM, that contains 24 million transactions of about 2000 users in the USA. Our main methodology was to apply multiple pre-processing techniques, so that

the features will be optimized in a way that will be well understood by the data mining algorithms. Since this a very large dataset, we gave importance to the type of sampling performed as well. Our main novelty in this project is the use of IVIS dimensionality reduction algorithm which no existing solutions have used. By using IVIS, we were able to acquire near perfect solution. We implemented 5 different algorithms on the final IVIS transformed dataset. From the results, it can be inferred that ***XGBoost performed the best of all algorithms with an ROC-AUC score of 97.4 percent***. For future work, we plan to interpret the IVIS transformed features.

References

[1] Antonio Gulli and Sujit Pal. *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[2] IBM. Transaction Dataset. https://www.kaggle.com/datasets/ealtman2019/credit-card-transactions?select=sd254_users.csv.

[3] Mark Lutz. *Programming python*. " O'Reilly Media, Inc.", 2001.

[4] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[5] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[6] Benjamin Szubert, Jennifer E Cole, Claudia Monaco, and Ignat Drozdov. IVIS offical Docs. https://bering-ivis.readthedocs.io/en/latest/supervised.html.

Appendices

A User Manual

Before running our program on your system, the following requirements must be installed for smooth running of the program. The entire program was written and run in google colab. Using this platform gives us access to a lot of packages by default.

- ***Python latest version, pip latest version***
- ***pip install imbalanced-learn***
- ***!pip install ivis[gpu]***
- ***pip install -U scikit-learn***
- ***pip install xgboost***

B Columns Pre-Processing Table

Column Name	Operation Performed
Use Chip, Card Brand, Card Type, Has Chip, Card on Dark Web	Changed type to Category
Use Chip	Renaming unique values to S, Ch and ON
Is Fraud?	Changed to binary values 0 for No and 1 for Yes
	Changed name of the column to 'label'
Errors?	Fill nan with 'No Errors'
Merchant State	Fill nan with 'Online'
Amount, Per Capita Income - Zipcode, Yearly Income - Person, Total Debt	Removed $ symbol and changed type to int
Zip	Fill nan with 0 for online transactions, 99999 for international
Card Brand	Renaming unique values to M, V, A and D (Corresponding to Master Card, Visa, Amex and Discover)
Card Type	Renaming unique values to D, C, DP. (Corresponding to Debit, Credit and Debit (Prepaid))
Apartment	Fill nan with 0, Update the values in apartment to 0 if value is 0 else 1 and change type to int
Gender	Update to 1 if Female and 0 for Male
Boolean Features	Changed boolean features to integer (0 or 1)

Figure 7: Data cleaned columns with the respective explanations

Column Created	Value
International	Created this new column based on Merchant state. If Merchant State is in US, then value is 1 else 0
Timec', 'Date', 'TranYear', 'TranMonth', 'TranWeek', 'TranDay', 'TranDayofweek', 'TranDayofyear', 'TranIs_month_end', 'TranIs_month_start', 'TranIs_quarter_end', 'TranIs_quarter_start', 'TranIs_year_end', 'TranIs_year_start', 'TranElapsed'	Created these 14 columns from the Date column and used add_datepart helper function to create additional columns. These columns were created to get different kinds of time so that the model can get a sense of time (since XGBoost or Random Forest doesn't take time into account like LSTM)
Timec	New categorical feature based on Time column which has 6 unique values based on the time (Late night, Early Morning, Morning, Noon, Evening and Night
Hour, hr_sin, hr_cos, mnth_sin, mnth_cos	Created these 5 columns based on Date, Hour and TranMonth respectively. This is to capture the cycle in time (that is to let algorithm know that after 24 its 1)
enc_MN, enc_MS, enc_UI	These columns were created by frequency encoding the Merchant_Name, Merchant_State and User
Bad PIN, Technical Glitch, Bad Card Number, Bad CVV, Bad Expiration, Bad Zipcode, Errors, Insufficient Balance	For each unique value z in this column, created a new column which has 1 for z present and 0 for absent (similar to one hot encoding)
Expires_year, Expries_month	From Expires column in Users table
Acct Open Date_year, Acct Open Date_month	From Acct Open Date in Users table
frequent pin	By subtracting Acct Open Date_year from Year PIN Last Changed
frequent pin change cardm	By subtracting Expiry from Acct open date to find appoximately how often the user has changed cards
Distance	Calculate the distance between customer and merchant as KM, for online transactions it will be 0 and for international its 10000 KM

Figure 8: Feature Engineered Columns with their explanations

Column Deleted	Reason
Errors?,Year,Month,Day, Merchant_City, Time, User, Expires, Acct Open Date, Year PIN Last changed, Latitude, Longitude, Zip, Address, City, State	Feature engineered based on this so no longer needed
Birth_Month	Age already present so removed this
Card Number	16 digit number which is unique for all 2000 users. Since we created other new columns that will have impact on the class attribute, we removed this
Card on Dark Web	Only 1 unique value so no impact on class attribute
CVV	We could not find any relationship between this and class attribute.

Figure 9: Deleted Columns with their explanations