DEPARTMENT OF BIG DATA ANALYTICS

BD2P2 | ADVANCED STATISTICS LAB

Project Report On –

CREDIT CARD FRAUD DETECTION

Team Members:

Rashmi S -21BDA02

Aparna K -21BDA24

Tony Tomy -21BDA44

Index

Context

In 2019, credit card fraud losses amounted to $28.65 billion worldwide. It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase. We developed an ML model, which works on **XGBOOST** to reduce the loss faced by these credit card companies. About 90% reduction in the cost could be achieved using this technique.

Let's Understand Our Dataset better

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions., you can find the link here.

The data is distributed as follows: -

- **(V1 — V28) - masked using Principal Component Analysis (PCA) (confidential)**
- **Time - Represents the seconds elapsed between the current transaction and the first transaction in the dataset**
- **Amount - Represents the total transaction value (In thousands)**
- **Class - consists of two values: 1 for fraudulent cases 0 for non-fraudulent cases**

Methodology

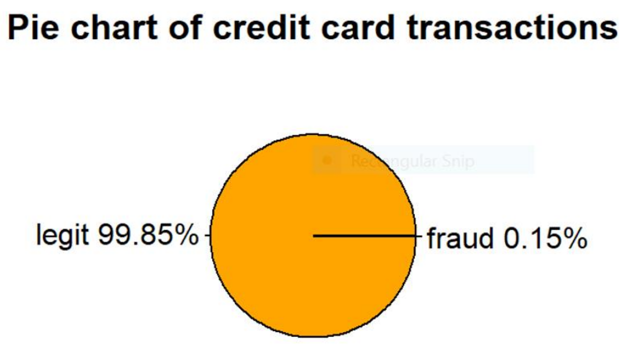
1.1 Importing the Libraries

The necessary libraries such as tidyverse, reshape2, xgboost, ROSE, smotefamily etc were imported

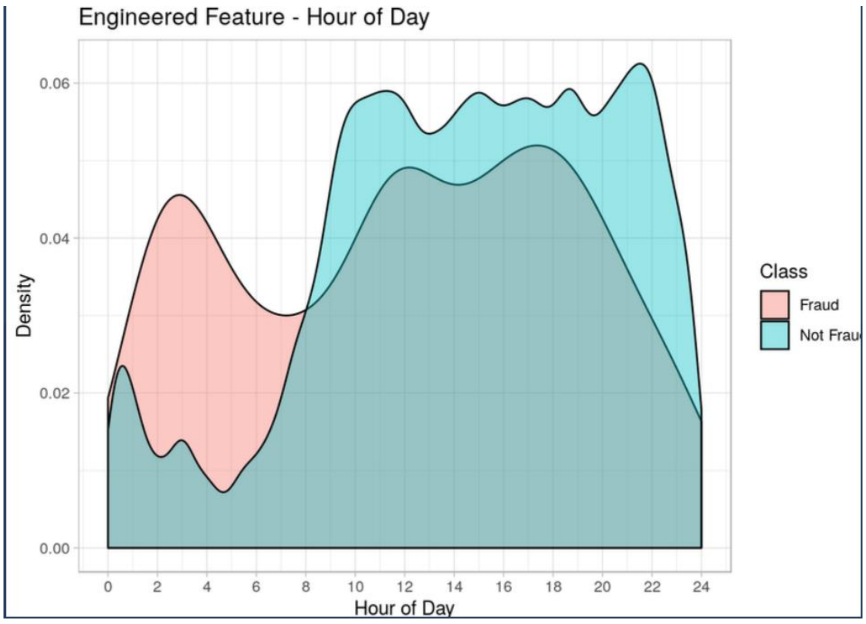
1.2 EDA

While performing basic EDA, it was observed that apart from the features time and amount, all are anonymous. There were no NULL values present in the dataset. While analysing the data, it was observed that the dataset was imbalanced.

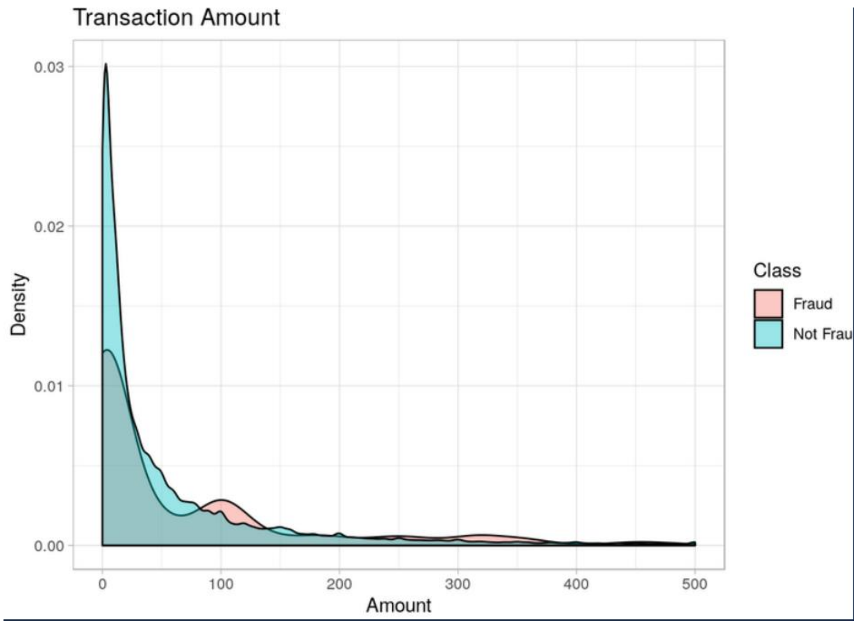
A pie-chart plotted using the dataset showed that 99.83% of the data is of not fraud cases and the remaining 0.17% is of fraud cases.



A graph plotted between the density and hour of day indicates that the number of fraud cases are higher during the early hours of the day.



A graph plotted between amount and transaction amount indicates that as the transaction amount increases, the number of fraud cases are also increasing.



1.3 Feature Engineering

To move further, it is important to balance the dataset. Imbalanced datasets are those where there is a severe skew in the class distribution, such as 1:100 or 1:1000 examples in the minority class to the majority class.

This bias in the training dataset can influence many machine learning algorithms, leading some to ignore the minority class entirely. If we train a binary classification model without fixing this problem, the model will be completely biased. This is a problem as it is typically the minority class on which predictions are most important.

One approach to addressing the problem of class imbalance is to randomly resample the training dataset. The two main approaches to randomly resampling an imbalanced dataset are to delete examples from the majority class, called under sampling, and to duplicate examples from the minority class, called oversampling. The methods that we used for balancing the datasets are:

1.Random over sampling

2.Random under sampling

3. SMOTE

We define Random Sampling as a naive technique because when performed it assumes nothing of the data. It involves creating a new transformed version of our data in which there is a new class distribution to reduce the influence of the data on our algorithm.

Random Over Sampling

Random oversampling involves randomly selecting examples from the minority class, with replacement, and adding them to the training dataset. Oversampling duplicates instances from minority classes by replacing and supplementing the training data. In the Oversampling, some instances are duplicated several times. Here, the number of fraud cases are increased so as to make the dataset balanced.

The advantage of oversampling is that you do not have to delete data points, so you do not delete any valuable information. On the other hand, you are creating data that is not real, so you may be introducing false information into your model.

Random Under Sampling

Oversampling duplicates instances from minority classes by replacing and supplementing the training data. In the Oversampling, some instances are duplicated several times. Random under sampling involves randomly selecting examples from the majority class and deleting them from the training dataset. In the random under-sampling, the majority class instances are discarded at random until a more balanced distribution is reached. In credit card fraud detection, we reduce the number of non-fraud cases using Random Under sampling.

The following figure shows Random Oversampling and Random Under Sampling:

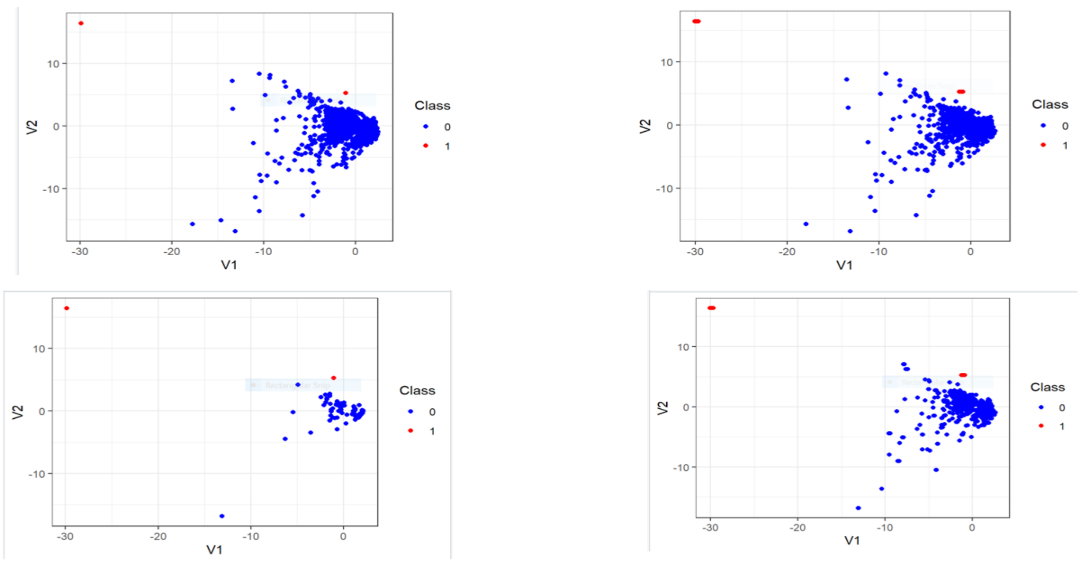


Fig1: The actual dataset

Fig2: Random Under sampling

Fig3: Random Over sampling

Fig4: Both RUS and ROS

The disadvantage with both the Sampling methods are that, in both the cases there is a change in the number of fraud and non-fraud cases. The disadvantage of under sampling is that you lose a lot of valuable data. The disadvantage of oversampling is that it creates many duplicate data points.

1.4 Balancing the Dataset using SMOTE

SMOTE stands for Synthetic Minority Oversampling Technique. SMOTE is an algorithm that performs data augmentation by creating synthetic data points based on the original data points. SMOTE can be seen as an advanced version of oversampling, or as a specific algorithm for data augmentation. The advantage of SMOTE is that you are not generating duplicates, but rather creating synthetic data points that are slightly different from the original data points.

The SMOTE algorithm works as follows:

- You draw a random sample from the minority class.
- For the observations in this sample, you will identify the k nearest neighbours.
- You will then take one of those neighbours and identify the vector between the current data point and the selected neighbour.
- You multiply the vector by a random number between 0 and 1.
- To obtain the synthetic data point, you add this to the current data point.

This operation is actually very much like slightly moving the data point in the direction of its neighbour. This way, you make sure that your synthetic data point is not an exact copy of an existing data point while making sure that it is also not too different from the known observations in your minority class.

After performing all the three methods on the dataset, it was clear that the most efficient one was SMOTE.

.

1.5 Decision Tree in brief

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

We applied decision tree but the accuracy was very low, and hence it was dropped. (The same happened with that of logistic Regression)

1.6 Implementing XG-Boost

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree-based algorithms are considered best-in-class right now.

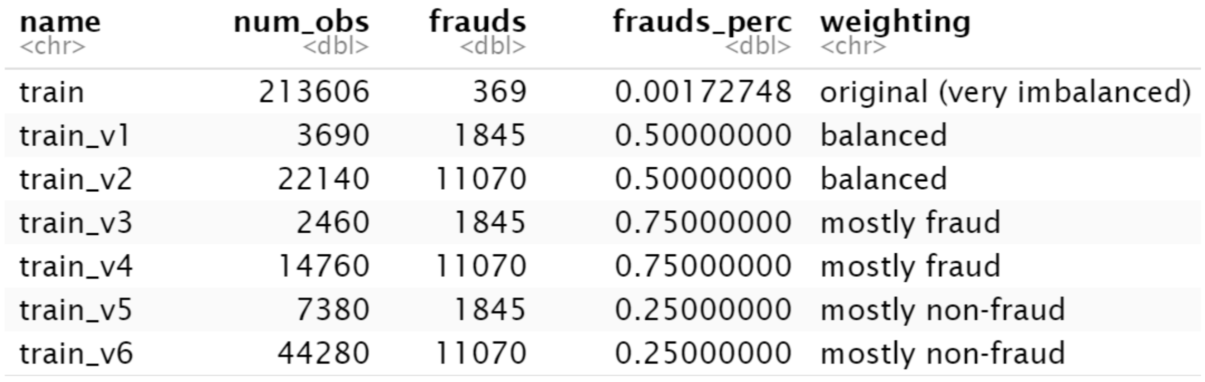
The following steps were implemented to find the optimal solution

Step 1 – The dataset was split into training and testing i.e., 75:25 respectively

Step 2 – The train data was further subdivided into based on 6 different combinations of SMOTE & RUS.

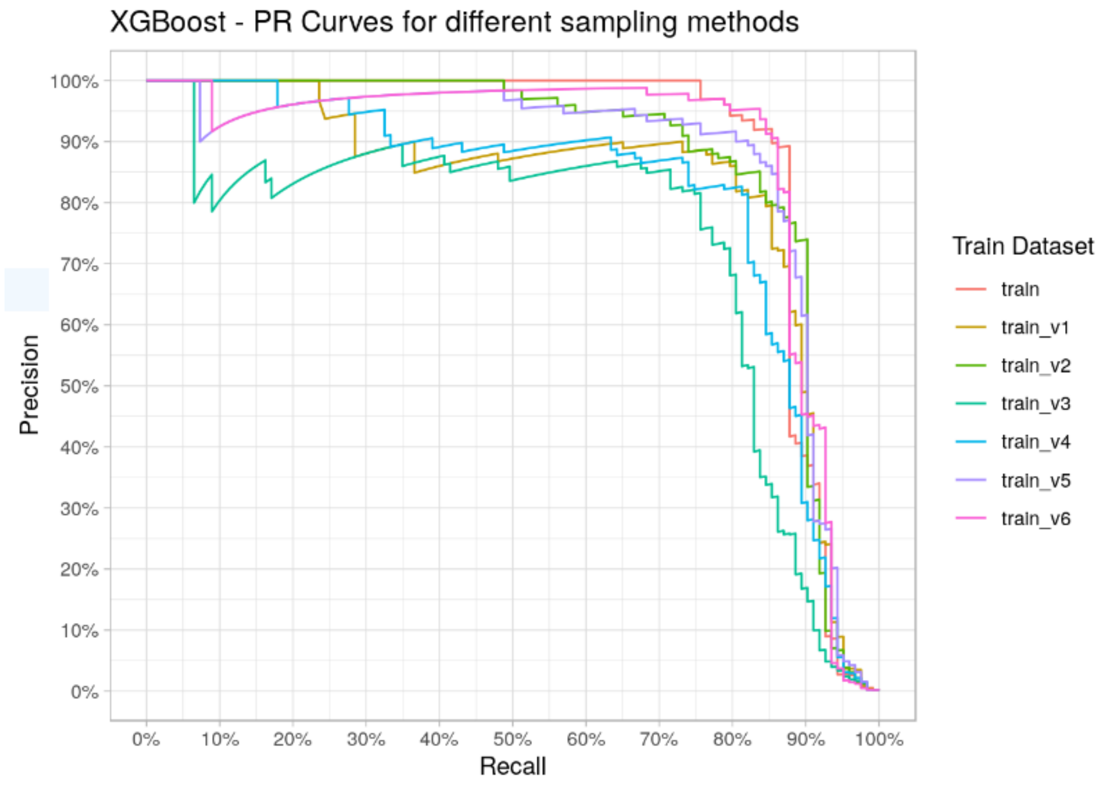
a) The unaltered, highly class-imbalanced dataset
b) A balanced dataset with *less* up-sampling, train_v1
c) A balanced dataset with *more* up-sampling, train_v2
d) A fraud-majority dataset with *less* up-sampling, train_v3
e) A fraud-majority dataset with *more* up-sampling, train_v4
f) A fraud-minority dataset with *less* up-sampling, train_v5
g) A fraud-minority dataset with *more* up-sampling.

The summary of the following tables are shown below



name	num_obs	frauds	frauds_perc	weighting
<chr>	<dbl>	<dbl>	<dbl>	<chr>
train	213606	369	0.00172748	original (very imbalanced)
train_v1	3690	1845	0.50000000	balanced
train_v2	22140	11070	0.50000000	balanced
train_v3	2460	1845	0.75000000	mostly fraud
train_v4	14760	11070	0.75000000	mostly fraud
train_v5	7380	1845	0.25000000	mostly non-fraud
train_v6	44280	11070	0.25000000	mostly non-fraud

Step 3 – Training the 6+1 datasets and checking their PR Score



Using this we see that unaltered data has the maximum Precision-Recall value when compared to others, hence we choose this and proceed

Step 4 – Tuning the parameters

The general rules are as follows:

1. Fix learning rate (eta) reasonably high, find optimum number of trees (nrounds)
2. Tune max_depth and min_child_weight
3. Tune gamma
4. Tune column & row subsampling
5. Reduce learning rate and increase number of trees, find optimum for above parameters

After checking the AUC scores for each of the following parameters, we found that these are the optimal values to train our dataset

- eta = 0.05
- nrounds = 1400
- max_depth = 5
- min_child_weight = 1
- gamma = 0
- colsample_bytree = 0.8
- subsample = 0.6

Step 5 – The model was fit using the following parameters

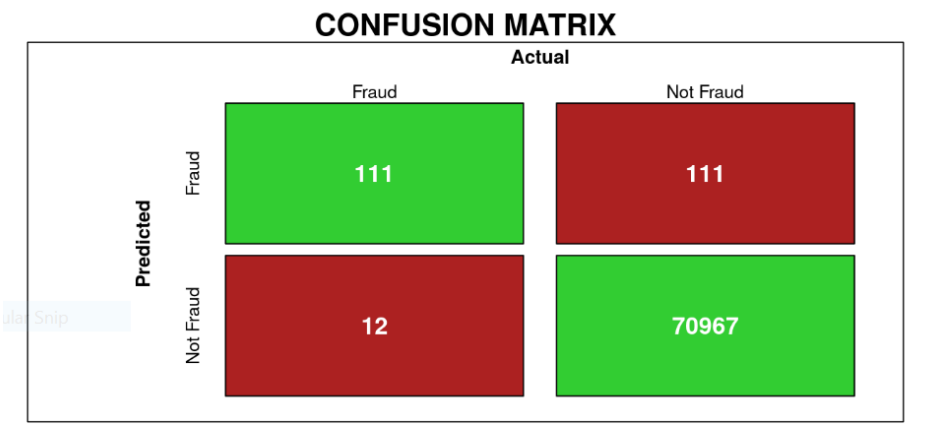
1.7 Testing the accuracy parameters

The output was as follows

```
[1] "Area under the Precision-Recall curve: 0.8282471"
```

```
[1] "Area under the ROC curve: 0.9733025"
```

Plotting confusion matrix after using cost based cut-off



This way we try to decrease the false positives and false negatives.

1.8 Business Impact and Conclusion

There are obviously just 2 types of error our model can make:

False Positive (FP) - predicting *fraud* when the transaction was *not* fraud

- Consequences: Card is locked, card owner calls & explains the mistake, short investigation (security questions etc.), card is unlocked
- Estimated business cost: Low

False Negative (FN) - predicting *not fraud* when the transaction *was* fraud

- Consequences: Fraud is reported, card is locked, larger investigation takes place, card company liable for refunding
- Estimated business cost: High

Let's Make an average estimate

Since the dataset is of a 2-day period, and test data is 25% of this (half a day worth of transactions).

- 111 + 12 = 123 frauds, therefore ~ 246 frauds per day
- **Previous annual business cost** = (£145.61)*(246)*(365) = **£13,074,440.52**
- With our **final model**, we achieved half-day errors of 12 FN's and 49 FP's, with a half-day business cost of £1874.74
- **New annual business cost** = (£1874.74)*(2)*(365) = **£1,368,557.28**

Therefore, an upper bound estimate for annual cost-savings is:
£13,074,440.52 - £1,368,557.28 =~ **£11.7M**

This is a reduction in fraud-handling costs of ~90%!

1.9 References

https://www.geeksforgeeks.org/decision-tree/

https://xgboost.readthedocs.io/en/stable/

https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

https://data-flair.training/blogs/data-science-machine-learning-project-credit-card-fraud-detection/#:~:text=The%20aim%20of%20this%20R,fraudulent%20from%20non%2Dfraudulent%20one.

http://rstudio-pubs-static.s3.amazonaws.com/334864_28050f7860dd4927a596872f0cd52401.html

https://spd.group/machine-learning/credit-card-fraud-detection/

Git hub Link for Presentation, Code and Report

https://github.com/rashmisreenath/Credit-Card-Fraud-Detection

- - -