

Collaborative group project on Credit Card Fraud Detection

ALY-6020 Predictive Analytics

Fall 2021

**Submitted by – (Group 2)**

Srinidheesh Ranganathan

Hemakumar Jabbireddy

Yongcheng Li

10/31/2021

Project Task Status:

1. Summary of the data set, check for missing, n/a values and imbalanced data.

2. Data visualizations.

3. Building the model for an imbalanced data set.

4. Data Preparation.

5. Predictive Models

I. Decision Tree

II. XG boost

|  |  |  |
| --- | --- | --- |
| **MODULES** | **TASKS** | **STATUS** |
| Module 2 | Data Selection and Group project proposal | Completed |
| Module 3 | First Prediction Model | Complete |
| Module 4 | Second Prediction Model | Completed |
| Module 5 | Final Report and Group Presentation | Completed |

Executive Summary:

Payment fraud represents a significant and growing issue in the world. With the rise in computing platforms, the scale and diversity of credit card fraud have significantly changed. This is due to the rise in both online transactions and e-commerce platforms.

Credit card fraud happens when a credit/debit card or card information is stolen, or even when the fraudster uses the information for his/her personal gains. To control these fraudulent activities, fraud detection systems were introduced. But such systems pose operational challenges because the responsibility of the management and cybersecurity would be uncoordinated sometimes. And moreover, the design of such systems is particularly challenging due to the non-stationary distribution of data.

The issue most enterprises face here is the lack of incident data, as there is limited information on smaller attacks as in most cases they are not reported thoroughly. Through this project, we aim to implement and assess the performance of various machine learning models on the dataset to successfully predict fraudulent transactions. Since public data are scarce due to confidentiality, the focus of the project is on predictive performance rather than inference.

Introduction:

In this project, we use a rich dataset retrieved from Kaggle that contains 284,807 credit card transactions occurring over two days in Europe. It was collected and also analyzed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection.

The dataset contains over 31 variables with nearly 284,807 credit card transactions. An important attribute of the dataset is that it has been processed to protect cardholder privacy. Because of privacy concerns, we cannot provide the original features and more background information about the data. This suggests that the data is substantially imbalanced. Positive frauds account for 0.172 percent of total transactions. We only have the following features V1 through V28, which are referred to as the primary components, because it involves confidential data. Aside from that, we've been given time and a transaction amount.

Another issue to overcome is the dataset's extreme imbalance. With a large number of non-fraudulent transactions in place, Random Undersampling can be used to reduce the number of non-fraudulent transactions and match it to measure the number of fraudulent transactions.

Business Challenge:

For every credit card organization, detecting fraud transactions is crucial. Any business needs to detect potential frauds so that clients aren't charged for products they didn't buy. The goal is to create a classifier that can determine whether a transaction is fraudulent or not.

First Prediction Model

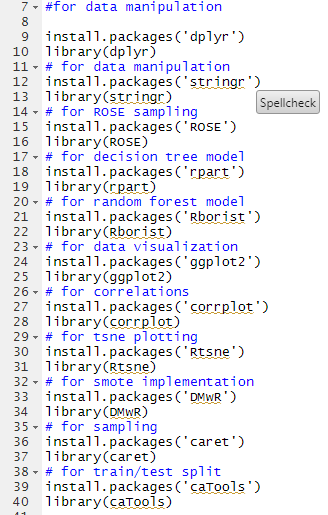
Implementation Approach:

Data Dictionary:

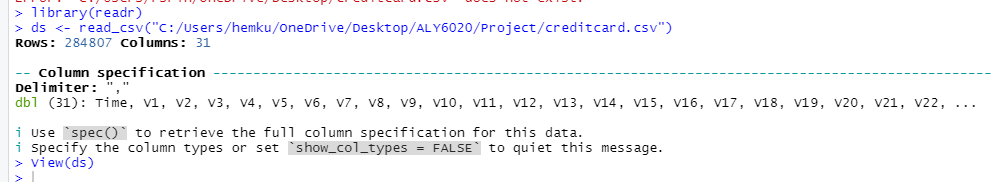
* V1 - V28 - 28 principal components based on an unknown set of data features that contain knowledge about every transaction.
* Time: contains the seconds passed between every transaction and the initial transaction in the dataset.
* Amount: the transaction Cost, this feature can be utilized for example-dependent cost-sensitive learning.
* Class: the response variable, and it takes value 1 in case of fraud and 0 contrarily.

Data Preprocessing:

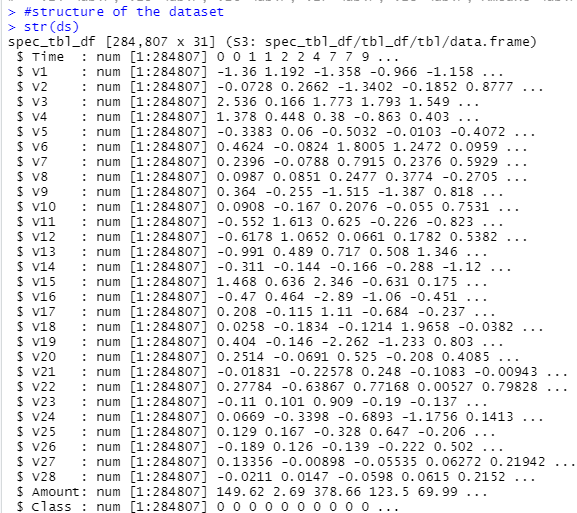
First, we have installed and loaded all the required libraries for the analysis.



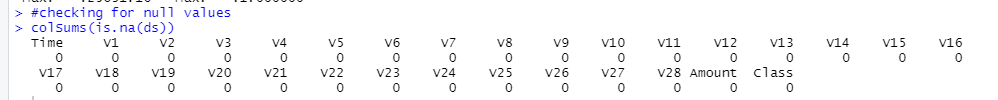
Then, the dataset is loaded into R using the library ‘readr’.



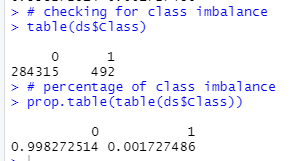
The structure of the dataset shows that the dataset contains 284,807 rows and 31 columns. All the variables are in Numeric.

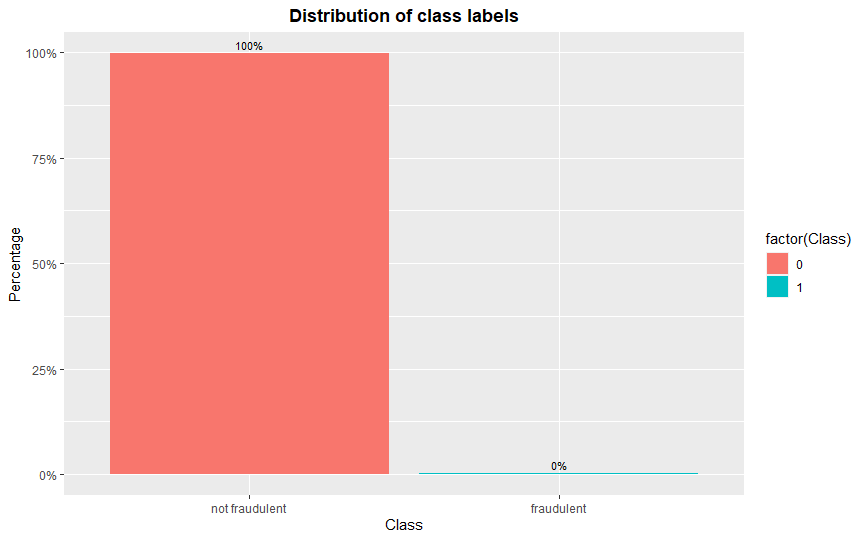


There are no missing values in the dataset.



The dataset is clearly skewed, with non-fraudulent transactions accounting for 99.8% of all transactions. Even a classifier that diagnoses all transactions as non-fraudulent will have above 99 percent accuracy, thus a simple measure like accuracy isn't relevant here. AUC would be a good indicator of model performance here (Area Under the Precision-Recall Curve).

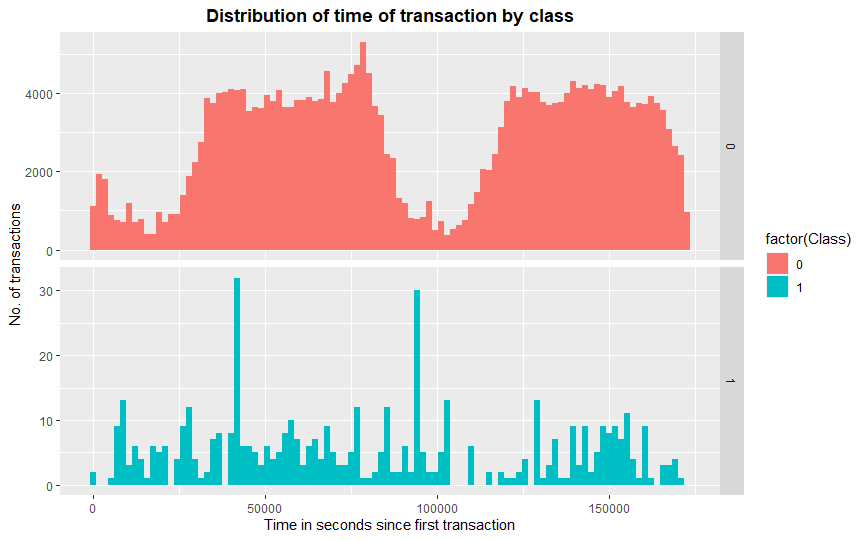




Exploratory data analysis:

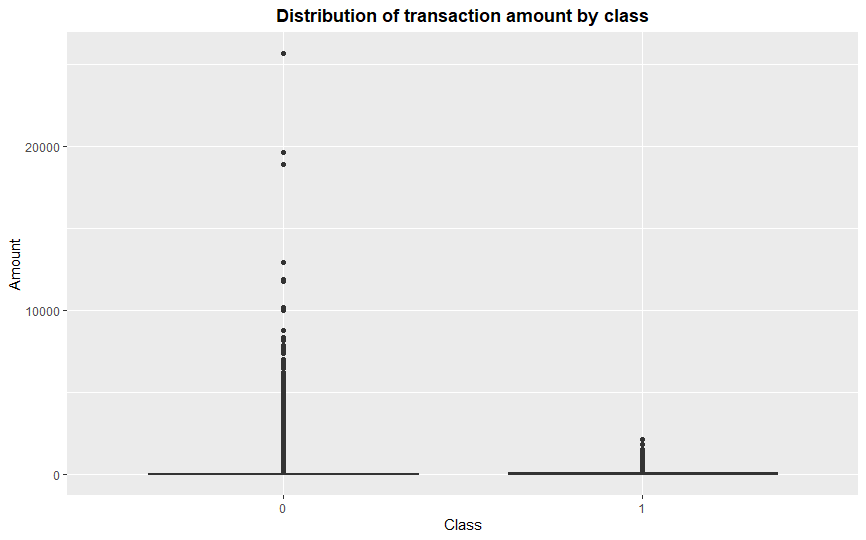
We will obtain insights into the dataset using data visualizations after the data has been explored and preprocessed. To form valuable inferences, we will use the correlation matrix, bar plot, and histogram to find the patterns in our data. On all of the numerical values, the correlation matrix has been shown.

Histogram of variable Time by Class:



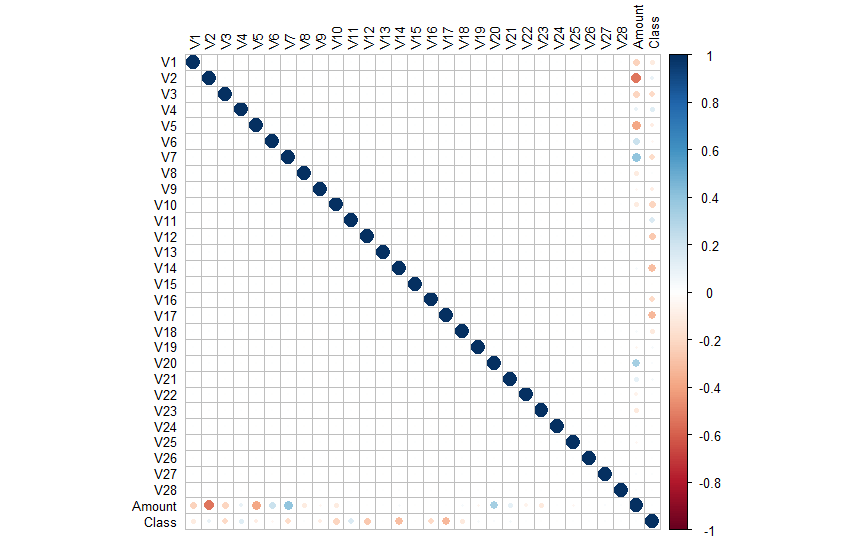
The ‘Time' feature appears to be the same in both sorts of transactions. It might be argued that fraudulent transactions are more evenly dispersed, whereas legitimate transactions are cyclical.

Boxplot of transaction amount by class:



For non-fraudulent transactions, there is definitely a lot more variation in transaction values.

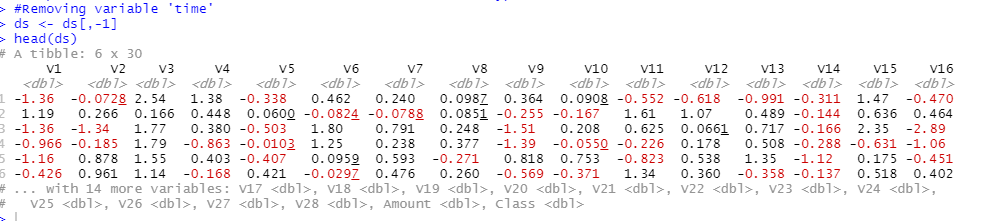
Correlation Matrix:



The correlation matrix determines the degree of association between the numerical variables present in the data. We see that the majority of the data aspects are unrelated, and this is because most of the features were subjected to a Principal Component Analysis (PCA) technique before publication. After propagating the basic features using PCA, V1 through V28 are most likely the Principal Components.

Data Preparation:

The 'Time' variable does not indicate the transaction's real-time but instead lists the facts in chronological order. We assume that the 'Time' attribute has little or no value incorrectly categorizing a fraud transaction based on the data visualization above. Hence, we exclude this column from the future study.



Before splitting the dataset into train and test, we made sure that all the variables are in numeric form. So, we changed the class variable to factors and scaled all variables before model building.

Data Analysis and Results:

Analysis by Hemakumar Jabbireddy:

Decision Tree:

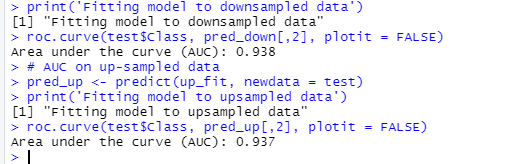
Because of the unequal distribution of the dependent variable, ML algorithms struggle with accuracy. As a result, existing classifiers' performance is skewed toward the majority class. The algorithms are accuracy-driven, reducing total error, with the minority class playing a minor role. The data set is assumed to have balanced class distributions using ML techniques. The techniques used to solve this problem are known as 'Sampling Methods.' In general, these strategies try to use some process to transform an unbalanced data distribution into a balanced distribution. The change is made by reducing the size of the original data set while maintaining the same amount of balance. The approaches utilized to deal with the unbalanced dataset are listed below.

Undersampling:

To balance the data collection, this method minimizes the number of observations from the dominant class. This method is best to utilize when the data set is significant, and reducing the number of training samples helps to reduce run time and storage issues. The removal of observations may cause the training data to lose key information about the majority class, which could be an issue for this method.

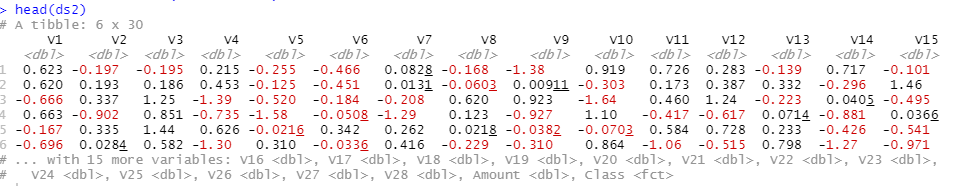
Oversampling:

This method is effective with minorities. To balance the data, it duplicates observations from the minority class. Upsampling is another term for it. Using this method has the advantage of not causing any data loss. The downside of this strategy is that, because it just adds duplicated observations from the original data set, it ends up adding many observations of various sorts, resulting in overfitting.

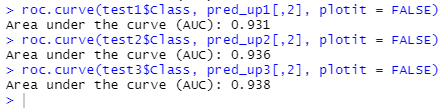


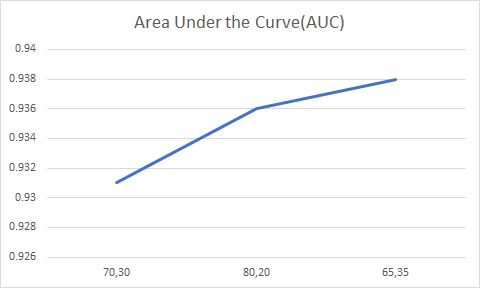
Here, both the downsampling and Upsampling methods have almost equal AUC scores. Since both the sampling techniques yielded the same AUC score, I choose to proceed with Upsampling method as it doesn’t cause any data loss.

I choose the data samples from 95001 to 190,000 for building the decision tree model.



First, I divided the data set into train and test of 70:30 and got an AUC score of 0.931. Secondly, I divided the data set into train and test set of 80:20 and got an AUC score of 0.936 and finally I divided the dataset into train and test set of 65:35 and got an AUC score of 0.938. From this, we can conclude that the decision tree classifier has given accurate results.

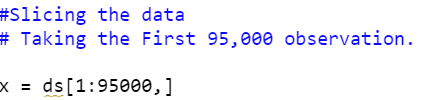




Analysis by Srinidheesh Ranganathan:

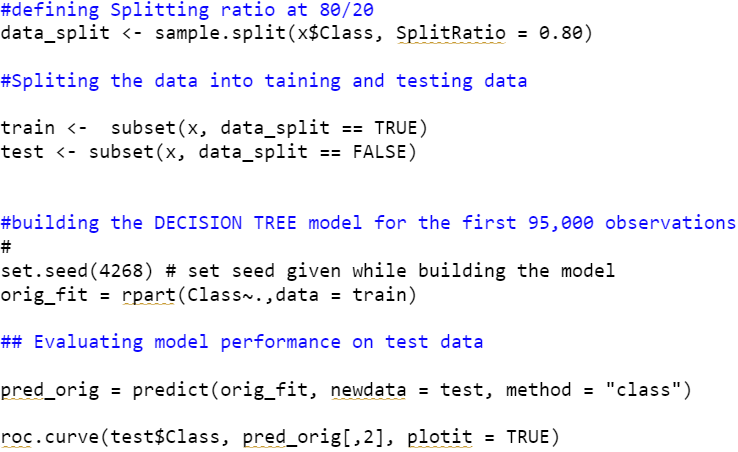
Decision Tree:

Tree based models laid the foundation for many classical machine learning algorithms like Random Forest, boosted Decision Trees, Bagging etc. They can be used for any type of data. numerical, categorical and so on. As they are easier to represent visually, we can make complex prediction models much simpler to interpret. We represent the data as a tree where each internal node denotes a test on an attribute, and each branch represents the outcome of the test, and the leaf nodes hold the class label. We sliced the original dataset into three parts, from which I built a decision tree using the first 95,000 observations.

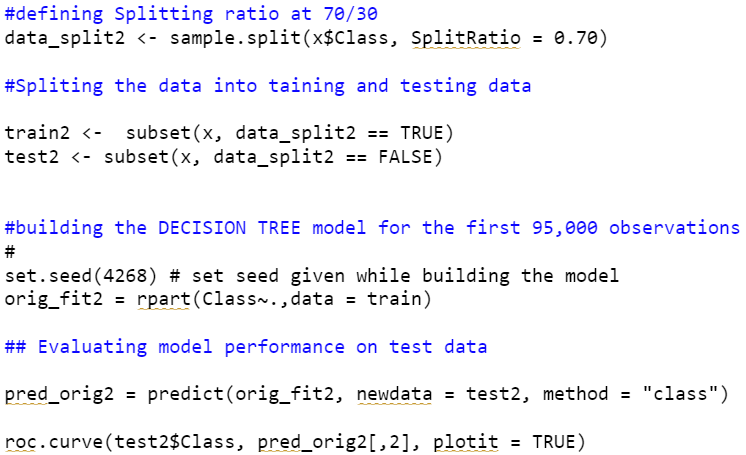


After slicing to the first 95,000 observations, we split the data into training and testing at a split ratio of 80:20; 70:30; 65:35. The set.seed() function is given because we want to get a reproducible random result. It is essential as we are optimizing the function that involves randomly generated numbers.

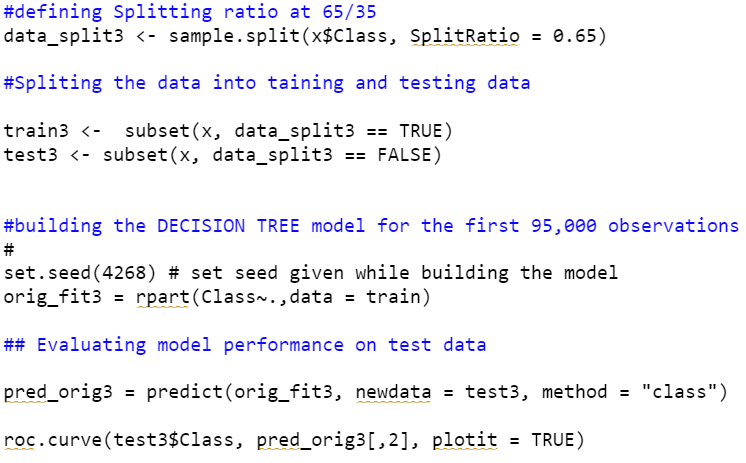
At 80:20,

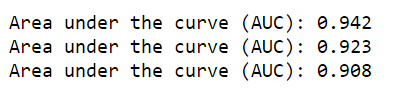


At 70:30



At 63:35,





The Decision tree for the first 95000 observations was built and the model’s performance was evaluated to help us judge and compare the performance when multiple seed values were inputted. When a different frequency of split ratio was given, we acquired the following AUC scores. Since the AUC scores are high, we can conclude that the model yields accurate results and it is positively skewed.

|  |  |  |  |
| --- | --- | --- | --- |
| **Ratios** | **80:20** | **70:30** | **65:35** |
| **AUC** | 0.942 | 0.923 | 0.908 |

Analysis by Yongcheng Li:

Decision tree model is a popular tool when analytics implement the classification analysis. It is a powerful algorithm to perform both classification tasks, capable of fitting complex dataset like what we are discovering now- “credit card fraud detection”. Since the features of the dataset are complex and large amount based, it is too difficult to generate a single model on whole dataset, due to the residual error. Thus, we have split the dataset into three different subsets.

In this part of analysis, we pick up the dataset from 19000 rows to 284807 rows.

图片包含 文本

描述已自动生成

After the split, we have last 95000 observations, then we need to separate data into train and test based on three different split ratio which are 0.70, 0.80 and 0.65.

文本

描述已自动生成

Since the sampling are picked up from different ratio, the sample size is an important factor that led to the accuracy of the model, we could have three different results on the decision model.

文本

描述已自动生成

We have AUC 1 at 0.936, and AUC2 at 0.957, AUC3 at 0.941.

Discussion:

The decision tree was built separately on the first 95,000 observations and the next 95000, and the final 94807 observations. In the first case, the AUC scores were the highest at the split ratio of 80:20. And in the second 95000 observations, the AUC was highest when the split ratio was 65:35 and in the last 94807 observations, the AUC was highest at 80:20 split ratio. Since two out of three analyses attained highest AUC score at 80:20, we recommend splitting the training and testing data at the 80:20 ratio to yield the best results.

Second Prediction Model

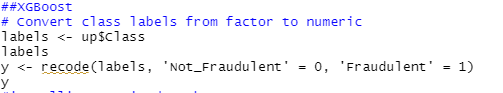
Data Analysis and Results

Analysis by Hemakumar Jabbireddy

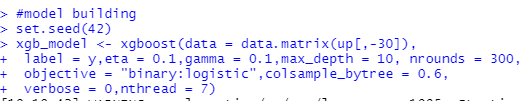
XGBoost:

XGBoost is a supervised learning algorithm that uses training data to predict a target variable. XGBoost is a high-speed, and high-performance implementation of gradient boosted decision trees.

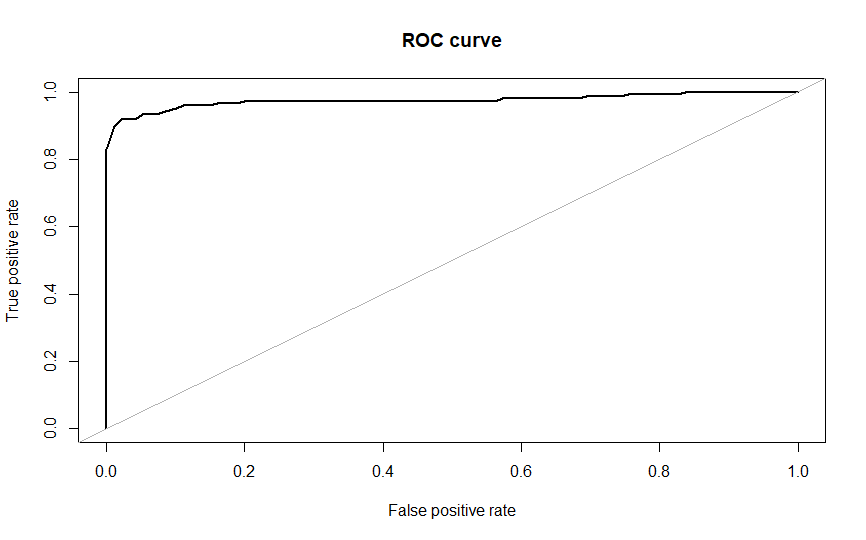
First, we converted the class label from factor to numeric.



Model building:

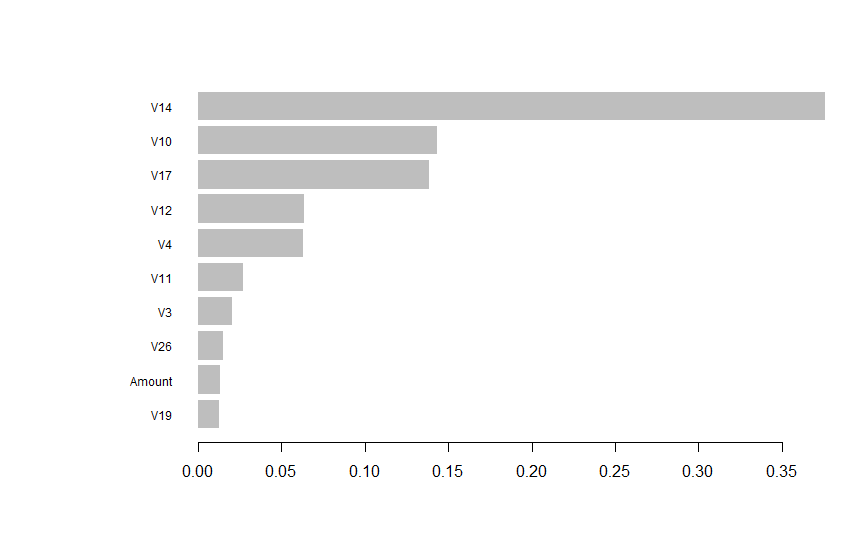






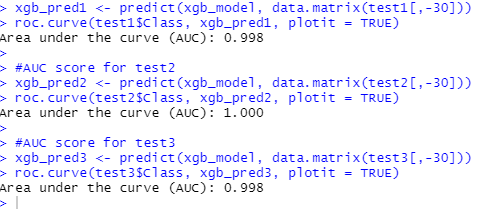
The XGBOOST model performed the best, with an auc score of 0.975, however Decision tree models also performed well.

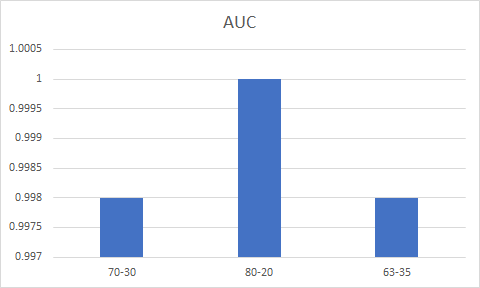
Feature importance:



V14 has got the highest importance followed by v10 and v17. The feature V19 and Amount has got the least importance. Clearly, we can say that the feature V14 is very important in determining credit fraud.XGBoost for samples 95001 to 190,000:

XGBoost for samples 95001 to 190,000:





Here, I divided the data set into a train and test of 70:30 and got an AUC score of 0.998. Secondly, I divided the data set into train and test set of 80:20 and got an AUC score of 1 and finally I divided the dataset into train and test set of 65:35 and got an AUC score of 0.998. From this, we can conclude that the split ratio of 80:20 has yielded a better AUC score than others and for this sample 80:20 split would be ideal to go with.

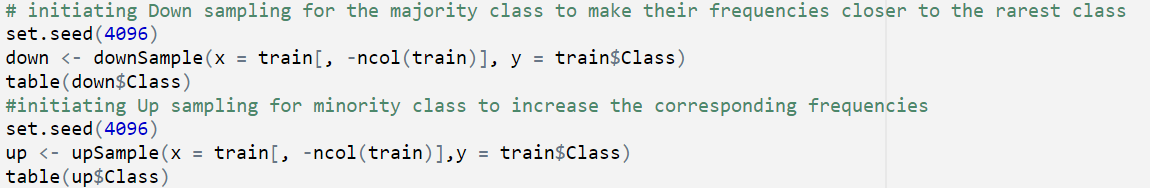
Analysis by Srinidheesh Ranganathan

XG boost:

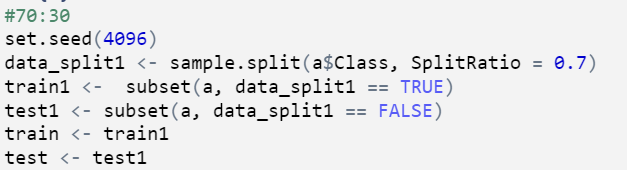
Boosting is an ensemble learning technique that builds a strong classifier from various weak classifiers in series. Such algorithms play an important role in dealing with bias-variance trade-off. XGBoost or extreme gradient boosting is a well-known optimized distributed library designed to implement scalable machine learning system for tree boosting. It comes under gradient boosting framework. It is used in regression and classification problems as it supports parallel processing to solve data science problems in a fast and efficient manner. Apart from its efficient memory management and cache optimization, it also has a different kind of regularizations to help reduce overfitting of data.

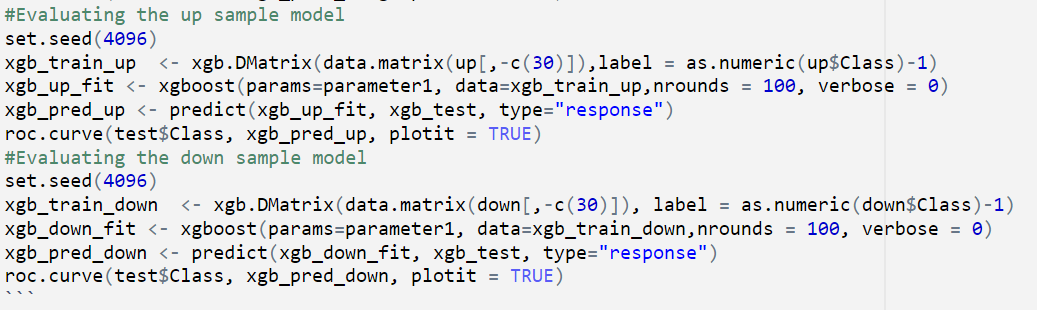
The analyses start with the same process as mentioned in the first prediction model. Along with the XGboost package, we import the necessary packages and remove the unnecessary variables. We split the data into training and testing at the split ratios of 70:30; 80:20; and 65:35.

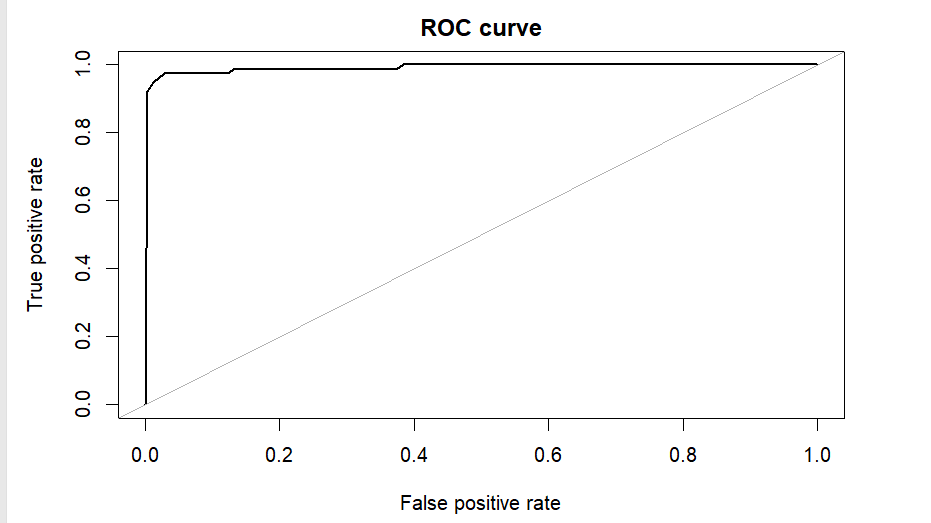
We initiate the Up Sampling for minority class to increase the corresponding frequencies. And the Down Sampling technique on the data for majority class to make their frequencies closer to the rarest class and evaluate the performance of the model.



At 70:30,







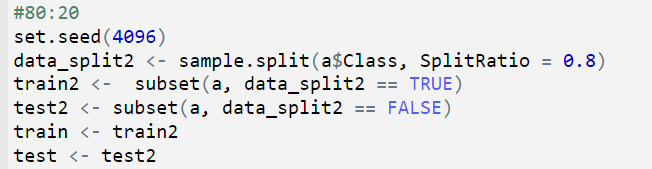






At 70:30, the original data set got an AUC of 0.985, while the AUC through Up Sampling technique and Down sampling technique were 0.994.

At 80:20,



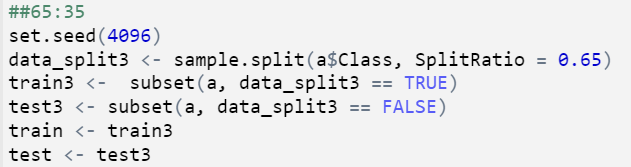






At 80:20, the original data set got an AUC of 0.983 while the AUC through Up Sampling technique was 0.995 and Down sampling technique were 0.988.

At 65:35









At 65:35, the original data set acquired an AUC of 0.986 while the AUC through Up Sampling was and Down sampling were 0.996. While the best scores were achieved using sampling techniques, the XGboost on the original data also performed decently and yielded closer results.

Analysis by Yongcheng Li

XGboost

XGboost is another model choice based on decision tree, which can improve the effectiveness of our prediction on dataset. Before the modeling, we took the same steps as the first model, set up train and test dataset by splitting ratio at 70-30, 80-20, 65-35. Also, we use up sampling and down sampling to see how the results vary among different sampling techniques.

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

Then we could use “xgb.DMatrix” to build up xgboost model and evaluate the model by AUC and confusion matrix.

图表

中度可信度描述已自动生成

文本, 信件

描述已自动生成

This is one of the examples of evaluation by ROC curve and confusion matrix.

文本

描述已自动生成

This is the table that shows all the results we have built by XGboost on different sampling techniques. After all the above work, this study mainly focuses on building appropriate models to predict the class of fraudulent apart with the class of non-fraudulent based on classification trees and Boosting models. To figure out which model performs best in prediction task, various of measures are used which includes AUC, sensitivity, specificity, precision, recall and F1 score. Models with highest of these measures are considered to the best models. And this study uses two types of data, the first one is the whole data and the second is a subset of 1/3 data, for both two types of data, 70% randomly selected is used as training and the left used as testing part, also, for each training set, up and down sampling methods are used to make the response's classes to be balanced. At last, for the 1/3 data, different randomly splits ratios are used which are 0.65, 0.7 and 0.8.

All of the performances are shown in the final table, and from the results, it can be found that all models show high accuracy, specificity and AUC values, however, the sensitivity, recall and F1 score values are low using up and down samplings. And among all results, it seems the models XGboost model using the whole original data performs relatively better than other models which show high AUC (over 0.9) and high sensitivity values (close to 0.9) at the same time, so for future data, to predict the class of Fraudulent or not, this study suggest using XGboost model with whole original data without up and down samplings.

Discussion:

After performing the XGBoost on three different set of observations, we concluded that XGboost is faster and efficient than decision trees as it provides the highest AUC scores. The best suited split ratio is 65:35, as it provided the highest results in most cases. It is also worthy to note that by further tuning the parameters of the XGBOOST model, we can achieve even better performance.

Conclusion:

Through this project we have tried to show different methods of dealing with unbalanced datasets like the fraudulent credit card transactions where the instances of fraudulent cases are few compared to the instances of normal transactions. We used the metric AREA UNDER ROC CURVE to evaluate how different methods of oversampling or under sampling the response variable can lead to better model training. We built a decision tree and XG boost to detect fraudulent transactions and based on the performance of the two models, XGboost received the highest score and best suited for imbalanced data.

References:

*1.* *Credit Card Fraud Detection*. (2018, March 23). Kaggle.<https://www.kaggle.com/mlg-ulb/creditcardfraud>

*2.* Sun, L. (2020, October 16). *Credit Card Fraud Detection - Towards Data Science*. Medium.<https://towardsdatascience.com/credit-card-fraud-detection-9bc8db79b956>