IEEE-CIS Fraud Detection

Detecting a Fraud Credit Card Transaction

Nakul Gupta
2017068
nakul17068@iiitd.ac.in

Shiva Saini
2017190
shiva17190@iiitd.ac.in

Sachin Singh
2017361
sachin17361@iiitd.ac.in

November 30, 2019

1 Abstract

This project is based on the detection of a fraudulent credit card transaction. The dataset contains over 5 lac data samples. For Exploratory Data Analysis, we used different pre-processing techniques like, correlation analysis, min-max scaling and NAN removal. We also used dataset balancing techniques like SMOTE and AdaSyn. For obtaining results we used one baseline model (Logistic Regression) and three advanced models (Multi-Layer Perceptron, XGBoost, Light Gradient Boosting Machine) from which the best model was picked on the basis of results obtained on the basis of the accuracy score and the plotted ROC Curve.

2 Introduction

The topic of this project is Credit Card Fraud Transaction Detection, which is done by using customer transactions and their identity as input. The project is a recent competition from Kaggle. The goal of the project is to detect whether a customer transaction is fraudulent or not, using an optimal classifier algorithm. Various fraud detection problems have been implemented until now, but the dataset provided by Vesta Corporation has more complex real-world data e-commerce transactions with a wide range of features from device type to product features. Fraud is increasing in the world, exponentially, which results in the loss of millions of dollars. Predicting fraud detection, more precisely, with the optimized algorithm will help companies to reduce fraud transactions.

2.1 Dataset and its Description

- The data is broken into two files identity and transaction, which are joined by TransactionID. Not all transactions have corresponding identity information.
- The transaction Table contains the following features: TransactionDT, TransactionAMT, ProductCD, card1 - card6, addr, dist, P_emaildomain and R_emaildomain, C1-C14, D1-D15, M1-M9, Vxxx.
- The identity table contains the following categorical features: DeviceType, DeviceInfo, id_12 - id_38.

2.2 Evaluation Metrics

The evaluation metric used in the project includes accuracy score and confusion matrix. Moreover, ROC Curve is also being plotted for the algorithms used.

2.3 Algorithms Used

A total of 4 algorithms are used in the project, which includes one baseline algorithm and three advanced algorithms. Following are the algorithms used:

1. Logistic Regression - implemented by Sachin
2. Multi-Layer Perceptron - implemented by Nakul
3. XGBoost - implemented by Shiva
4. Light Gradient Boosting Machine (LGBM) - implemented by Shiva

3 Related Work

The best AUC Score of this project on Kaggle competition is 0.96. To get best accuracy the three model were used CatBoost(0.9639), LGBM(0.9617), XGBoost(0.9602) algorithm. The final submission was a stack where LGBM was trained on top of the predictions of CAT and XGBoost and the other final submission was an ensemble with equal weights. For feature selection multiple techniques were used like forward feature selection (using single or groups of features), recursive feature elimination (using single or groups of features), permutation importance, adversarial validation, correlation analysis, time consistency, and train/test distribution analysis. During the validation process in this competition, XGBoost model did best predicting known UIDs with AUC = 0.99723. LGBM model did best predicting unknown UIDs with AUC = 0.92117. CAT model did best predicting questionable UIDs with AUC = 0.98834.

4 Methodology

For data preprocessing, we performed correlation analysis to check the correlation between different features of our dataset. We set the threshold of 0.7 to group them, out of which, one is chosen as the representative of that group.

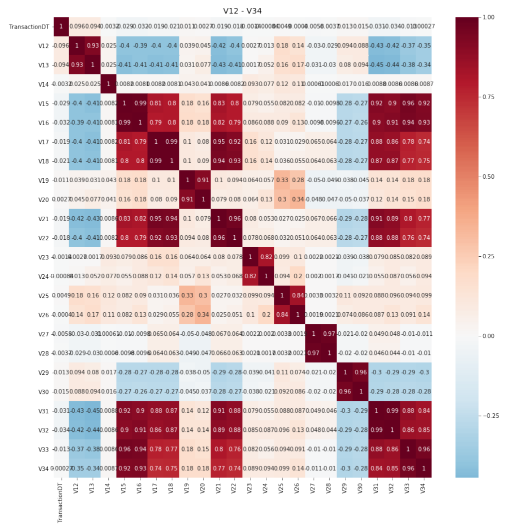

Figure 1: Correlation Analysis between V12-V34

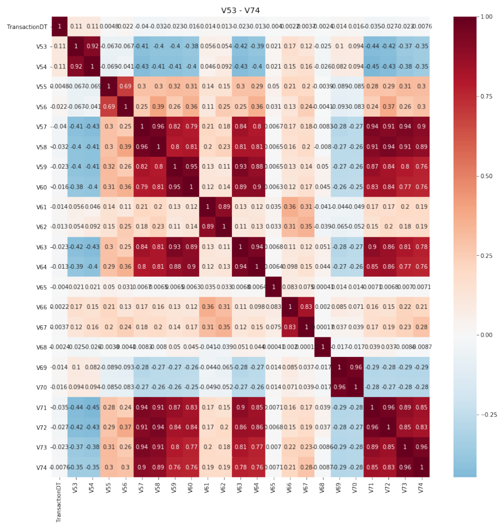

Figure 2: Correlation Analysis between V53-V74

Apart from correlation analysis, we checked for those columns which were filled with NAN values for more than 90% of data samples and for those columns which only had one unique value. We also checked for those columns which have 90% of the data samples as one value. All the columns found above were dropped from the original dataset. We also removed skewness from the data by using min-max scaling and replaced the NAN values, in the categorical columns, with ’EMPTY’ and in the numerical columns, with -1.

After preprocessing, the main challenge we faced during the analysis of dataset was finding if it was imbalanced or balanced. After plotting the countplot (shown below), we found out that the data was highly imbalanced.

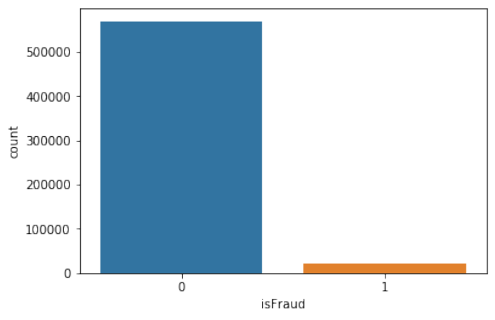

Figure 3: Count Plot of arget Variable

To work with the imbalanced dataset, we used a balancing technique, called SMOTE, which is an oversampling technique. We used it on the data to be used for non robust classifiers like Logistic Regression and Multi-Layer Perceptron.
We used the same balancing technique on the data used in XGBoost and Light Gradient Boosting Machine (LGBM). But, we then found out about the inbuilt objective function to be passed in XGBoost and LGBM to handle the imbalanced dataset during the training of the model.
These objective functions gave a better result so we ended up with using SMOTE oversampling technique for Logistic Regression and Multi-Layer Perceptron, and inbuilt objective function, 'scale_pos_weight' for XGBoost and 'is_unbalance' for LGBM.

5 Results

5.1 Logistic Regression

5.1.1 Imbalanced

The accuracy for the dataset without balancing:
Train Data:

$$96.5\%$$

Test Data:

$$96.4\%$$

Confusion Matrix:

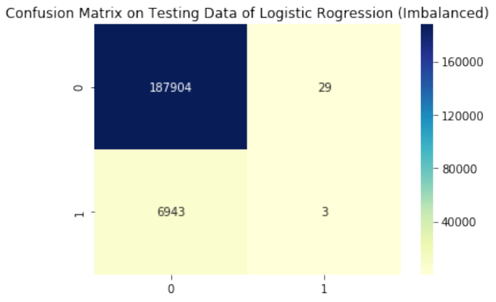

Figure 4: Confusion Matrix of Logistic Regression on Imbalanced Data

5.1.2 Balanced

The accuracy for the dataset with balancing using SMOTE technique:
Train Data:

$$72.1\%$$

Test Data:

$$71.9\%$$

Confusion Matrix:

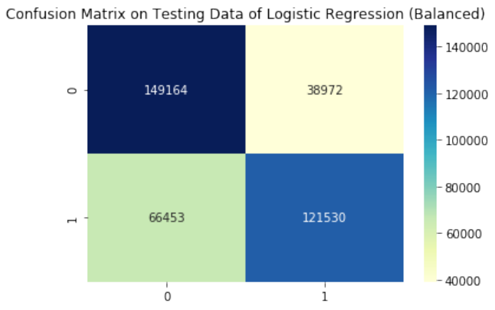

Figure 5: Confusion Matrix of Logistic Regression on Balanced Data

ROC Curve:

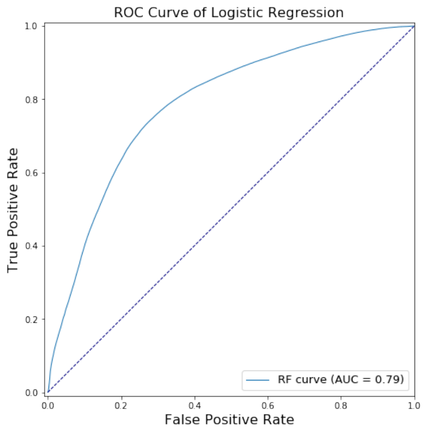

Figure 6: ROC Curve of Logistic Regression on Balanced Data

5.2 Multi-Layer Perceptron

5.2.1 Imbalanced

The accuracy for the dataset without balancing:
Train Data:

96.5%

Test Data:

96.4%

Confusion Matrix:

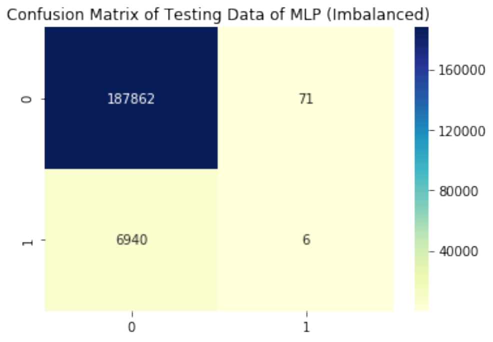

Figure 7: Confusion Matrix of MLP on Imbalanced Data

5.2.2 Balanced

The accuracy for the dataset with balancing using SMOTE technique:
Train Data:

75.97%

Test Data:

75.93%

Confusion Matrix:

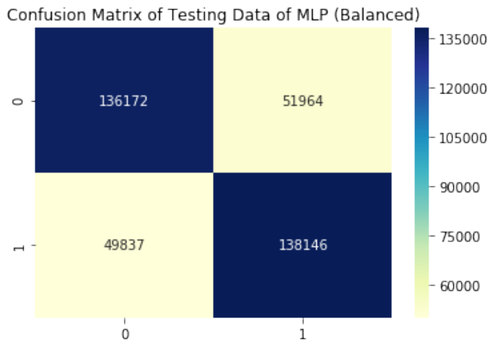

Figure 8: Confusion Matrix of MLP on Balanced Data

ROC Curve:

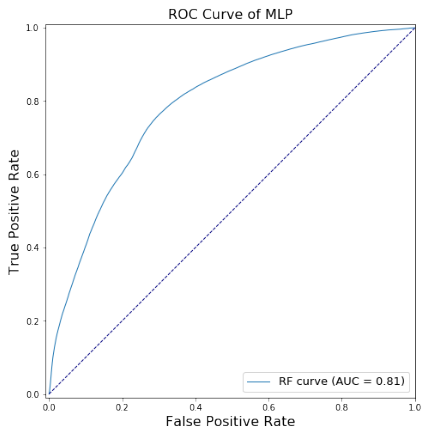

Figure 9: ROC Curve of MLP on Balanced Data

5.3 XGBoost

5.3.1 Imbalanced

The accuracy for the dataset without balancing:
Train Data:

97.4%

Test Data:

97.3%

Confusion Matrix:

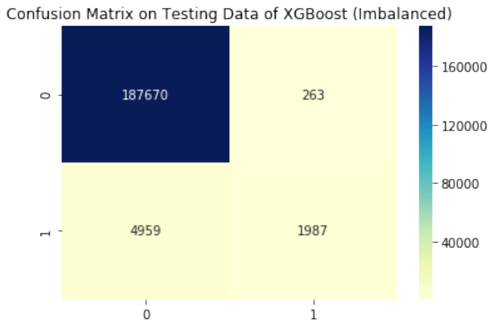

Figure 10: Confusion Matrix of XGBoost on Imbalanced Data

5.3.2 Balanced

The accuracy for the dataset with balancing by setting the fuction 'sacle_pos_weight' equal to (no. of samples with target value == 0)/(no. of samples with target value == 1):
Train Data:

85.1%

Test Data:

84.9%

Confusion Matrix:

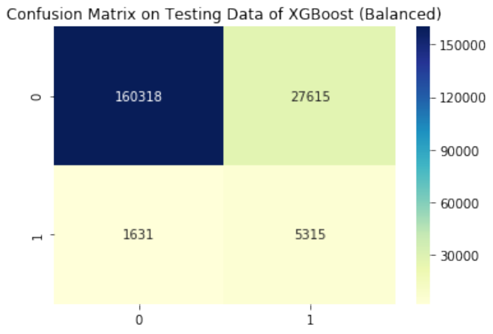

Figure 11: Confusion Matrix of XGBoost on Balanced Data

ROC Curve:

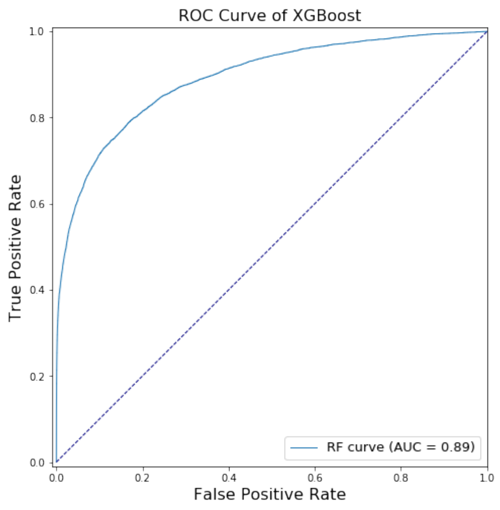

Figure 12: ROC Curve of XGBoost on Balanced Data

5.4 Light Gradient Boosting Machine (LGBM)

5.4.1 Imbalanced

The accuracy for the dataset without balancing:
Train Data:

97.9%

Test Data:

97.7%

Confusion Matrix:

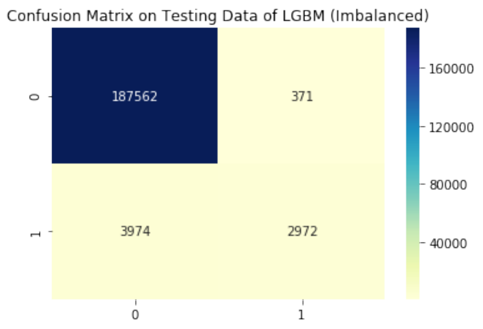

Figure 13: Confusion Matrix of LGBM on Imbalanced Data

5.4.2 Balanced

The accuracy for the dataset with balancing by setting the function 'is_unbalance' equal to 'True':
Train Data:

89.1%

Test Data:

88.9%

Confusion Matrix:

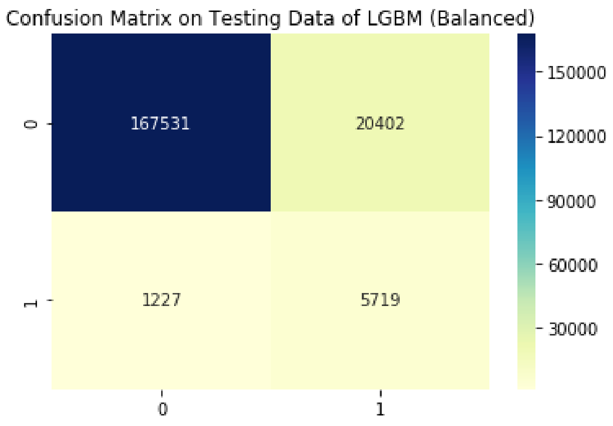

Figure 14: Confusion Matrix of LGBM on Balanced Data

ROC Curve:

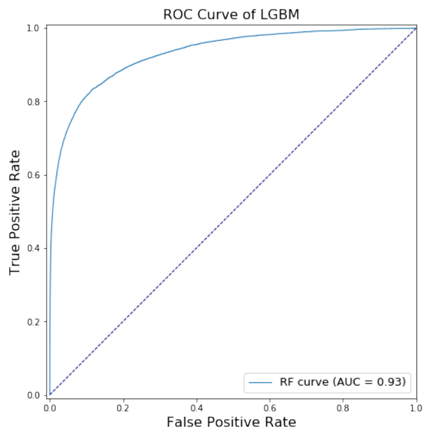

Figure 15: ROC Curve of LGBM on Balanced Data

In all the above models, the accuracy scores in the case of imbalanced data, are too high as the data is highly skewed towards one class. This can be verified by looking at the confusion matrices of the same. Hence, it is required to use different balancing techniques to get the right results.

6 Conclusion

From the above results, the AUC Scores for each model are: 0.79 (Logistic Regression), 0.81 (MLP), 0.89 (XGBoost), 0.93 (LGBM). This shows that Light Gradient Boosting Machine gives the best results than other models used in our project. Moreover, it also shows that in case of imbalanced data, the model used can overfit and it is important to apply balancing techniques like SMOTE or AdaSyn and setting objective functions like 'is_unbalance' 'scale_pos_weight' in models like XGBoost and LGBM. This will help to reduce the overfitting of the models and give out the right results.