Report on Credit Card Default –Classification of Unbalanced Data

|  |  |
| --- | --- |
| Version: | 0.1 |
| Date: | 9 April, 2017 |
| Author | Meenal Srivastava |
| Guide |  |

# Contents

[1 Contents 2](#_Toc482792798)

[1.1 List of Tables 2](#_Toc482792799)

[1.2 List of Figures 3](#_Toc482792800)

[2 Document Statements 4](#_Toc482792801)

[2.1 Purpose 4](#_Toc482792802)

[2.2 Assumptions and Dependencies 4](#_Toc482792803)

[2.3 In Scope 4](#_Toc482792804)

[2.4 Out of Scope 5](#_Toc482792805)

[2.5 Audience 5](#_Toc482792806)

[3 Introduction 6](#_Toc482792807)

[3.1 Project Overview 6](#_Toc482792808)

[3.2 Problem Statement 6](#_Toc482792809)

[3.3 Methodology 7](#_Toc482792810)

[3.4 Metrics 7](#_Toc482792811)

[4 Analysis 8](#_Toc482792812)

[4.1 The Credit Card Fraud detetction Data Set 8](#_Toc482792813)

[4.1.1 Organization of dataset 8](#_Toc482792814)

[4.1.2 Data Exploration 8](#_Toc482792815)

[4.1.3 Data Pre-processing 21](#_Toc482792816)

[4.2 Classification with Undersampling 31](#_Toc482792817)

[4.2.1 Logistic Regression 31](#_Toc482792818)

[4.2.2 Support Vector machines 35](#_Toc482792819)

[4.2.3 Decision Trees 38](#_Toc482792820)

[4.2.4 Random Forest 40](#_Toc482792821)

[4.2.5 KNN Model 40](#_Toc482792822)

[4.2.6 Neural Network Model 41](#_Toc482792823)

[4.2.7 Treebag Model 42](#_Toc482792824)

[4.3 Classification with Smote Sampling 43](#_Toc482792825)

[4.4 Ensembling 50](#_Toc482792826)

[4.5 Results 53](#_Toc482792827)

[4.6 Further work 54](#_Toc482792828)

## List of Tables

[Table 1 : Comparison of Sensitivity& Specificity for all Classification Algorithms for SMOTE sampling 53](#_Toc482792829)

[Table 2 : Comparison of Sensitivity& Specificity for all Classification Algorithms for under sampling 54](#_Toc482792830)

## List of Figures

[Figure 1: Genuine Vs Fraud cases Distribution 11](#_Toc482792831)

[Figure 2: Distribution of ‘Time’ variable 12](#_Toc482792832)

[Figure 3: Distribution of ‘Time’ variable wrt ‘Class’ 12](#_Toc482792833)

[Figure 4 : Distribution of ‘Time’ variable for Genuine Cases 13](#_Toc482792834)

[Figure 5 : Distribution of ‘Time’ variable for Fraud Cases 13](#_Toc482792835)

[Figure 6 : Distribution of ‘Amount’ variable 14](#_Toc482792836)

[Figure 7 : Distribution of ‘log (Amount)’ variable 14](#_Toc482792837)

[Figure 8 : Distribution of ‘Amount’ variable for Fraud Cases 15](#_Toc482792838)

[Figure 9 : Distribution of ‘log (Amount)’ variable for Genuine Cases 15](#_Toc482792839)

[Figure 10 : Distribution of ‘Amount’ variable for Genuine Cases 15](#_Toc482792840)

[Figure 11 : Distribution of ‘log (Amount) ’ variable for Genuine Cases 16](#_Toc482792841)

[Figure 12: Boxplot of Amount wrt Classes 16](#_Toc482792842)

[Figure 13 : Boxplot of ‘log (Amount)’ wrt to Classes 17](#_Toc482792843)

[Figure 14 : Synthetic Minority Oversampling Algorithm 23](#_Toc482792844)

[Figure 15: Generation of Synthetic Instances with the help of SMOTE 24](#_Toc482792845)

[Figure 16 : Approach to Ensemble based Methodologies 25](#_Toc482792846)

[Figure 17 : Approach to Bagging Methodology 26](#_Toc482792847)

[Figure 18: Boosting 27](#_Toc482792848)

[Figure 19: Adaptive Boosting 28](#_Toc482792849)

[Figure 20 ; Gradient Boosting 30](#_Toc482792850)

[Figure 21 : Variable Importance as per decision tree 31](#_Toc482792851)

[Figure 22: Confusion Matrix 32](#_Toc482792852)

[Figure 23 :ROC Curve for full data for Logistic regression with Under sampling 35](#_Toc482792853)

[Figure 24 : ROC Curve for full data with logistic regression with smote sampling 44](#_Toc482792854)

# Document Statements

The following sections describe the content of the document in terms of the following:

* Purpose. Explains why the document has been produced, providing a summary of the reasons and goals
* Assumption. Something held to be true to allow a project to proceed
* Dependency. A mandatory output from one project or piece of work that is required as an input for another project or piece of work
* In Scope. A summary of the areas covered by the document
* Out of Scope. Explicitly identifies any areas that are not covered by the document
* Audience. The type of reader expected to use this document
* Conventions. Typographical conventions used to clearly identify specific types of information

## Purpose

The purpose of this document is to demonstrate use of classification in predicting defaults in credit card customers.

## Assumptions and Dependencies

The following assumptions have been made in the writing of this document:

* Dataset “ Credit Card Default” has been taken assuming that the data available in train dataset has been correctly labelled and can be used to train the models.
* It is assumed that the dataset can be clearly demarcated and grouped under 2 categories based on the predictors.

## In Scope

The scope of this document is:

* Analysis of the “Credit Card Default” data
* Applying commonly used classification algorithms on the data
* Evaluation metrics involved in classification algorithms
* Comparing and evaluating methods for handling unbalanced data
* Evaluating ensemble (boosting and bagging) techniques on the unbalanced dataset

## Out of Scope

The following items are considered outside the scope of this document:

* As this document mainly deals with techniques for handling unbalanced data, fine tuning of the algorithms involved in classification, which may further enhance the results, was not done.

## Audience

Anyone who is interested in the techniques of handling unbalanced data, can benefit from this document.

# Introduction

## Project Overview

This project is about classifying transactions as fraud or genuine based on the transaction details. This is a case of skewed or unbalanced data as the percentage of fraud transactions is very low as compared to genuine transactions.

This project will focus on

1. supervised machine learning algorithms for automatic classification of transactions in genuine or fraud categories where some external mechanism (such as human feedback) provides information on the correct classification on the training set and
2. sampling techniques for handling skewed dataset

I will apply these classification algorithms on Credit card dataset which is a collection of 284,807 transactions made by credit cards in September 2013 by European cardholders. The transactions will be classified according to their details.

The idea is to test different methods on skewed data to compare if pre-processing techniques work better when there is an overwhelming majority class that can disrupt the efficiency of our predictive model or ensemble method can work fine on such datasets.

**My intention is to create models using:**

1. Logistic Regression
2. Decision trees
3. Bagged tree
4. SVMs
5. Neural networks
6. Ensembles

## Problem Statement

The datasets 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.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, there is no information about the original features and background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

## Methodology

I will explore the dataset in the beginning of the training part .

I will divide the dataset into two: hold out sample and the training sample(30/70 ratio)

For classification, I will work on two methodologies

1. For the first methodology, I will train the algorithms on balanced train dataset, evaluate its performance on the balanced cross validation dataset and then apply the trained algorithm on the unbalanced data.

For balancing the dataset, I will use below techniques and compare the performance based on the metrics

* 1. Under sampling
  2. SMOTE
  3. Tomek Links with smote

1. In the second approach, I will train the algorithm on unbalanced training dataset. I will use learners and combine them and will evaluate the performance of ensemble techniques in classification. I will work on below techniques.
   1. Bagging
   2. Boosting-GBM, XGBoost, Adaboost
   3. Weak learner Ensemble techniques
      1. Majority Vote
      2. Algorithm based ensembes
         1. Top layer algorithm used :Logistic regression
         2. Top layer algorithm used :GBM

Classification can be hierarchical and flat and also one or multi label. This is an example of **flat classification with one label** for each transactions**.**

## Metrics

This is a binary classification problem. Given the class imbalance ratio, accuracy cannot be the criterion. So I will measure the accuracy using the Area Under the Precision-Recall Curve (AUPRC). Confusion matrix accuracy is not meaningful for unbalanced classification.

1. Recall: which indicates how many of the relevant items that we identified,

Recall=TP/(TP+FN)

1. The F-Measure (or F-Score), which combines the precision and recall to give a single score, is defined to be the harmonic mean of the precision and recall:

(2 × *Precision* × *Recall*) / (*Precision* + *Recall*).

# Analysis

## The Credit Card Fraud detetction Data Set

### Organization of dataset

The data is organized into two classes, genuine and fraud transactions based on 31 variables

### Data Exploration

#### Importing the dataset in ‘R’

1. Importing the training dataset in R and studying its structure, we find there are 284807 observations of 31 variables.

Classes ‘data.table’ and 'data.frame': 284807 obs. of 31 variables:

$ Time : num 0 0 1 1 2 2 4 7 7 9 ...

$ V1 : num -1.36 1.192 -1.358 -0.966 -1.158 ...

$ V2 : num -0.0728 0.2662 -1.3402 -0.1852 0.8777 ...

$ V3 : num 2.536 0.166 1.773 1.793 1.549 ...

$ V4 : num 1.378 0.448 0.38 -0.863 0.403 ...

$ V5 : num -0.3383 0.06 -0.5032 -0.0103 -0.4072 ...

$ V6 : num 0.4624 -0.0824 1.8005 1.2472 0.0959 ...

$ V7 : num 0.2396 -0.0788 0.7915 0.2376 0.5929 ...

$ V8 : num 0.0987 0.0851 0.2477 0.3774 -0.2705 ...

$ V9 : num 0.364 -0.255 -1.515 -1.387 0.818 ...

$ V10 : num 0.0908 -0.167 0.2076 -0.055 0.7531 ...

$ V11 : num -0.552 1.613 0.625 -0.226 -0.823 ...

$ V12 : num -0.6178 1.0652 0.0661 0.1782 0.5382 ...

$ V13 : num -0.991 0.489 0.717 0.508 1.346 ...

$ V14 : num -0.311 -0.144 -0.166 -0.288 -1.12 ...

$ V15 : num 1.468 0.636 2.346 -0.631 0.175 ...

$ V16 : num -0.47 0.464 -2.89 -1.06 -0.451 ...

$ V17 : num 0.208 -0.115 1.11 -0.684 -0.237 ...

$ V18 : num 0.0258 -0.1834 -0.1214 1.9658 -0.0382 ...

$ V19 : num 0.404 -0.146 -2.262 -1.233 0.803 ...

$ V20 : num 0.2514 -0.0691 0.525 -0.208 0.4085 ...

$ V21 : num -0.01831 -0.22578 0.248 -0.1083 -0.00943 ...

$ V22 : num 0.27784 -0.63867 0.77168 0.00527 0.79828 ...

$ V23 : num -0.11 0.101 0.909 -0.19 -0.137 ...

$ V24 : num 0.0669 -0.3398 -0.6893 -1.1756 0.1413 ...

$ V25 : num 0.129 0.167 -0.328 0.647 -0.206 ...

$ V26 : num -0.189 0.126 -0.139 -0.222 0.502 ...

$ V27 : num 0.13356 -0.00898 -0.05535 0.06272 0.21942 ...

$ V28 : num -0.0211 0.0147 -0.0598 0.0615 0.2152 ...

$ Amount: num 149.62 2.69 378.66 123.5 69.99 ...

$ Class : int 0 0 0 0 0 0 0 0 0 0 ...

Apart from time, and amount, all the rest 28 variable are principal components and we do not have any information to make any sense out of them as in how they are related to the outcome or response variable, Class( whether the transaction was genuine or fraud)

1. Looking at the first 6 observations of the training data

Time V1 V2 V3 V4 V5

1: 0 -1.3598071 -0.07278117 2.5363467 1.3781552 -0.33832077

2: 0 1.1918571 0.26615071 0.1664801 0.4481541 0.06001765

3: 1 -1.3583541 -1.34016307 1.7732093 0.3797796 -0.50319813

4: 1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888

5: 2 -1.1582331 0.87773676 1.5487178 0.4030339 -0.40719338

6: 2 -0.4259659 0.96052304 1.1411093 -0.1682521 0.42098688

V6 V7 V8 V9 V10

1: 0.46238778 0.23959855 0.09869790 0.3637870 0.09079417

2: -0.08236081 -0.07880298 0.08510165 -0.2554251 -0.16697441

3: 1.80049938 0.79146096 0.24767579 -1.5146543 0.20764287

4: 1.24720317 0.23760894 0.37743587 -1.3870241 -0.05495192

5: 0.09592146 0.59294075 -0.27053268 0.8177393 0.75307443

6: -0.02972755 0.47620095 0.26031433 -0.5686714 -0.37140720

V11 V12 V13 V14 V15 V16

1: -0.5515995 -0.61780086 -0.9913898 -0.3111694 1.4681770 -0.4704005

2: 1.6127267 1.06523531 0.4890950 -0.1437723 0.6355581 0.4639170

3: 0.6245015 0.06608369 0.7172927 -0.1659459 2.3458649 -2.8900832

4: -0.2264873 0.17822823 0.5077569 -0.2879237 -0.6314181 -1.0596472

5: -0.8228429 0.53819555 1.3458516 -1.1196698 0.1751211 -0.4514492

6: 1.3412620 0.35989384 -0.3580907 -0.1371337 0.5176168 0.4017259

V17 V18 V19 V20 V21

1: 0.20797124 0.02579058 0.40399296 0.25141210 -0.018306778

2: -0.11480466 -0.18336127 -0.14578304 -0.06908314 -0.225775248

3: 1.10996938 -0.12135931 -2.26185709 0.52497973 0.247998153

4: -0.68409279 1.96577500 -1.23262197 -0.20803778 -0.108300452

5: -0.23703324 -0.03819479 0.80348692 0.40854236 -0.009430697

6: -0.05813282 0.06865315 -0.03319379 0.08496767 -0.208253515

V22 V23 V24 V25 V26

1: 0.277837576 -0.11047391 0.06692808 0.1285394 -0.1891148

2: -0.638671953 0.10128802 -0.33984648 0.1671704 0.1258945

3: 0.771679402 0.90941226 -0.68928096 -0.3276418 -0.1390966

4: 0.005273597 -0.19032052 -1.17557533 0.6473760 -0.2219288

5: 0.798278495 -0.13745808 0.14126698 -0.2060096 0.5022922

6: -0.559824796 -0.02639767 -0.37142658 -0.2327938 0.1059148

V27 V28 Amount Class

1: 0.133558377 -0.02105305 149.62 0

2: -0.008983099 0.01472417 2.69 0

3: -0.055352794 -0.05975184 378.66 0

4: 0.062722849 0.06145763 123.50 0

5: 0.219422230 0.21515315 69.99 0

6: 0.253844225 0.08108026 3.67 0

1. Looking at the summary of the training dataset after changing the ‘Class’ variable to factor

Time V1 V2

Min. : 0 Min. :-56.40751 Min. :-72.71573

1st Qu.: 54202 1st Qu.: -0.92037 1st Qu.: -0.59855

Median : 84692 Median : 0.01811 Median : 0.06549

Mean : 94814 Mean : 0.00000 Mean : 0.00000

3rd Qu.:139321 3rd Qu.: 1.31564 3rd Qu.: 0.80372

Max. :172792 Max. : 2.45493 Max. : 22.05773

V3 V4 V5

Min. :-48.3256 Min. :-5.68317 Min. :-113.74331

1st Qu.: -0.8904 1st Qu.:-0.84864 1st Qu.: -0.69160

Median : 0.1799 Median :-0.01985 Median : -0.05434

Mean : 0.0000 Mean : 0.00000 Mean : 0.00000

3rd Qu.: 1.0272 3rd Qu.: 0.74334 3rd Qu.: 0.61193

Max. : 9.3826 Max. :16.87534 Max. : 34.80167

V6 V7 V8

Min. :-26.1605 Min. :-43.5572 Min. :-73.21672

1st Qu.: -0.7683 1st Qu.: -0.5541 1st Qu.: -0.20863

Median : -0.2742 Median : 0.0401 Median : 0.02236

Mean : 0.0000 Mean : 0.0000 Mean : 0.00000

3rd Qu.: 0.3986 3rd Qu.: 0.5704 3rd Qu.: 0.32735

Max. : 73.3016 Max. :120.5895 Max. : 20.00721

V9 V10 V11

Min. :-13.43407 Min. :-24.58826 Min. :-4.79747

1st Qu.: -0.64310 1st Qu.: -0.53543 1st Qu.:-0.76249

Median : -0.05143 Median : -0.09292 Median :-0.03276

Mean : 0.00000 Mean : 0.00000 Mean : 0.00000

3rd Qu.: 0.59714 3rd Qu.: 0.45392 3rd Qu.: 0.73959

Max. : 15.59500 Max. : 23.74514 Max. :12.01891

V12 V13 V14

Min. :-18.6837 Min. :-5.79188 Min. :-19.2143

1st Qu.: -0.4056 1st Qu.:-0.64854 1st Qu.: -0.4256

Median : 0.1400 Median :-0.01357 Median : 0.0506

Mean : 0.0000 Mean : 0.00000 Mean : 0.0000

3rd Qu.: 0.6182 3rd Qu.: 0.66251 3rd Qu.: 0.4931

Max. : 7.8484 Max. : 7.12688 Max. : 10.5268

V15 V16 V17

Min. :-4.49894 Min. :-14.12985 Min. :-25.16280

1st Qu.:-0.58288 1st Qu.: -0.46804 1st Qu.: -0.48375

Median : 0.04807 Median : 0.06641 Median : -0.06568

Mean : 0.00000 Mean : 0.00000 Mean : 0.00000

3rd Qu.: 0.64882 3rd Qu.: 0.52330 3rd Qu.: 0.39968

Max. : 8.87774 Max. : 17.31511 Max. : 9.25353

V18 V19 V20

Min. :-9.498746 Min. :-7.213527 Min. :-54.49772

1st Qu.:-0.498850 1st Qu.:-0.456299 1st Qu.: -0.21172

Median :-0.003636 Median : 0.003735 Median : -0.06248

Mean : 0.000000 Mean : 0.000000 Mean : 0.00000

3rd Qu.: 0.500807 3rd Qu.: 0.458949 3rd Qu.: 0.13304

Max. : 5.041069 Max. : 5.591971 Max. : 39.42090

V21 V22 V23

Min. :-34.83038 Min. :-10.933144 Min. :-44.80774

1st Qu.: -0.22839 1st Qu.: -0.542350 1st Qu.: -0.16185

Median : -0.02945 Median : 0.006782 Median : -0.01119

Mean : 0.00000 Mean : 0.000000 Mean : 0.00000

3rd Qu.: 0.18638 3rd Qu.: 0.528554 3rd Qu.: 0.14764

Max. : 27.20284 Max. : 10.503090 Max. : 22.52841

V24 V25 V26

Min. :-2.83663 Min. :-10.29540 Min. :-2.60455

1st Qu.:-0.35459 1st Qu.: -0.31715 1st Qu.:-0.32698

Median : 0.04098 Median : 0.01659 Median :-0.05214

Mean : 0.00000 Mean : 0.00000 Mean : 0.00000

3rd Qu.: 0.43953 3rd Qu.: 0.35072 3rd Qu.: 0.24095

Max. : 4.58455 Max. : 7.51959 Max. : 3.51735

V27 V28 Amount

Min. :-22.565679 Min. :-15.43008 Min. : 0.00

1st Qu.: -0.070840 1st Qu.: -0.05296 1st Qu.: 5.60

Median : 0.001342 Median : 0.01124 Median : 22.00

Mean : 0.000000 Mean : 0.00000 Mean : 88.35

3rd Qu.: 0.091045 3rd Qu.: 0.07828 3rd Qu.: 77.17

Max. : 31.612198 Max. : 33.84781 Max. :25691.16

The data will need to be normalised as the ‘Amount’ column is not in th same range as the rest columns.

1. Distribution of Factor Variable Class

0: 284315

1: 492

Converting to percentages we see

0 1

0.998272514 0.001727486

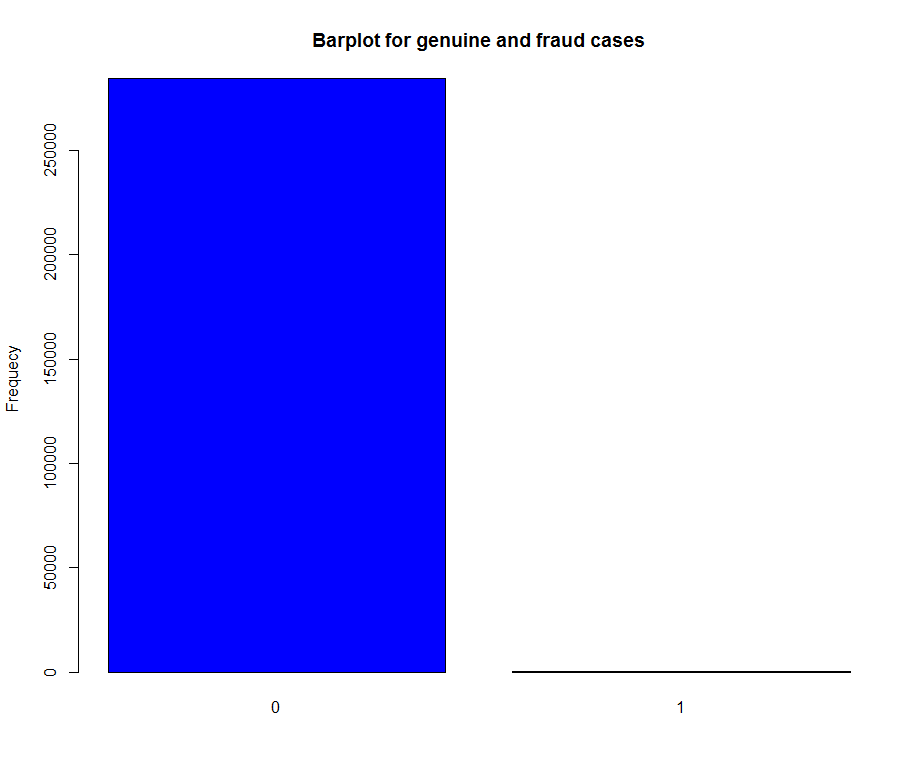


Figure 1: Genuine Vs Fraud cases Distribution

Only 0.17% transactions are fraud, hence the data is highly skewed in favour of genuine transcations. The data is unbalanced and needs to be handled differently.

1. Checking for missing values:

Time V1 V2 V3 V4 V5 V6 V7 V8 V9

0 0 0 0 0 0 0 0 0 0

V10 V11 V12 V13 V14 V15 V16 V17 V18 V19

0 0 0 0 0 0 0 0 0 0

V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount

0 0 0 0 0 0 0 0 0 0

Class

0

There are no missing values in the training dataset

1. Exploring the ‘Time’ variable further, we see there are 124592 unique time instances in the dataset. Studying there distribution, we find there are 2 peaks.

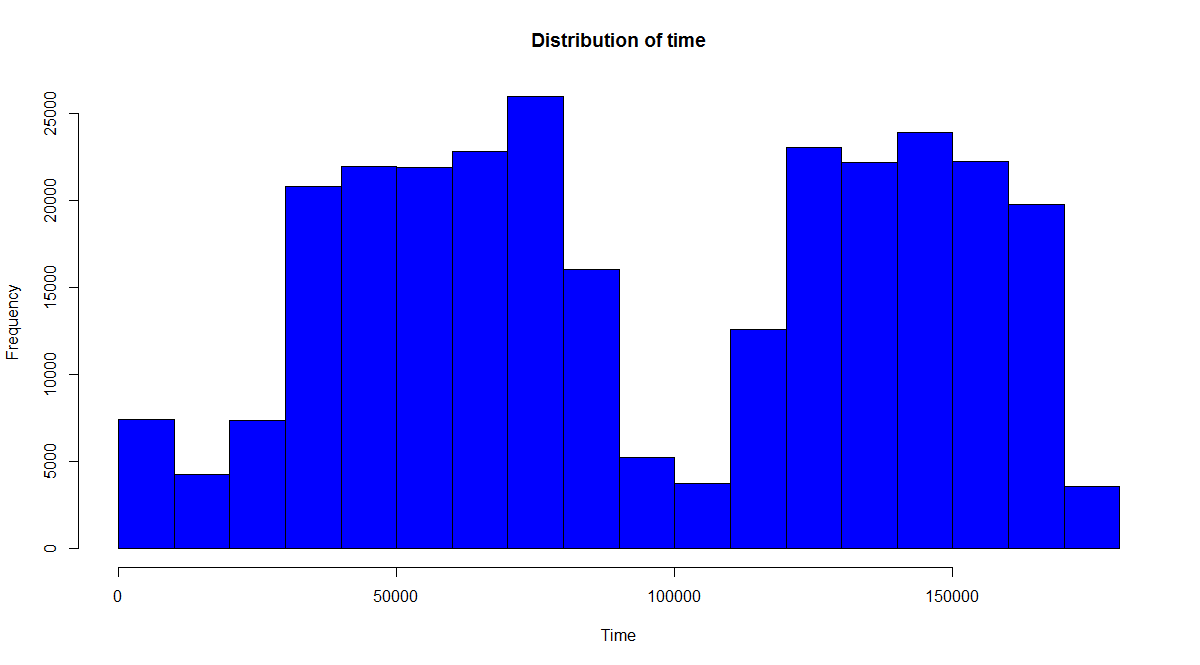


Figure 2: Distribution of ‘Time’ variable

Exploring the time with respect to whether the transaction was fraud or genuine.

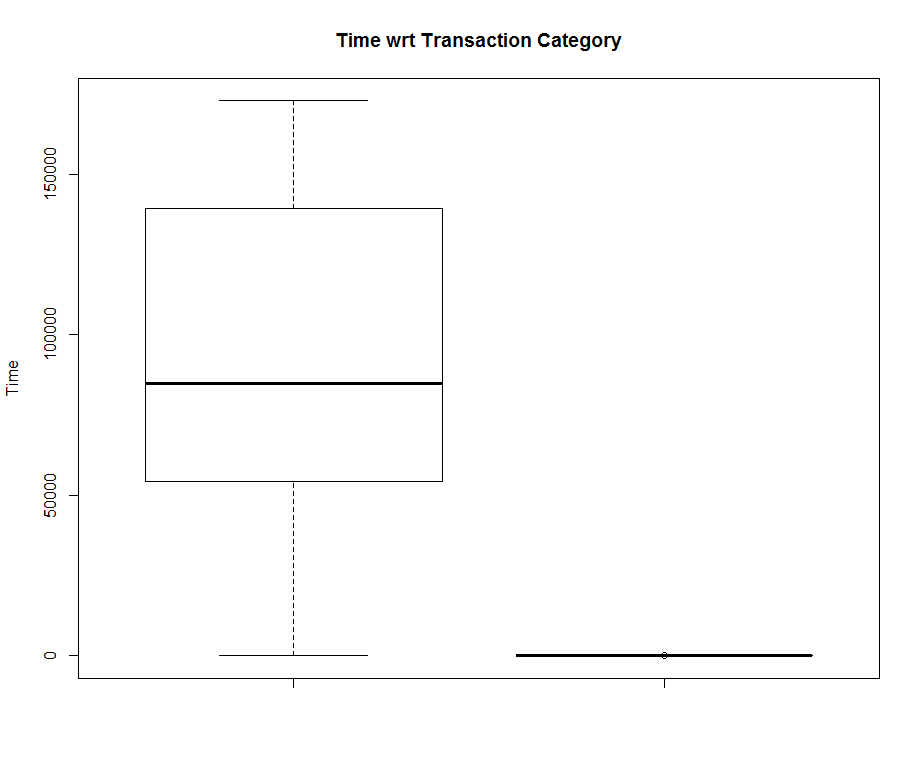


Figure 3: Distribution of ‘Time’ variable wrt ‘Class’

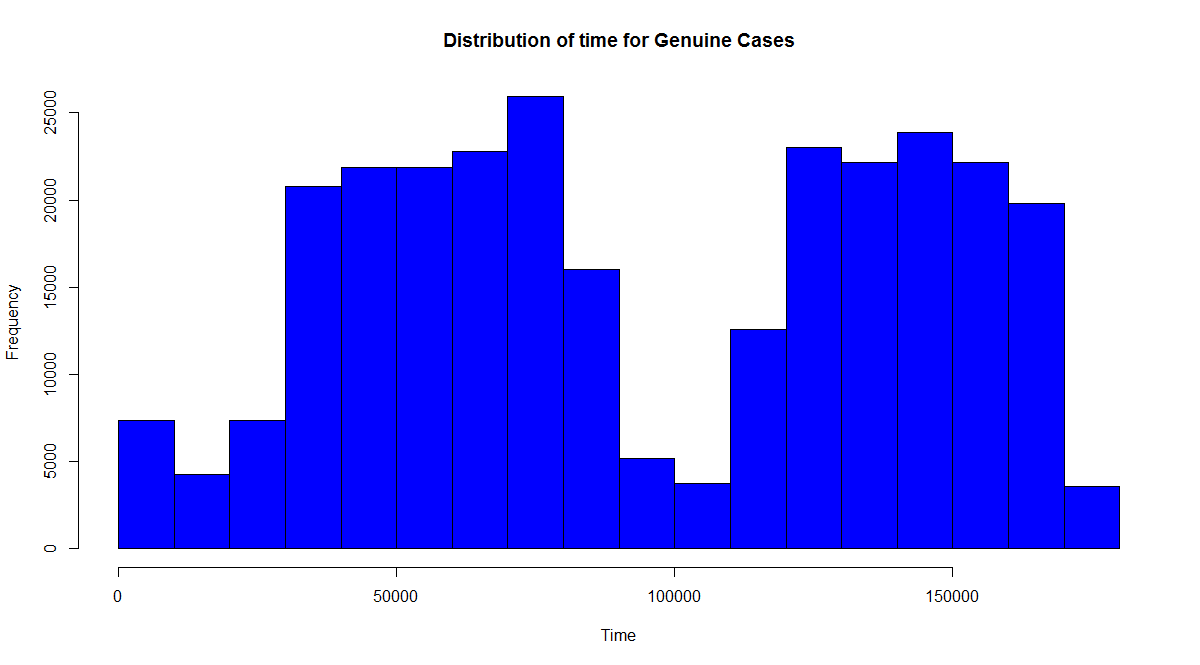


Figure 4 : Distribution of ‘Time’ variable for Genuine Cases

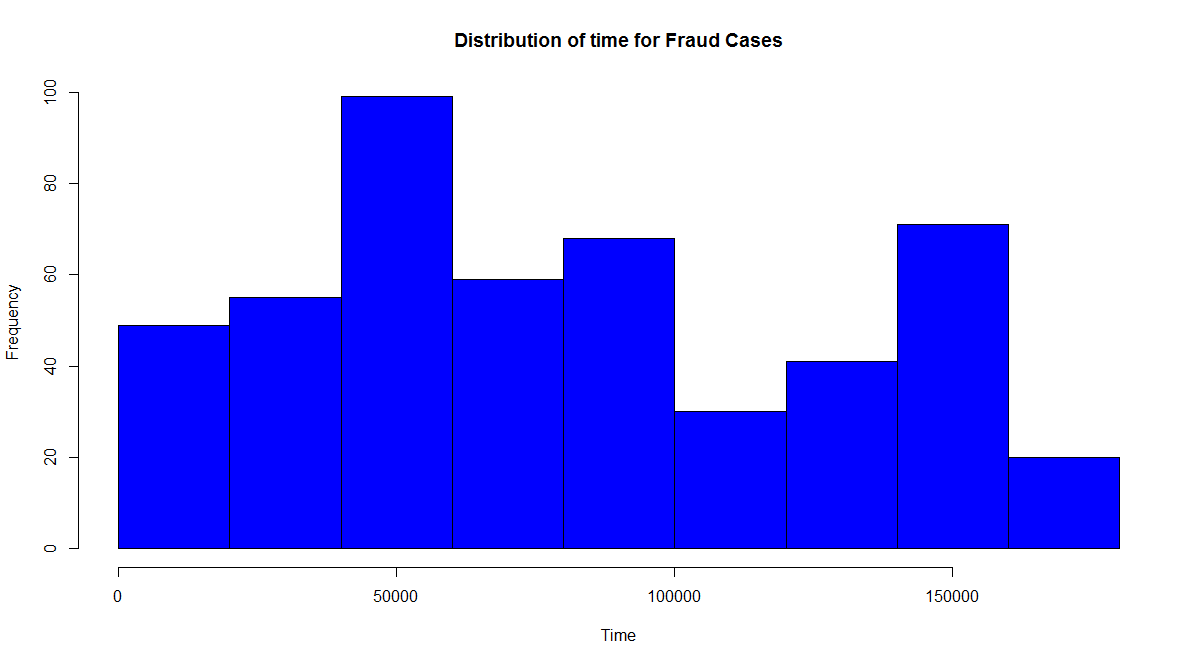


Figure 5 : Distribution of ‘Time’ variable for Fraud Cases

Time can be broken down to hours of day this can be studied further and I feel that can be added as a predictor in the dataset, but I have omitted the time variable as I mainly intend to study techniques for handling unbalanced data.

1. Distribution of Amount

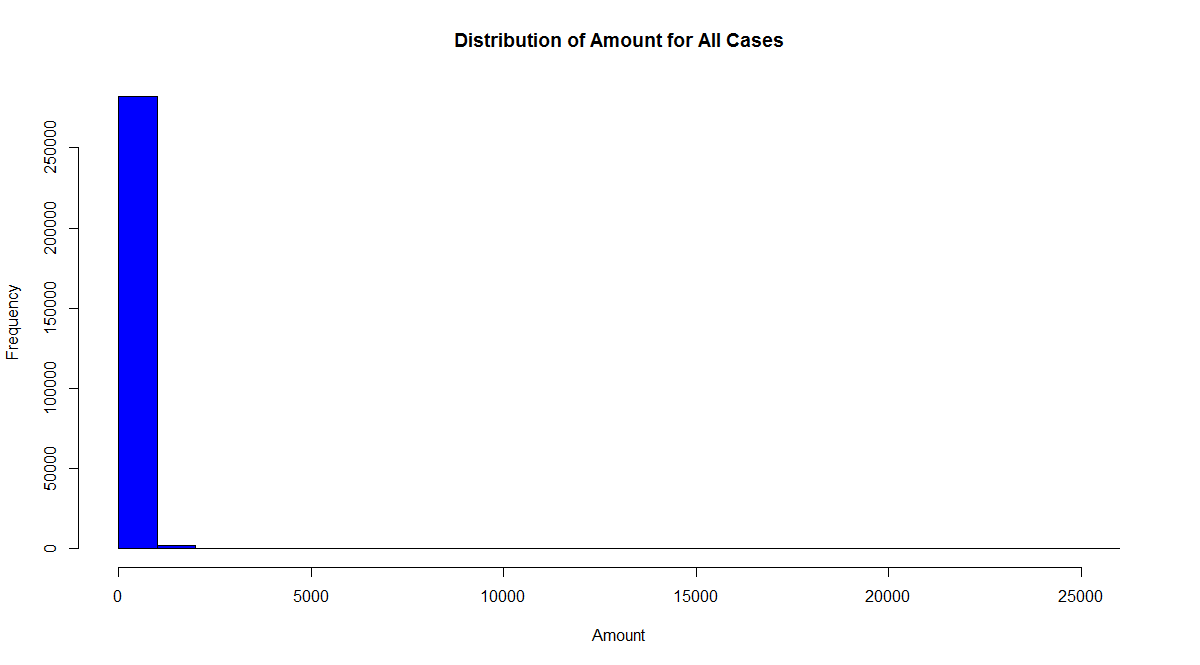


Figure 6 : Distribution of ‘Amount’ variable

There are outliers in the data and most of the transaction is very less, below 1000.

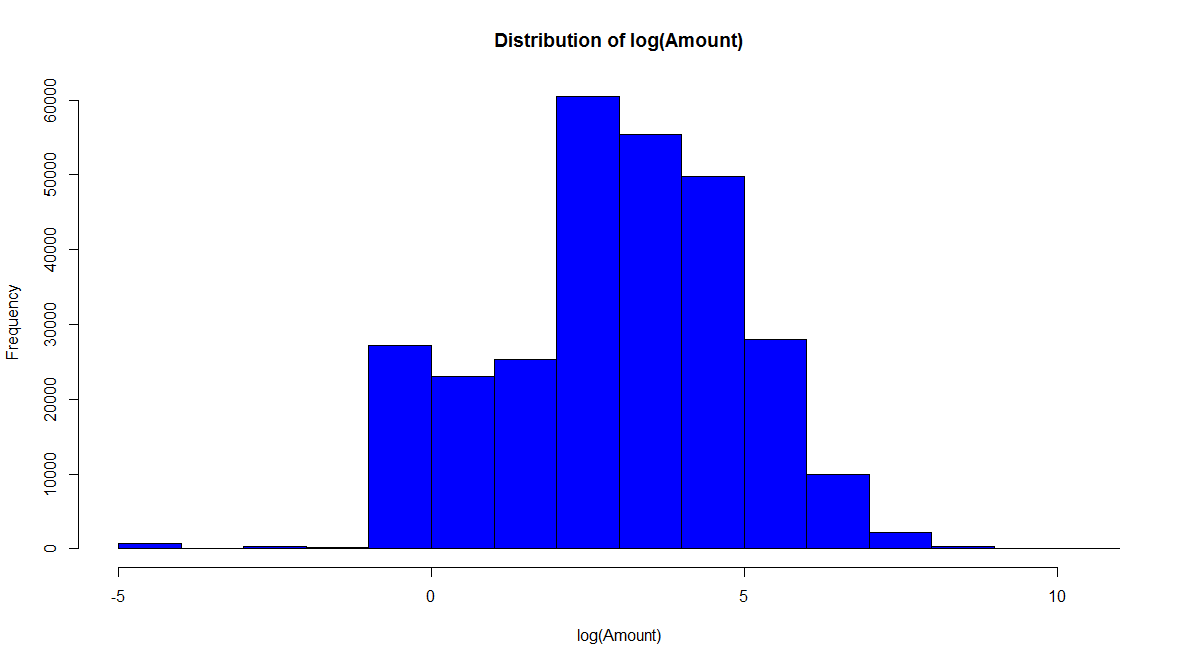


Figure 7 : Distribution of ‘log (Amount)’ variable

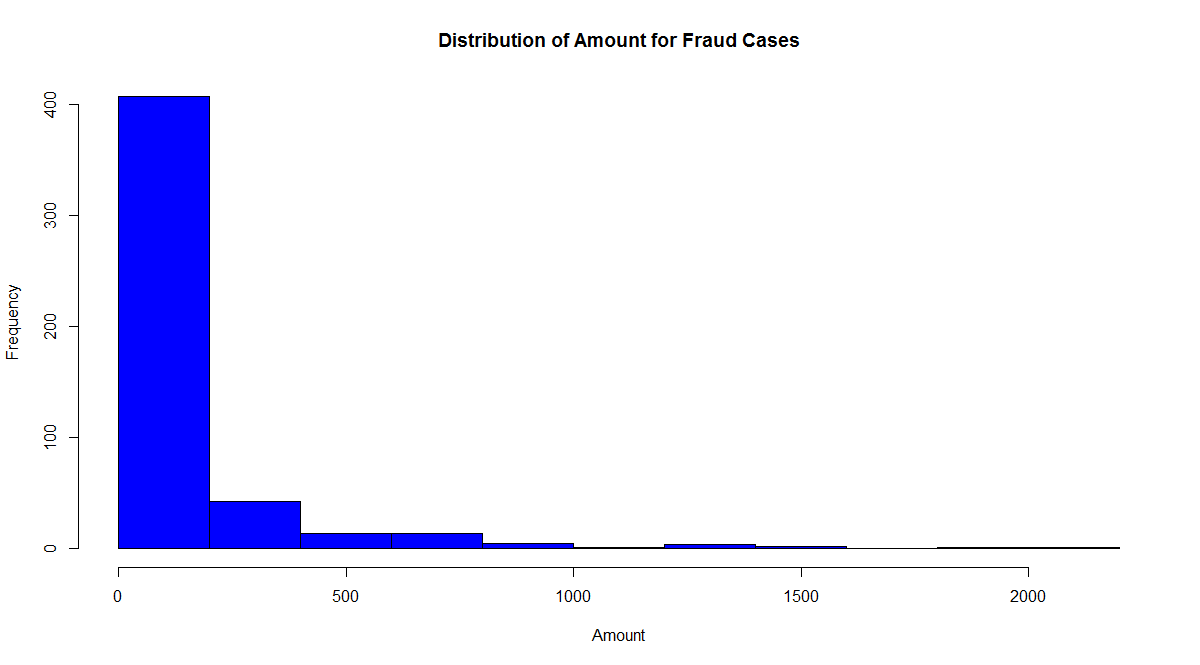


Figure 8 : Distribution of ‘Amount’ variable for Fraud Cases

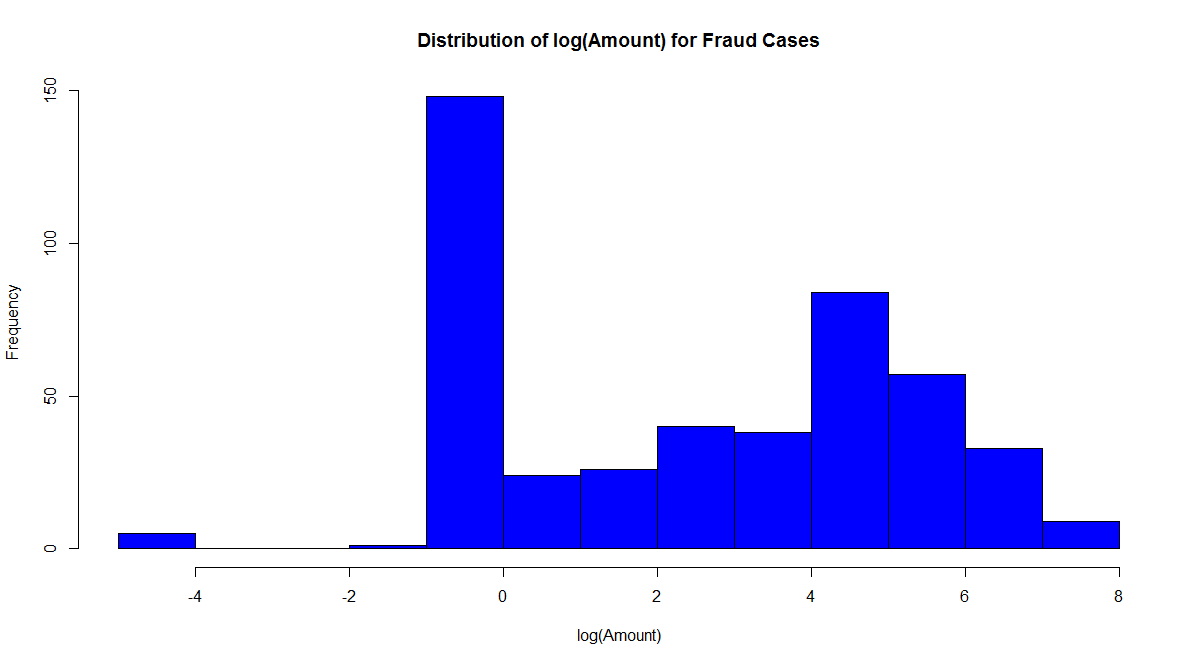


Figure 9 : Distribution of ‘log (Amount)’ variable for Genuine Cases

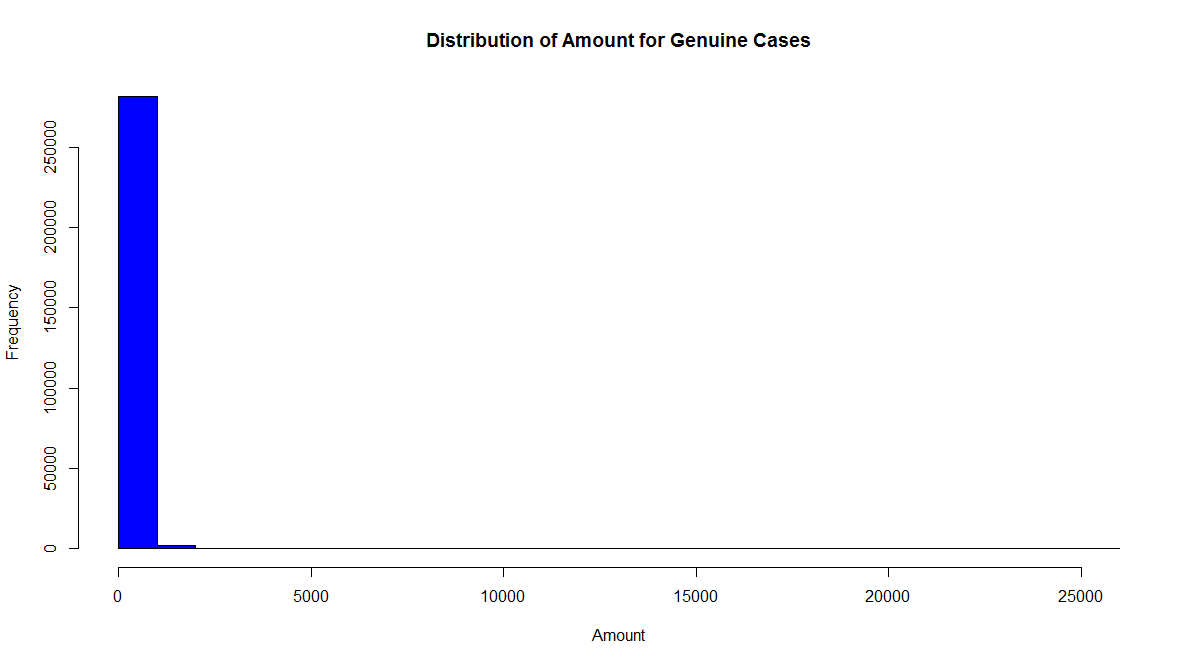


Figure 10 : Distribution of ‘Amount’ variable for Genuine Cases

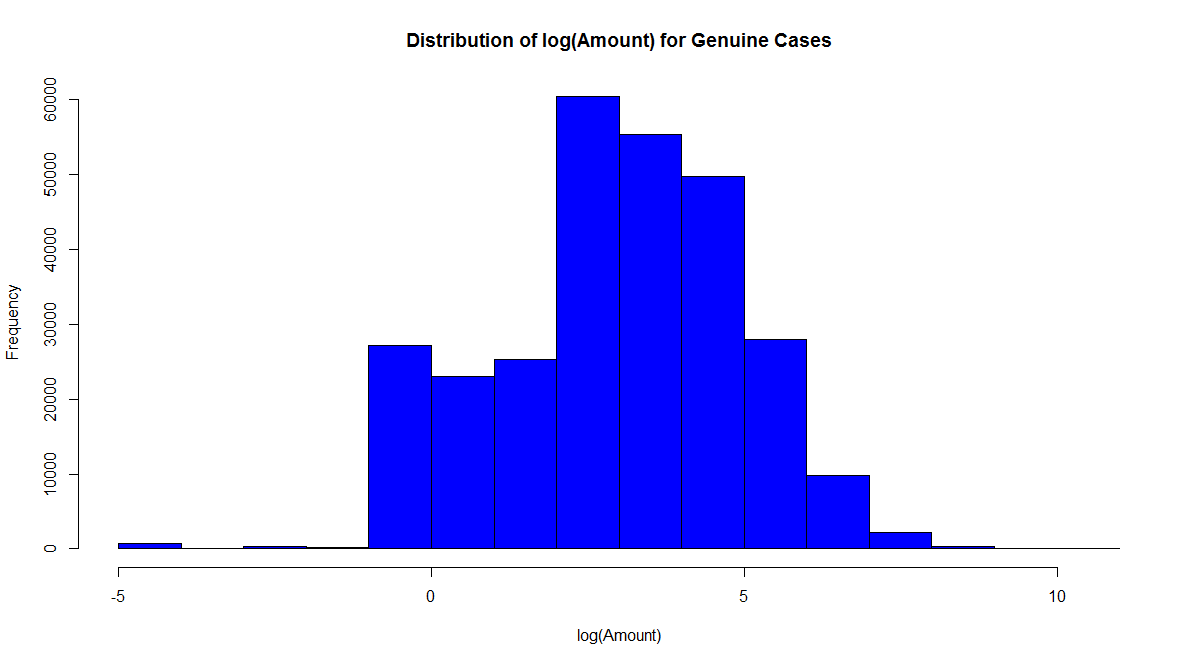


Figure 11 : Distribution of ‘log (Amount) ’ variable for Genuine Cases

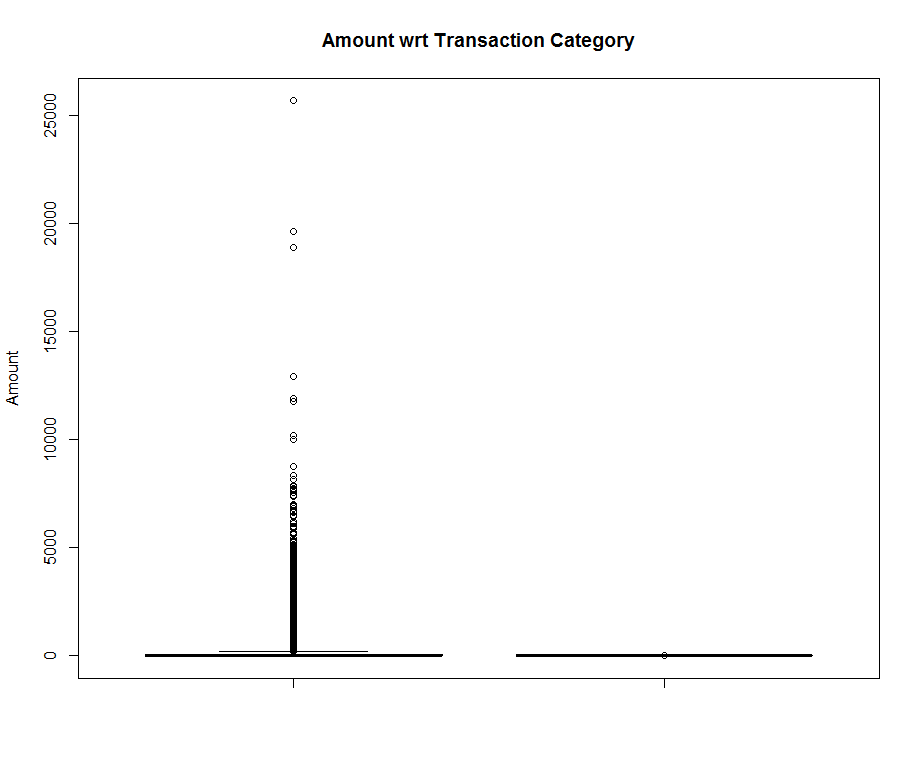


Figure 12: Boxplot of Amount wrt Classes

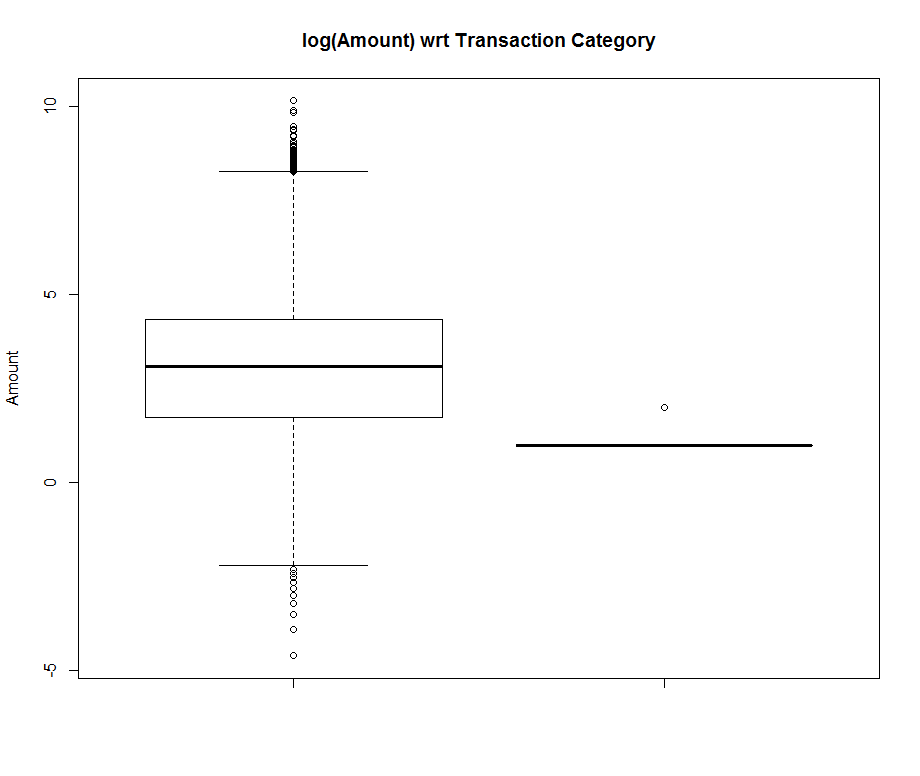


Figure 13 : Boxplot of ‘log (Amount)’ wrt to Classes

1. Correlation between numeric variables

Time V1 V2 V3 V4

Time 0 0.1173963 -1.059333e-02 -4.196182e-01 -1.052602e-01

V1 0 0.0000000 3.778136e-12 -2.120257e-12 -1.730054e-13

V2 0 0.0000000 0.000000e+00 2.325761e-12 -2.314686e-12

V3 0 0.0000000 0.000000e+00 0.000000e+00 2.036707e-13

V4 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V5 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V6 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V7 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V8 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V9 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V10 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V11 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V12 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V13 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V14 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V15 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V16 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V17 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V18 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V19 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V20 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V21 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V22 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V23 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V24 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V25 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V26 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V27 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V28 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

Amount 0 0.0000000 0.000000e+00 0.000000e+00 0.000000e+00

V5 V6 V7 V8

Time 1.730721e-01 -6.301647e-02 8.471437e-02 -3.694943e-02

V1 -3.473088e-12 -1.305631e-13 -1.119259e-13 2.114763e-12

V2 -1.832434e-12 9.438398e-13 5.403357e-12 2.130883e-14

V3 -4.031873e-12 -1.579536e-13 3.403903e-12 -1.272100e-12

V4 -2.551362e-13 1.083577e-12 8.132495e-13 7.334960e-13

V5 0.000000e+00 -6.976693e-14 1.573973e-11 -2.038679e-12

V6 0.000000e+00 0.000000e+00 -2.798679e-12 -5.442407e-13

V7 0.000000e+00 0.000000e+00 0.000000e+00 5.528828e-12

V8 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V9 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V10 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V11 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V12 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V13 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V14 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V15 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V16 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V17 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V18 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V19 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V20 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V21 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V22 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V23 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V24 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V25 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V26 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V27 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V28 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

Amount 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V9 V10 V11 V12

Time -8.660434e-03 3.061663e-02 -2.476894e-01 1.243481e-01

V1 3.065844e-14 -2.615246e-12 1.866405e-12 -1.238573e-12

V2 3.239585e-13 1.463464e-12 -8.309685e-13 6.138242e-13

V3 -6.807727e-13 -1.610430e-12 8.717905e-13 -2.728519e-12

V4 -7.143013e-13 -1.938999e-12 1.874617e-12 5.393679e-13

V5 -1.000998e-12 -7.194046e-13 -5.926563e-13 1.812423e-12

V6 2.036790e-12 7.430242e-13 1.014877e-12 -9.265766e-13

V7 5.094250e-13 1.675369e-12 -8.529871e-13 -2.837256e-13

V8 -2.243110e-12 -1.661103e-12 1.296801e-12 -3.855504e-13

V9 0.000000e+00 1.185490e-12 -3.972520e-13 -1.904274e-12

V10 0.000000e+00 0.000000e+00 3.430329e-13 5.111505e-13

V11 0.000000e+00 0.000000e+00 0.000000e+00 1.331729e-12

V12 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V13 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V14 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V15 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V16 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V17 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V18 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V19 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V20 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V21 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V22 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V23 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V24 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V25 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V26 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V27 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V28 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

Amount 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V13 V14 V15 V16

Time -6.590202e-02 -9.875682e-02 -1.834533e-01 1.190287e-02

V1 7.588569e-13 -1.877223e-13 -3.599550e-13 -1.143074e-12

V2 -1.181074e-12 -3.386305e-13 2.195254e-13 -7.999133e-13

V3 -1.021702e-12 -5.580998e-13 6.434778e-13 -8.746083e-13

V4 6.814056e-13 -1.403813e-12 1.526496e-12 3.077037e-13

V5 -6.994600e-14 -1.117481e-13 -1.593520e-12 -1.564220e-14

V6 1.484749e-12 -1.212805e-12 -1.054804e-12 1.374362e-12

V7 -8.171356e-13 2.038166e-12 1.074658e-12 -1.477819e-12

V8 7.721577e-13 -2.596307e-12 1.649074e-12 -1.830276e-12

V9 8.754413e-13 -1.271310e-12 8.634986e-13 1.239536e-12

V10 -1.507737e-12 3.397763e-13 1.392171e-12 2.619889e-13

V11 -2.925017e-12 -2.253790e-12 -8.980160e-14 8.364901e-13

V12 1.121360e-12 -4.837575e-13 3.208513e-13 -2.602930e-12

V13 0.000000e+00 -6.034182e-13 6.137809e-13 8.995503e-13

V14 0.000000e+00 0.000000e+00 2.013186e-13 8.617494e-13

V15 0.000000e+00 0.000000e+00 0.000000e+00 1.766565e-13

V16 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V17 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V18 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V19 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V20 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V21 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V22 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V23 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V24 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V25 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V26 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V27 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V28 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

Amount 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V17 V18 V19 V20

Time -7.329721e-02 9.043813e-02 2.897530e-02 -5.086602e-02

V1 1.671034e-12 -5.738981e-13 -2.770259e-12 2.659800e-13

V2 2.028193e-12 -1.911846e-14 -2.237620e-13 5.841829e-13

V3 -1.060045e-12 -8.848397e-13 -1.061063e-12 1.872899e-12

V4 1.152258e-14 -1.309557e-12 -9.754720e-13 -2.347247e-12

V5 1.714854e-13 1.101746e-12 5.958173e-13 -1.726553e-13

V6 7.430585e-13 6.863019e-13 1.148786e-12 -2.382079e-12

V7 -1.232384e-12 -4.281534e-13 -3.742299e-12 8.068884e-12

V8 7.023776e-13 1.499751e-12 1.988196e-12 -1.885463e-13

V9 -1.449475e-12 7.182584e-13 -8.787636e-13 1.270360e-12

V10 -1.492679e-13 -2.144505e-12 -5.616723e-13 3.339162e-12

V11 8.489461e-13 -1.860178e-12 -6.192420e-13 1.070962e-12

V12 -1.246917e-12 -2.375783e-12 -1.635058e-12 2.857089e-12

V13 1.764975e-13 -2.650139e-13 2.157014e-13 1.075049e-12

V14 -1.352673e-14 -1.983548e-13 1.098958e-12 -4.335007e-13

V15 -3.027588e-14 9.679611e-13 -1.067964e-12 1.466354e-12

V16 -1.142475e-12 -1.637220e-13 3.472458e-13 1.187096e-12

V17 0.000000e+00 -1.138642e-13 9.991246e-13 -2.221621e-12

V18 0.000000e+00 0.000000e+00 2.917325e-12 6.648566e-13

V19 0.000000e+00 0.000000e+00 0.000000e+00 2.661184e-12

V20 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V21 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V22 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V23 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V24 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V25 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V26 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V27 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V28 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

Amount 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V21 V22 V23 V24

Time 4.473573e-02 1.440591e-01 5.114236e-02 -1.618187e-02

V1 -3.276289e-12 2.281880e-12 -2.971105e-12 -1.029808e-12

V2 2.280472e-12 -2.545385e-13 -4.856090e-12 6.431295e-13

V3 6.734713e-13 -8.920902e-13 4.146940e-12 3.407909e-12

V4 -2.696261e-12 4.346169e-13 -4.161082e-12 -2.368708e-12

V5 -1.752047e-12 7.094118e-13 3.616245e-12 -2.806516e-13

V6 1.476788e-12 -1.144793e-12 -1.527754e-12 1.552047e-12

V7 2.787946e-12 -8.133466e-13 -4.292763e-12 -2.553546e-12

V8 -4.022367e-12 -2.679667e-12 9.014502e-13 -1.074450e-12

V9 3.040344e-12 -7.469595e-13 -1.010961e-12 8.578070e-13

V10 -5.545037e-13 -1.315069e-13 1.173374e-12 6.405629e-13

V11 1.105630e-13 -5.673237e-14 1.724666e-12 -1.162630e-12

V12 8.107889e-13 -2.346385e-12 -6.875719e-13 -2.911073e-12

V13 -2.037287e-12 -5.491493e-13 3.508057e-12 1.224058e-13

V14 -4.556948e-13 2.571614e-12 8.288834e-13 -3.382055e-12

V15 5.922406e-13 -4.116283e-13 -9.848142e-13 -3.256287e-12

V16 -1.067773e-12 2.009230e-12 4.055937e-13 -4.060641e-13

V17 1.793779e-12 2.284493e-13 -9.949956e-13 -2.073142e-12

V18 -2.185736e-12 1.393360e-12 -2.160547e-12 4.303983e-12

V19 -3.315520e-13 7.048287e-14 -7.118479e-13 1.326278e-12

V20 -3.892573e-12 1.632991e-12 -1.019681e-11 1.267509e-12

V21 0.000000e+00 -3.415736e-12 1.067057e-12 2.350230e-12

V22 0.000000e+00 0.000000e+00 -9.439954e-13 -1.123566e-12

V23 0.000000e+00 0.000000e+00 0.000000e+00 2.354124e-12

V24 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V25 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V26 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V27 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V28 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

Amount 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

V25 V26 V27 V28

Time -2.330828e-01 -4.140710e-02 -5.134591e-03 -9.412688e-03

V1 1.144481e-12 1.835308e-12 7.625090e-12 -9.765161e-13

V2 -9.422500e-13 -4.127907e-13 -9.857527e-13 2.525309e-12

V3 5.716851e-13 -2.577010e-12 -5.041607e-12 5.189076e-12

V4 1.619786e-12 -3.044168e-13 -1.455882e-12 -2.832441e-12

V5 1.451593e-12 -1.897617e-13 -2.124320e-12 1.010196e-11

V6 -2.723970e-12 3.351090e-12 1.481032e-12 -6.070378e-13

V7 -7.409884e-13 -4.476432e-12 -1.328623e-11 2.957869e-13

V8 -3.268952e-12 1.043946e-12 -3.499905e-12 1.866516e-12

V9 -1.588939e-12 -7.725877e-13 2.429260e-12 -1.406274e-12

V10 2.795281e-12 -2.739481e-13 1.553045e-12 5.116602e-12

V11 -1.352499e-12 2.718122e-12 -3.950311e-12 -4.247916e-12

V12 1.103404e-12 2.813185e-13 5.952698e-13 -7.428078e-12

V13 -1.513904e-12 -2.008413e-12 4.975687e-12 -6.777926e-12

V14 8.301094e-13 -3.301576e-13 -2.447708e-12 -1.700049e-12

V15 -1.725416e-12 5.477245e-13 -4.690803e-12 -4.214954e-12

V16 7.635441e-13 -1.323306e-12 7.022702e-12 5.739081e-13

V17 4.513480e-12 2.940560e-12 -1.324533e-12 1.854021e-12

V18 5.433069e-13 -1.810667e-12 -4.949689e-12 4.113181e-12

V19 9.270827e-13 2.412074e-12 -2.201415e-12 3.450654e-12

V20 -1.593323e-12 1.468201e-13 -2.996751e-12 6.124798e-12

V21 -3.120470e-12 8.462666e-13 -8.531033e-13 4.257084e-12

V22 1.968000e-12 -1.014017e-12 -1.725931e-13 5.948453e-12

V23 -3.750811e-12 -1.002322e-12 9.199173e-12 3.819501e-12

V24 -3.918120e-12 1.604849e-12 1.554599e-12 1.380798e-11

V25 0.000000e+00 2.112455e-12 -6.219543e-13 -8.597497e-12

V26 0.000000e+00 0.000000e+00 2.374815e-12 -1.036862e-11

V27 0.000000e+00 0.000000e+00 0.000000e+00 -4.440549e-12

V28 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

Amount 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

Amount

Time -0.010596373

V1 -0.227708653

V2 -0.531408939

V3 -0.210880475

V4 0.098731666

V5 -0.386356256

V6 0.215981180

V7 0.397311278

V8 -0.103079096

V9 -0.044245602

V10 -0.101502141

V11 0.000103977

V12 -0.009541802

V13 0.005293409

V14 0.033751172

V15 -0.002985848

V16 -0.003909527

V17 0.007309042

V18 0.035650341

V19 -0.056150787

V20 0.339403405

V21 0.105998928

V22 -0.064800646

V23 -0.112632554

V24 0.005146217

V25 -0.047836863

V26 -0.003208037

V27 0.028825463

V28 0.010258216

Amount 0.000000000

My Observation:

1) For fraud transaction the amount is very low. Time dependency can be studied by breaking the time in seconds into 24hour format and then studying the time pattern

2) For, amount there are outliers.

3) The data is unbalanced in favour of genuine transactions.

4) There are no missing values in the dataset.

5) Not much of feature generation can be done on the det as the variables are hidden and the values are derived from PCA.

### Data Pre-processing

#### Splitting the dataset

As this is an unbalanced dataset, the first step is to determine methodologies to handle this. To do this:

1. I did resampling to balance the instances of genuine and fraud transactions. The main objective of balancing classes is to either increasing the frequency of the minority class or decreasing the frequency of the majority class.
2. I worked on two approaches of resampling: under sampling and smote sampling

##### Random Under Sampling

Random Under sampling aims to balance class distribution by randomly eliminating majority class examples.  This is done until the majority and minority class instances are balanced out.

* **Advantages**
  + It can help improve run time and storage problems by reducing the number of training data samples when the training data set is huge.
* **Disadvantages**
  + It can discard potentially useful information which could be important for building rule classifiers.
  + The sample chosen by random under sampling may be a biased sample. And it will not be an accurate representative of the population. Thereby, resulting in inaccurate results with the actual test data set.

Under sampling option is available in the ‘Caret’ package.

When I apply the same I get a dataset of 984 instances with both classes balanced.

0 1

492 492

After that, I split the sample into training and validation set in the ratio of 70/30, with 690 observations in training set and 294 observations in validation set.

##### Random Over-Sampling

Over-Sampling increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample.

* **Advantages**
  + Unlike under sampling this method leads to no information loss.
  + Outperforms under sampling
* **Disadvantages**
  + It increases the likelihood of overfitting since it replicates the minority class events.

##### Cluster-Based Over Sampling

In this case, the K-means clustering algorithm is independently applied to minority and majority class instances. This is to identify clusters in the dataset. Subsequently, each cluster is oversampled such that all clusters of the same class have an equal number of instances and all classes have the same size. 

* **Advantages**
  + This clustering technique helps overcome the challenge between class imbalance. Where the number of examples representing positive class differs from the number of examples representing a negative class.
  + Also, overcome challenges within class imbalance, where a class is composed of different sub clusters. And each sub cluster does not contain the same number of examples.
* **Disadvantages**
  + The main drawback of this algorithm, like most oversampling techniques is the possibility of over-fitting the training data.

##### Informed Over Sampling: Synthetic Minority Over-sampling Technique

This technique is followed to avoid overfitting which occurs when exact replicas of minority instances are added to the main dataset. A subset of data is taken from the minority class as an example and then new synthetic similar instances are created. These synthetic instances are then added to the original dataset. The new dataset is used as a sample to train the classification models.

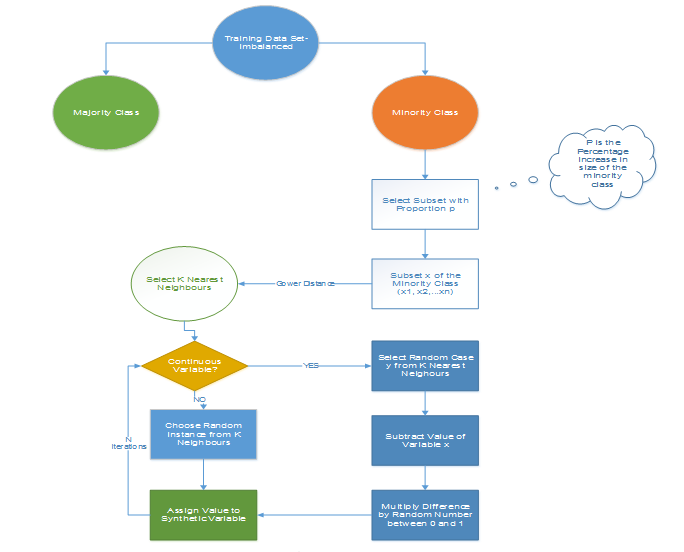
* **Advantages**
  + Mitigates the problem of overfitting caused by random oversampling as synthetic examples are generated rather than replication of instances
  + No loss of useful information
* **Disadvantages**
  + While generating synthetic examples SMOTE does not take into consideration neighbouring examples from other classes. This can result in increase in overlapping of classes and can introduce additional noise
  + SMOTE is not very effective for high dimensional data
* 

Figure 14 : **Synthetic Minority Oversampling Algorithm**

\*\*N is the number of attributes

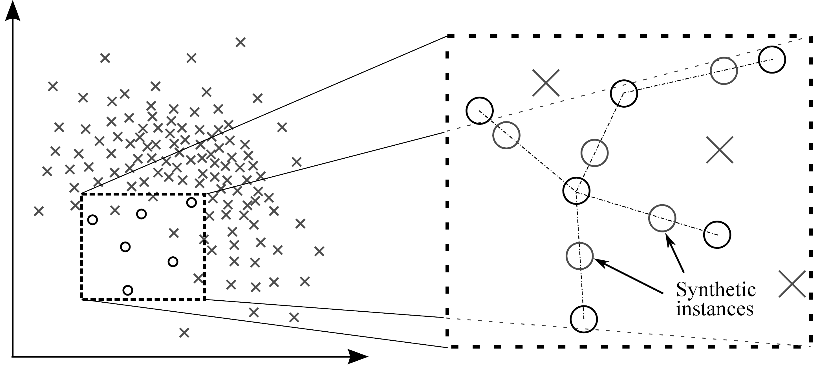
* 

Figure 15: **Generation of Synthetic Instances with the help of SMOTE**

SMOTE is available in ‘unbalanced’ and in ‘DMwR’ packages. On applying SMOTE, I get dataset with the below distribution of genuine and fraud cases.

0 1

1968 1476

Then I split the data into balanced train and validation set.The train set has 2412 observations with genuine case being 1378 and fraud cases being 1034.The validation set has 1032 observations with 442 fraud and 590 genuine cases.

##### Modified synthetic minority oversampling technique (MSMOTE)

It is a modified version of SMOTE. SMOTE does not consider the underlying distribution of the minority class and latent noises in the dataset. To improve the performance of SMOTE a modified method MSMOTE is used.

This algorithm classifies the samples of minority classes into 3 distinct groups – Security/Safe samples, Border samples, and latent nose samples. This is done by calculating the distances among samples of the minority class and samples of the training data.

Security samples are those data points which can improve the performance of a classifier. While on the other hand, noise are the data points which can reduce the performance of the classifier.  The ones which are difficult to categorize into any of the two are classified as border samples.

While the basic flow of MSOMTE is the same as that of SMOTE (discussed in the previous section).  In MSMOTE the strategy of selecting nearest neighbors is different from SMOTE**.** The algorithm randomly selects a data point from the k nearest neighbors for the security sample, selects the nearest neighbor from the border samples and does nothing for latent noise.

#### Algorithmic Ensemble Techniques

In this approach, balancing the dataset is not required. The above section, deals with handling imbalanced data by resampling original data to provide balanced classes. In this section, we are going to look at an alternate approach i.e.  Modifying existing classification algorithms to make them appropriate for imbalanced data sets.

The main objective of ensemble methodology is to improve the performance of single classifiers. The approach involves constructing several two stage classifiers from the original data and then aggregate their predictions.

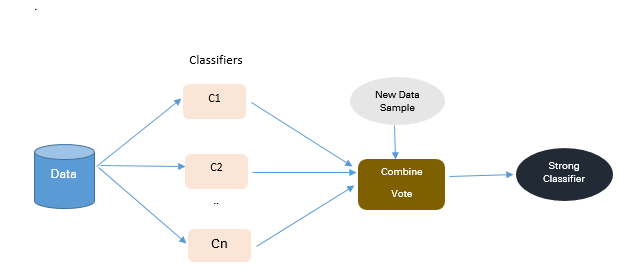


Figure 16 **: Approach to Ensemble based Methodologies**

##### Bagging Based

Bagging is an abbreviation of Bootstrap Aggregating. The conventional bagging algorithm involves generating ‘n’ different bootstrap training samples with replacement. And training the algorithm on each bootstrapped algorithm separately and then aggregating the predictions at the end.

Bagging is used for reducing Overfitting in order to create strong learners for generating accurate predictions. Unlike boosting, bagging allows replacement in the bootstrapped sample.

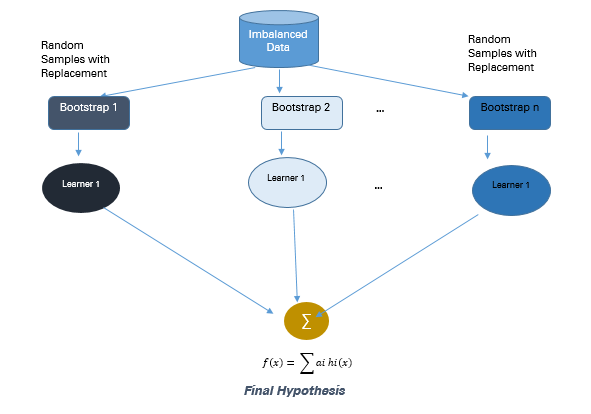


Figure 17 : **Approach to Bagging Methodology**

* **Advantages**
  + Improves stability & accuracy of machine learning algorithms
  + Reduces variance
  + Overcomes overfitting
  + Improved misclassification rate of the bagged classifier
  + In noisy data environments bagging outperforms boosting

* **Disadvantages**
  + Bagging works only if the base classifiers are not bad to begin with. Bagging bad classifiers can further degrade performance

##### Boosting-Based

Boosting is an ensemble technique to combine weak learners to create a strong learner that can make accurate predictions. Boosting starts out with a base classifier / weak classifier that is prepared on the training data.

What are base learners / weak classifiers?

The base learners / Classifiers are weak learners i.e. the prediction accuracy is only slightly better than average. A classifier learning algorithm is said to be weak when small changes in data induce big changes in the classification model.

In the next iteration, the new classifier focuses on or places more weight to those cases which were incorrectly classified in the last round.

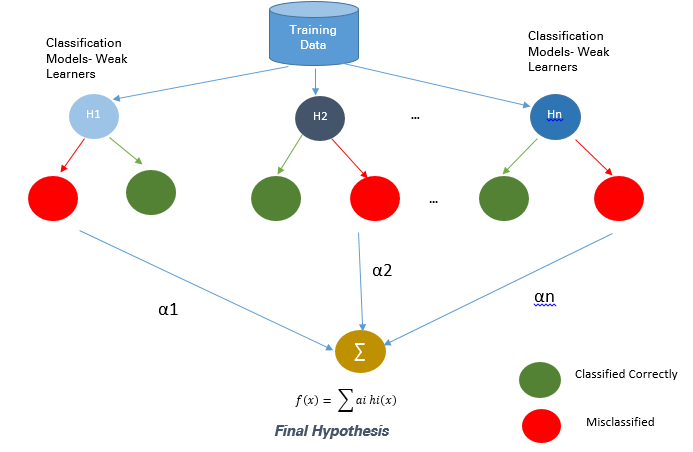


Figure 18: Boosting

##### Adaptive Boosting- Ada Boost

Ada Boost is the first original boosting technique which creates a highly accurate prediction rule by combining many weak and inaccurate rules.  Each classifier is serially trained with the goal of correctly classifying examples in every round that were incorrectly classified in the previous round.

For a learned classifier to make strong predictions it should follow the following three conditions:

* The rules should be simple
* Classifier should have been trained on sufficient number of training examples
* The Classifier should have low training error for the training instances

Each of the weak hypothesis has an accuracy slightly better than random guessing i.e. Error Term € (t) should be slightly more than ½-β where β >0. This is the fundamental assumption of this boosting algorithm which can produce a final hypothesis with a small error

After each round, it gives more focus to examples that are harder to classify.  The quantity of focus is measured by a weight, which initially is equal for all instances. After each iteration, the weights of misclassified instances are increased and the weights of correctly classified instances are decreased.

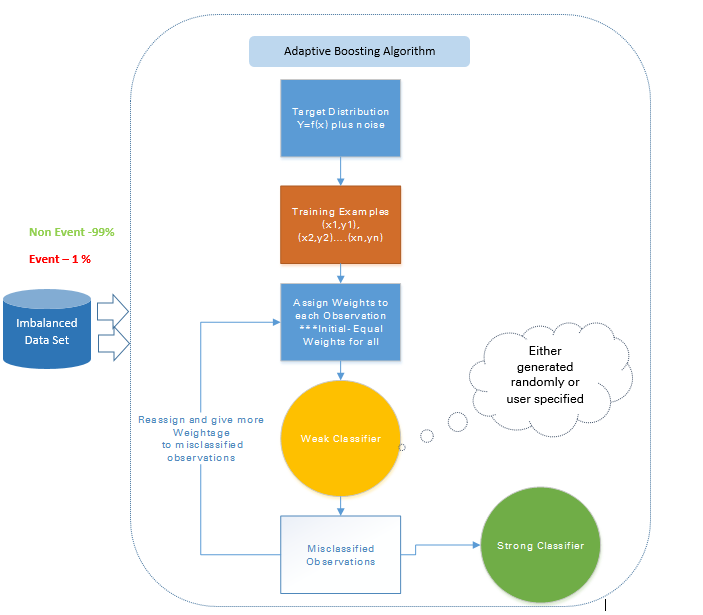
****

Figure 19****:** Adaptive Boosting**

* **Advantages**
  1. Very Simple to implement
  2. Good generalization- suited for any kind of classification problem ü Not prone to overfitting

* **Disadvantages**
  1. Sensitive to noisy data and outliers

##### Gradient Tree Boosting

In Gradient Boosting many models are trained sequentially. It is a numerical optimization algorithm where each model minimizes the loss function, **y = ax+b+e**, using the Gradient Descent Method.

Decision Trees are used as weak learners in Gradient Boosting.

While both Adaboost and Gradient Boosting work on weak learners / classifiers. And try to boost them into a strong learner, there are some fundamental differences in the two methodologies. Adaboost either requires the users to specify a set of weak learners  or randomly generates the weak learners before the actual learning process. The weight of each learner is adjusted at every step depending on whether it predicts a sample correctly.

On the other hand, Gradient Boosting builds the first learner on the training dataset to predict the samples, calculates the loss (Difference between real value and output of the first learner). And use this loss to build an improved learner in the second stage.

At every step, the residual of the loss function is calculated using the Gradient Descent Method and the new residual becomes a target variable for the subsequent iteration.

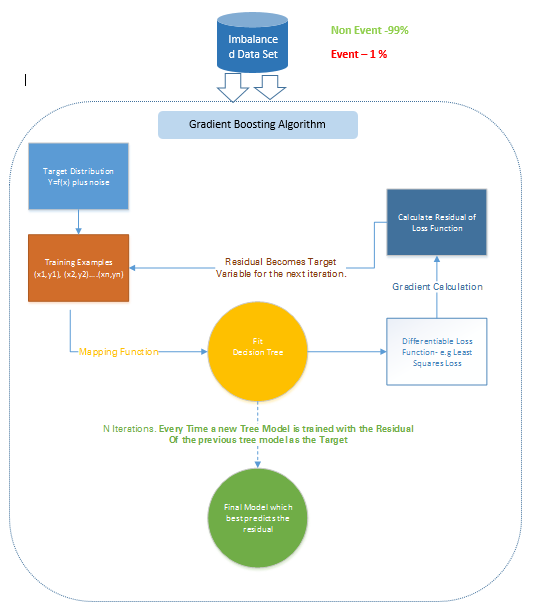


Figure 20 ; Gradient Boosting

**Disadvantages**

* Gradient Boosted trees are harder to fit than random forests
* Gradient Boosting Algorithms generally have 3 parameters which can be fine-tuned, Shrinkage parameter, depth of the tree, the number of trees. Proper training of each of these parameters is needed for a good fit. If parameters are not tuned correctly it may result in over-fitting.

**XG Boost**

XGBoost (Extreme Gradient Boosting) is an advanced and more efficient implementation of Gradient Boosting Algorithm discussed in the previous section.

Advantages over Other Boosting Techniques

* It is 10 times faster than the normal Gradient Boosting as it implements parallel processing. It is highly flexible as users can define custom optimization objectives and evaluation criteria, has an inbuilt mechanism to handle missing values.
* Unlike gradient boosting which stops splitting a node as soon as it encounters a negative loss, XG Boost splits up to the maximum depth specified and prunes the tree backward and removes splits beyond which there is an only negative loss.

#### Feature Selection/importance

As there are not too many featured involved, feature selection was not done. But I ranked the features according to their importance.

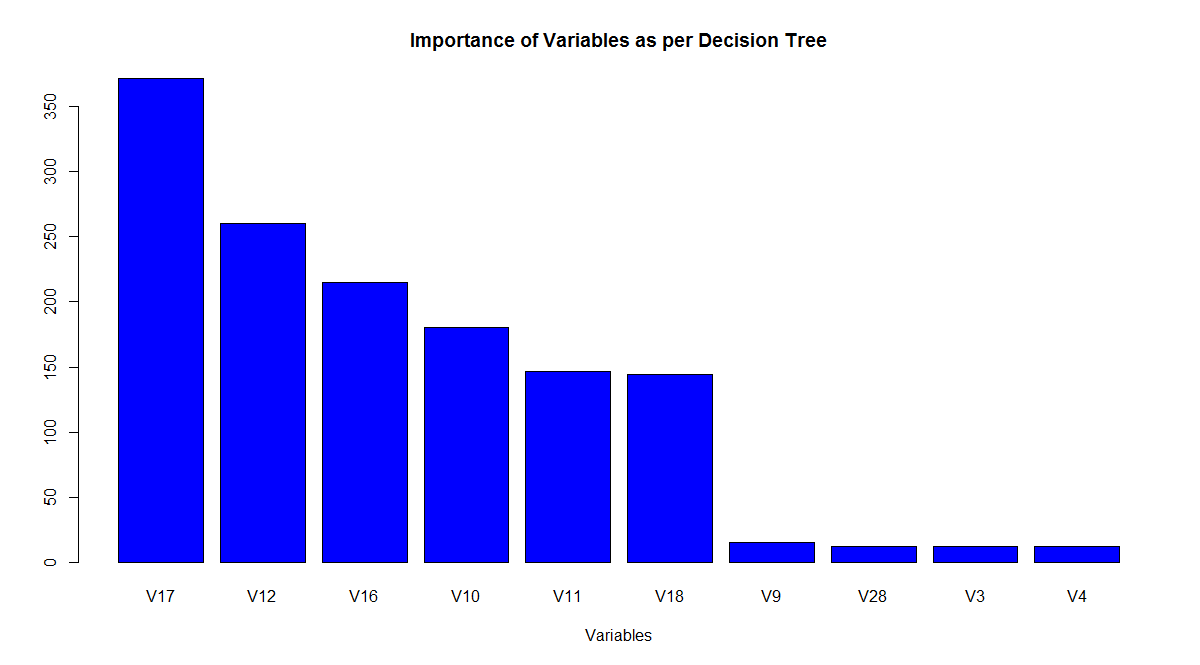


Figure 21 : Variable Importance as per decision tree

## Classification with Undersampling

Different algorithms were tried for classifying the documents. I trained & tested all of these on balanced train dataset and then retested on the balanced validation dataset and then on the unbalanced test sample and then compared.

Also, I tried different algorithmic models to combine weak learners developed on unbalanced dataset.

### Logistic Regression

Important Points

1. GLM does not assume a linear relationship between dependent and independent variables. However, it assumes a linear relationship between link function and independent variables in logit model.
2. The dependent variable need not to be normally distributed.
3. It does not uses OLS (Ordinary Least Square) for parameter estimation. Instead, it uses maximum likelihood estimation (MLE).
4. Errors need to be independent but not normally distributed.

#### Performance of Logistic Regression Model

To evaluate the performance of a logistic regression model, I considered below metrics

##### **AIC (Akaike Information Criteria)** –

The analogous metric of adjusted R² in logistic regression is AIC. AIC is the measure of fit which penalizes model for the number of model coefficients. Therefore, we always prefer model with minimum AIC value.

##### **Null Deviance and Residual Deviance** –

Null Deviance indicates the response predicted by a model with nothing but an intercept. Lower the value, better the model. Residual deviance indicates the response predicted by a model on adding independent variables. Lower the value, better the model.

##### **Confusion Matrix:**

It is nothing but a tabular representation of Actual vs Predicted values. This helps us to find the accuracy of the model and avoid overfitting. This is how it looks like:

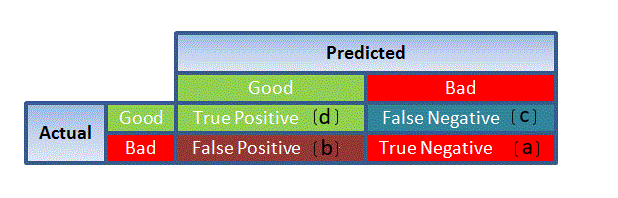
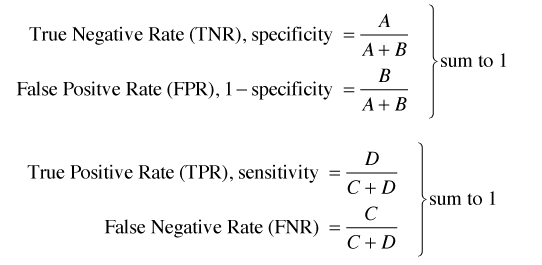
[](https://www.analyticsvidhya.com/wp-content/uploads/2015/11/1111.png)

Figure 22: Confusion Matrix

**accuracy** of the model is calculated as: (TP+TN)/(TP+TN+FP+FN)

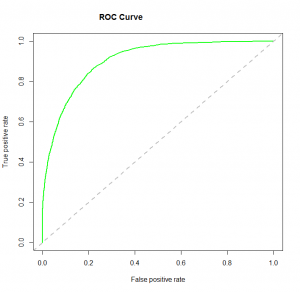
From confusion matrix, Specificity and Sensitivity can be derived as illustrated below:

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/11/9.png)

Specificity and Sensitivity plays a crucial role in deriving ROC curve.

##### **ROC Curve:**

Receiver Operating Characteristic (ROC) summarizes the model’s performance by evaluating the trade-offs between true positive rate (sensitivity) and false positive rate (1- specificity). For plotting ROC, it is advisable to assume p > 0.5 since we are more concerned about success rate. ROC summarizes the predictive power for all possible values of p > 0.5.  The area under curve (AUC), referred to as index of accuracy (A) or concordance index, is a perfect performance metric for ROC curve. Higher the area under curve, better the prediction power of the model. Below is a sample ROC curve. The ROC of a perfect predictive model has TP equals 1 and FP equals 0. This curve will touch the top left corner of the graph.

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/11/logit_roc.png)

**Note:** For model performance, we can also consider likelihood function. It is called so, because it selects the coefficient values which maximizes the likelihood of explaining the observed data. It indicates goodness of fit as its value approaches one, and a poor fit of the data as its value approaches zero.

Computing the Confusion Matrix for the Validation set:

Reference

Prediction 0 1

0 336 25

1 9 320

Accuracy : 0.9507

95% CI : (0.9318, 0.9656)

No Information Rate : 0.5

P-Value [Acc > NIR] : <2e-16

Kappa : 0.9014

Mcnemar's Test P-Value : 0.0101

Sensitivity : 0.9739

Specificity : 0.9275

Pos Pred Value : 0.9307

Neg Pred Value : 0.9726

Prevalence : 0.5000

Detection Rate : 0.4870

Detection Prevalence : 0.5232

Balanced Accuracy : 0.9507

'Positive' Class : 0

Testing the model on the full unbalanced data set

Reference

Prediction 0 1

0 274071 39

1 10244 453

Accuracy : 0.9639

95% CI : (0.9632, 0.9646)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.0779

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.96397

Specificity : 0.92073

Pos Pred Value : 0.99986

Neg Pred Value : 0.04235

Prevalence : 0.99827

Detection Rate : 0.96230

Detection Prevalence : 0.96244

Balanced Accuracy : 0.94235

'Positive' Class : 0

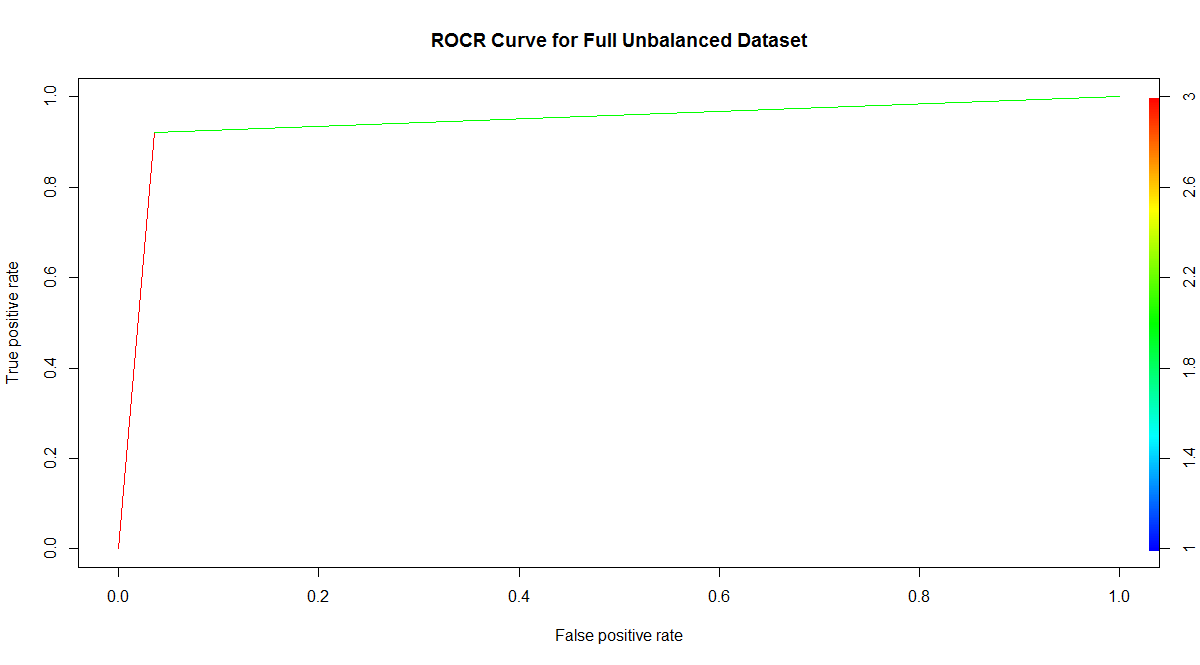


Figure 23 :ROC Curve for full data for Logistic regression with Under sampling

### Support Vector machines

A support vector machine constructs a hyperplane or set of hyperplanes in a high or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. If the data is not linearly separable, the finite-dimensional space is mapped into a much higher-dimensional space, presumably making the separation easier in that space(kernels)

For SVM, I developed models with both linear and radial kernels.

Following steps were executed on balanced training and validation dataset for developing a linear SVM model

* Created container of training dataset
* Model was trained on cost varying from 0.1 to 100
* Created container of testing dataset
* Test data set vocab was made same as the train vocab terms.
* Results were created by applying model on the test dataset
* Confusion matrix was generated and evaluation metrics were recorded

##### Linear SVM

Reference

Prediction 0 1

0 147 15

1 0 132

Accuracy : 0.949

95% CI : (0.9172, 0.9712)

No Information Rate : 0.5

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.898

Mcnemar's Test P-Value : 0.0003006

Sensitivity : 1.0000

Specificity : 0.8980

Pos Pred Value : 0.9074

Neg Pred Value : 1.0000

Prevalence : 0.5000

Detection Rate : 0.5000

Detection Prevalence : 0.5510

Balanced Accuracy : 0.9490

'Positive' Class : 0

Confusion Matrix for full unbalanced dataset

Reference

Prediction 0 1

0 278111 46

1 6204 446

Accuracy : 0.9781

95% CI : (0.9775, 0.9786)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.1221

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.97818

Specificity : 0.90650

Pos Pred Value : 0.99983

Neg Pred Value : 0.06707

Prevalence : 0.99827

Detection Rate : 0.97649

Detection Prevalence : 0.97665

Balanced Accuracy : 0.94234

'Positive' Class : 0

##### For Radial Function

Tuning the parameters sigma and cost was done with caret

C Accuracy Kappa

0.25 0.9144982 0.8289886

0.50 0.9214554 0.8429037

1.00 0.9226065 0.8451999

Tuning parameter 'sigma' was held constant at a value of 0.05208331

The final values used for the model were sigma = 0.05208331 and C = 1.

Confusion Matrix and Statistics on the balanced Validation set

Reference

Prediction 0 1

0 145 12

1 2 135

Accuracy : 0.9524

95% CI : (0.9214, 0.9737)

No Information Rate : 0.5

P-Value [Acc > NIR] : < 2e-16

Kappa : 0.9048

Mcnemar's Test P-Value : 0.01616

Sensitivity : 0.9864

Specificity : 0.9184

Pos Pred Value : 0.9236

Neg Pred Value : 0.9854

Prevalence : 0.5000

Detection Rate : 0.4932

Detection Prevalence : 0.5340

Balanced Accuracy : 0.9524

'Positive' Class : 0

Confusion Matrix and Statistics on full unbalanced data

Reference

Prediction 0 1

0 276837 41

1 7478 451

Accuracy : 0.9736

95% CI : (0.973, 0.9742)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.1042

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.97370

Specificity : 0.91667

Pos Pred Value : 0.99985

Neg Pred Value : 0.05688

Prevalence : 0.99827

Detection Rate : 0.97202

Detection Prevalence : 0.97216

Balanced Accuracy : 0.94518

'Positive' Class : 0

##### With Pkg ‘e1071’

Confusion Matrix and Statistics on Validation set

Reference

Prediction 0 1

0 142 11

1 5 136

Accuracy : 0.9456

95% CI : (0.9131, 0.9686)

No Information Rate : 0.5

P-Value [Acc > NIR] : <2e-16

Kappa : 0.8912

Mcnemar's Test P-Value : 0.2113

Sensitivity : 0.9660

Specificity : 0.9252

Pos Pred Value : 0.9281

Neg Pred Value : 0.9645

Prevalence : 0.5000

Detection Rate : 0.4830

Detection Prevalence : 0.5204

Balanced Accuracy : 0.9456

'Positive' Class : 0

Confusion Matrix and Statistics on full unbalanced dataset

Reference

Prediction 0 1

0 271811 32

1 12504 460

Accuracy : 0.956

95% CI : (0.9552, 0.9567)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.0653

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.95602

Specificity : 0.93496

Pos Pred Value : 0.99988

Neg Pred Value : 0.03548

Prevalence : 0.99827

Detection Rate : 0.95437

Detection Prevalence : 0.95448

Balanced Accuracy : 0.94549

'Positive' Class : 0

Overall Analysis of SVM:

1) Performance of SVM was at par with regression model.

2) Also we can see that the linear model from pkg ‘e1071’ performed much better than the radial model and linear model from svm pkg.

### Decision Trees

For Decision tree following steps were executed

* Train dataset was split into training and validation set
* Model was trained for different values of cp on training data set and results were tested on validation set.

cp Accuracy Kappa

0.01304348 0.9116050 0.8231878

0.01739130 0.9083869 0.8167713

0.82898551 0.7407818 0.4847296

The optimal value of cp used for the model was 0.01304348.

* Accuracy was used to select the optimal model using the largest value.
* Variable importance

V14 V10 V11 V17 V12 V3 V4 V8

20 16 16 15 15 14 1 1

Evaluating the result on the Validation set:

Reference

Prediction 0 1

0 135 11

1 12 136

Accuracy : 0.9218

95% CI : (0.8849, 0.9498)

No Information Rate : 0.5

P-Value [Acc > NIR] : <2e-16

Kappa : 0.8435

Mcnemar's Test P-Value : 1

Sensitivity : 0.9184

Specificity : 0.9252

Pos Pred Value : 0.9247

Neg Pred Value : 0.9189

Prevalence : 0.5000

Detection Rate : 0.4592

Detection Prevalence : 0.4966

Balanced Accuracy : 0.9218

'Positive' Class : 0

Testing the result on the whole unbalanced dataset:

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 256592 29

1 27723 463

Accuracy : 0.9026

95% CI : (0.9015, 0.9036)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.029

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.90249

Specificity : 0.94106

Pos Pred Value : 0.99989

Neg Pred Value : 0.01643

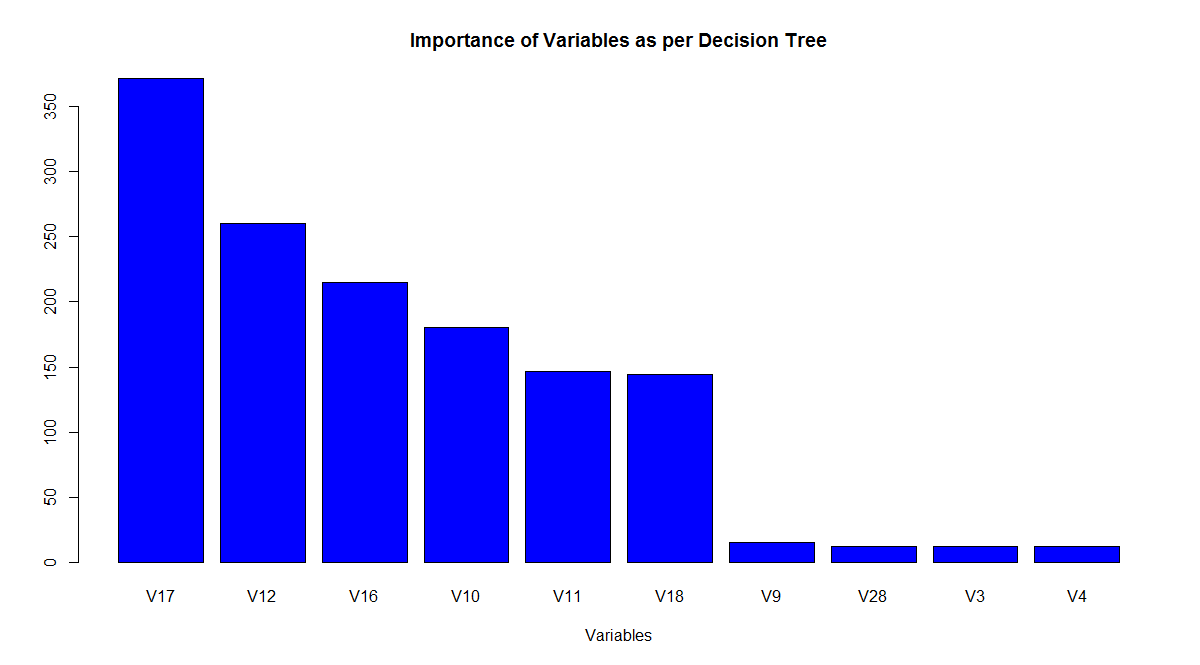
Prevalence : 0.99827

Detection Rate : 0.90093

Detection Prevalence : 0.90103

Balanced Accuracy : 0.92177

'Positive' Class : 0



Overall Analysis of Decision tree

Decision-tree learning algorithms are based on heuristics such as the greedy algorithms where locally-optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally-optimal decision tree.

But for under sampling, decision trees gave the best result after the bagged tree. Its specificity was 0.94 which was quite impressive taking into consideration that tree models are also quite interpretable.

Decision trees are quite simple to understand and interpret as it give us business rules wherein we can classify the test data fast once the model has been trained,

### Random Forest

**Random forests** or random decision forests are an ensemble learning method for classification that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) Random decision forests correct for decision trees' habit of overfitting to their training set. I planned to try this also, but due to time constraint, I had to leave this out for moment and will try this at some later stage.

### KNN Model

I planned to try this also, but again due to time constraint, I had to leave this out but will surely try this at some later stage on this data.

### Neural Network Model

I trained the neural network with the ‘caret’ package to get the hidden number of layer and decay parameters

Resampling results across tuning parameters:

size decay Accuracy Kappa

1 0e+00 0.9244152 0.8488055

1 1e-04 0.9235418 0.8470805

1 1e-01 0.9302554 0.8605160

3 0e+00 0.9090723 0.8181203

3 1e-04 0.9111315 0.8222770

3 1e-01 0.9203782 0.8407715

5 0e+00 0.9105048 0.8209759

5 1e-04 0.9157603 0.8315132

5 1e-01 0.9261630 0.8523194

Accuracy was used to select the optimal model using the largest value.

The final values used for the model were size = 1 and decay = 0.1.

Generating the Confusion Matrix and Statistics for the validation set

Reference

Prediction 0 1

0 143 15

1 4 132

Accuracy : 0.9354

95% CI : (0.9009, 0.9606)

No Information Rate : 0.5

P-Value [Acc > NIR] : < 2e-16

Kappa : 0.8707

Mcnemar's Test P-Value : 0.02178

Sensitivity : 0.9728

Specificity : 0.8980

Pos Pred Value : 0.9051

Neg Pred Value : 0.9706

Prevalence : 0.5000

Detection Rate : 0.4864

Detection Prevalence : 0.5374

Balanced Accuracy : 0.9354

'Positive' Class : 0

Confusion Matrix and Statistics for the full unbalanced dataset

Reference

Prediction 0 1

0 273340 36

1 10975 456

Accuracy : 0.9613

95% CI : (0.9606, 0.962)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.0734

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.96140

Specificity : 0.92683

Pos Pred Value : 0.99987

Neg Pred Value : 0.03989

Prevalence : 0.99827

Detection Rate : 0.95974

Detection Prevalence : 0.95986

Balanced Accuracy : 0.94411

'Positive' Class : 0

### Treebag Model

I also tried to classify the observation with the bagged tree model

Confusion Matrix and Statistics on the validation set

Reference

Prediction 0 1

0 140 14

1 7 133

Accuracy : 0.9286

95% CI : (0.8929, 0.9552)

No Information Rate : 0.5

P-Value [Acc > NIR] : <2e-16

Kappa : 0.8571

Mcnemar's Test P-Value : 0.1904

Sensitivity : 0.9524

Specificity : 0.9048

Pos Pred Value : 0.9091

Neg Pred Value : 0.9500

Prevalence : 0.5000

Detection Rate : 0.4762

Detection Prevalence : 0.5238

Balanced Accuracy : 0.9286

'Positive' Class : 0

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 270544 16

1 13771 476

Accuracy : 0.9516

95% CI : (0.9508, 0.9524)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.0615

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.95156

Specificity : 0.96748

Pos Pred Value : 0.99994

Neg Pred Value : 0.03341

Prevalence : 0.99827

Detection Rate : 0.94992

Detection Prevalence : 0.94998

Balanced Accuracy : 0.95952

'Positive' Class : 0

Bagged tree gave the best results even outperforming the ensembles.

## Classification with Smote Sampling

#### Logistic Regression

Confusion Matrix and Statistics for validation set

Reference

Prediction 0 1

0 578 38

1 12 404

Accuracy : 0.9516

95% CI : (0.9366, 0.9638)

No Information Rate : 0.5717

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9003

Mcnemar's Test P-Value : 0.000407

Sensitivity : 0.9797

Specificity : 0.9140

Pos Pred Value : 0.9383

Neg Pred Value : 0.9712

Prevalence : 0.5717

Detection Rate : 0.5601

Detection Prevalence : 0.5969

Balanced Accuracy : 0.9468

'Positive' Class : 0

Confusion Matrix and Statistics for full dataset

Reference

Prediction 0 1

0 278264 40

1 6051 452

Accuracy : 0.9786

95% CI : (0.9781, 0.9791)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.1264

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.97872

Specificity : 0.91870

Pos Pred Value : 0.99986

Neg Pred Value : 0.06951

Prevalence : 0.99827

Detection Rate : 0.97703

Detection Prevalence : 0.97717

Balanced Accuracy : 0.94871

'Positive' Class : 0

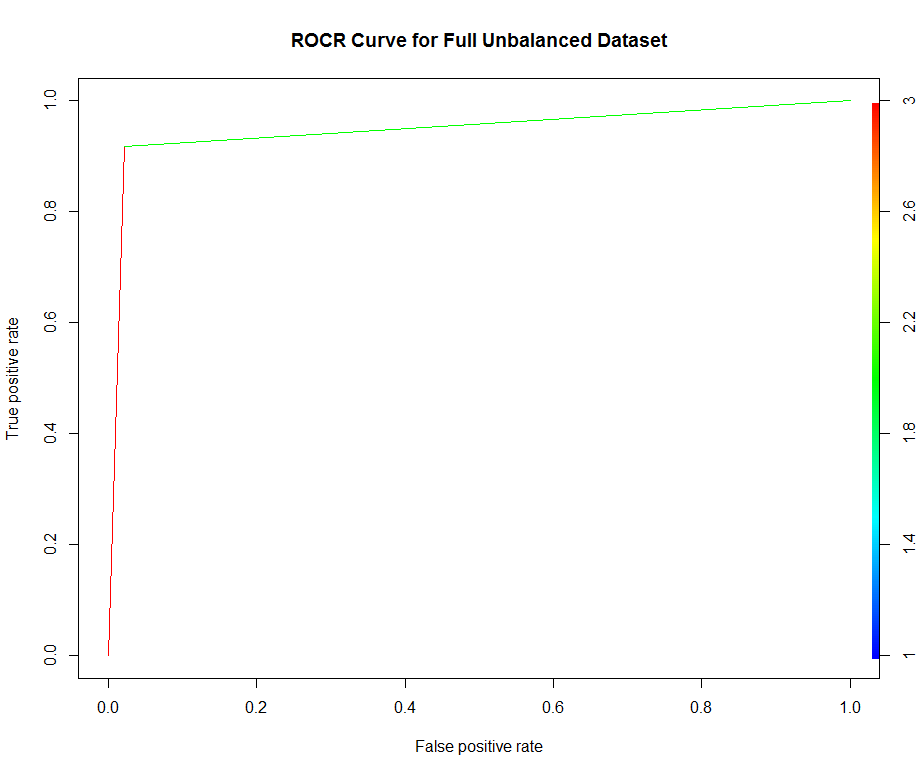


Figure 24 : ROC Curve for full data with logistic regression with smote sampling

#### Decision Tree

Tuning of ‘cp’ was done with the caret package.

Resampling results across tuning parameters ‘cp’

cp Accuracy Kappa

0.008704062 0.9383879 0.8731414

0.014990329 0.9319249 0.8592353

0.841392650 0.7685049 0.4706320

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was cp = 0.008704062.

Confusion Matrix and Statistics for validation set

Reference

Prediction 0 1

0 572 51

1 18 391

Accuracy : 0.9331

95% CI : (0.9161, 0.9476)

No Information Rate : 0.5717

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.8622

Mcnemar's Test P-Value : 0.000117

Sensitivity : 0.9695

Specificity : 0.8846

Pos Pred Value : 0.9181

Neg Pred Value : 0.9560

Prevalence : 0.5717

Detection Rate : 0.5543

Detection Prevalence : 0.6037

Balanced Accuracy : 0.9271

'Positive' Class : 0

Confusion Matrix and Statistics for full unbalanced dataset

Reference

Prediction 0 1

0 276231 55

1 8084 437

Accuracy : 0.9714

95% CI : (0.9708, 0.972)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.094

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.97157

Specificity : 0.88821

Pos Pred Value : 0.99980

Neg Pred Value : 0.05129

Prevalence : 0.99827

Detection Rate : 0.96989

Detection Prevalence : 0.97008

Balanced Accuracy : 0.92989

'Positive' Class : 0

For Smote method, this was the least performing of all the algorithms I tried, although for under sampling, it gave the best results

#### Support Vector Machine

Confusion Matrix and Statistics with linear Function

Reference

Prediction 0 1

0 582 44

1 8 398

Accuracy : 0.9496

95% CI : (0.9344, 0.9621)

No Information Rate : 0.5717

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.896

Mcnemar's Test P-Value : 1.212e-06

Sensitivity : 0.9864

Specificity : 0.9005

Pos Pred Value : 0.9297

Neg Pred Value : 0.9803

Prevalence : 0.5717

Detection Rate : 0.5640

Detection Prevalence : 0.6066

Balanced Accuracy : 0.9434

'Positive' Class : 0

Confusion Matrix and Statistics on full unbalanced dataset

Reference

Prediction 0 1

0 279916 46

1 4399 446

Accuracy : 0.9844

95% CI : (0.9839, 0.9848)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.1645

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.98453

Specificity : 0.90650

Pos Pred Value : 0.99984

Neg Pred Value : 0.09205

Prevalence : 0.99827

Detection Rate : 0.98283

Detection Prevalence : 0.98299

Balanced Accuracy : 0.94552

'Positive' Class : 0

Confusion Matrix and Statistics with radial function on validation set

Reference

Prediction 0 1

0 583 37

1 7 405

Accuracy : 0.9574

95% CI : (0.9432, 0.9689)

No Information Rate : 0.5717

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9122

Mcnemar's Test P-Value : 1.232e-05

Sensitivity : 0.9881

Specificity : 0.9163

Pos Pred Value : 0.9403

Neg Pred Value : 0.9830

Prevalence : 0.5717

Detection Rate : 0.5649

Detection Prevalence : 0.6008

Balanced Accuracy : 0.9522

'Positive' Class : 0

Confusion Matrix and Statistics on full unbalanced dataset

Reference

Prediction 0 1

0 282018 40

1 2297 452

Accuracy : 0.9918

95% CI : (0.9915, 0.9921)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.2768

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.9919

Specificity : 0.9187

Pos Pred Value : 0.9999

Neg Pred Value : 0.1644

Prevalence : 0.9983

Detection Rate : 0.9902

Detection Prevalence : 0.9903

Balanced Accuracy : 0.9553

'Positive' Class : 0

With package ‘e1071’

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 577 34

1 13 408

Accuracy : 0.9545

95% CI : (0.9399, 0.9663)

No Information Rate : 0.5717

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9064

Mcnemar's Test P-Value : 0.003531

Sensitivity : 0.9780

Specificity : 0.9231

Pos Pred Value : 0.9444

Neg Pred Value : 0.9691

Prevalence : 0.5717

Detection Rate : 0.5591

Detection Prevalence : 0.5921

Balanced Accuracy : 0.9505

'Positive' Class : 0

Confusion Matrix and Statistics on full unbalanced dataset

Reference

Prediction 0 1

0 279764 29

1 4551 463

Accuracy : 0.9839

95% CI : (0.9835, 0.9844)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.1656

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.98399

Specificity : 0.94106

Pos Pred Value : 0.99990

Neg Pred Value : 0.09234

Prevalence : 0.99827

Detection Rate : 0.98229

Detection Prevalence : 0.98240

Balanced Accuracy : 0.96253

'Positive' Class : 0

#### Neural Network

Resampling results across tuning parameters:

size decay Accuracy Kappa

1 0e+00 0.9529901 0.9035064

1 1e-04 0.9527397 0.9029161

1 1e-01 0.9516633 0.9008986

3 0e+00 0.9512442 0.9005035

3 1e-04 0.9493376 0.8967055

3 1e-01 0.9568891 0.9118888

5 0e+00 0.9546498 0.9075292

5 1e-04 0.9534839 0.9052166

5 1e-01 0.9610262 0.9204907

Accuracy was used to select the optimal model using the largest value.

The final values used for the model were size = 5 and decay = 0.1.

Confusion Matrix and Statistics on validation set

Reference

Prediction 0 1

0 574 19

1 16 423

Accuracy : 0.9661

95% CI : (0.9531, 0.9763)

No Information Rate : 0.5717

P-Value [Acc > NIR] : <2e-16

Kappa : 0.9307

Mcnemar's Test P-Value : 0.7353

Sensitivity : 0.9729

Specificity : 0.9570

Pos Pred Value : 0.9680

Neg Pred Value : 0.9636

Prevalence : 0.5717

Detection Rate : 0.5562

Detection Prevalence : 0.5746

Balanced Accuracy : 0.9649

'Positive' Class : 0

Confusion Matrix and Statistics on full unbalanced dataset

Reference

Prediction 0 1

0 274586 13

1 9729 479

Accuracy : 0.9658

95% CI : (0.9651, 0.9665)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.0865

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.96578

Specificity : 0.97358

Pos Pred Value : 0.99995

Neg Pred Value : 0.04692

Prevalence : 0.99827

Detection Rate : 0.96411

Detection Prevalence : 0.96416

Balanced Accuracy : 0.96968

'Positive' Class : 0

Results of neural network was quite impressive with smote sampling.

#### Tree Bag

On Validation set

Reference

Prediction 0 1

0 583 28

1 7 414

Accuracy : 0.9661

95% CI : (0.9531, 0.9763)

No Information Rate : 0.5717

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9303

Mcnemar's Test P-Value : 0.0007232

Sensitivity : 0.9881

Specificity : 0.9367

Pos Pred Value : 0.9542

Neg Pred Value : 0.9834

Prevalence : 0.5717

Detection Rate : 0.5649

Detection Prevalence : 0.5921

Balanced Accuracy : 0.9624

'Positive' Class : 0

Confusion Matrix and Statistics on full set

Reference

Prediction 0 1

0 279909 11

1 4406 481

Accuracy : 0.9845

95% CI : (0.984, 0.9849)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.1763

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.98450

Specificity : 0.97764

Pos Pred Value : 0.99996

Neg Pred Value : 0.09842

Prevalence : 0.99827

Detection Rate : 0.98280

Detection Prevalence : 0.98284

Balanced Accuracy : 0.98107

'Positive' Class : 0

## Ensembling

Generating prediction from logistic regression and decision tree and then stacking them with gbm

Top layer model:

Stochastic Gradient Boosting

199366 samples

2 predictor

2 classes: 'No', 'Yes'

No pre-processing

Resampling: Cross-Validated (10 fold, repeated 5 times)

Summary of sample sizes: 179430, 179430, 179429, 179429, 179429, 179429, ...

Resampling results across tuning parameters:

interaction.depth n.trees Accuracy Kappa

1 50 0.9982705 0.04474046

1 100 0.9982695 0.04576122

1 150 0.9982615 0.04574661

2 50 0.9982665 0.03542075

2 100 0.9982575 0.03170045

2 150 0.9982565 0.03547326

3 50 0.9983588 0.14383374

3 100 0.9983769 0.15390410

3 150 0.9983819 0.15564427

Tuning parameter 'shrinkage' was held constant at a value of 0.1

Tuning parameter 'n.minobsinnode' was held constant at a value of 10

Accuracy was used to select the optimal model using the largest value.

The final values used for the model were n.trees = 150, interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.

On testing the model, we find the predictions are not good for the fraud cases.

Confusion Matrix and Statistics

Reference

Prediction No Yes

No 85279 145

Yes 15 2

Accuracy : 0.9981

95% CI : (0.9978, 0.9984)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 0.8668

Kappa : 0.024

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.99982

Specificity : 0.01361

Pos Pred Value : 0.99830

Neg Pred Value : 0.11765

Prevalence : 0.99828

Detection Rate : 0.99810

Detection Prevalence : 0.99980

Balanced Accuracy : 0.50671

'Positive' Class : No

#### UnderSample data

On stacking tree and logistics regression model with gbm, we get below result on the undersampled validation set

Confusion Matrix and Statistics

Reference

Prediction No Yes

No 141 13

Yes 6 134

Accuracy : 0.9354

95% CI : (0.9009, 0.9606)

No Information Rate : 0.5

P-Value [Acc > NIR] : <2e-16

Kappa : 0.8707

Mcnemar's Test P-Value : 0.1687

Sensitivity : 0.9592

Specificity : 0.9116

Pos Pred Value : 0.9156

Neg Pred Value : 0.9571

Prevalence : 0.5000

Detection Rate : 0.4796

Detection Prevalence : 0.5238

Balanced Accuracy : 0.9354

'Positive' Class : No

On checking the model on the whole dataset, results are as below:

Confusion Matrix and Statistics

Reference

Prediction No Yes

No 265993 23

Yes 18322 469

Accuracy : 0.9356

95% CI : (0.9347, 0.9365)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.0454

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.93556

Specificity : 0.95325

Pos Pred Value : 0.99991

Neg Pred Value : 0.02496

Prevalence : 0.99827

Detection Rate : 0.93394

Detection Prevalence : 0.93402

Balanced Accuracy : 0.94440

'Positive' Class : No

#### Smote Data

Here again I made a logistic regression and a tree model and then stacked with a GBM model. On testing the stacked GBM model on the validation dataset, I got below results:

Confusion Matrix and Statistics

Reference

Prediction No Yes

No 573 36

Yes 17 406

Accuracy : 0.9486

95% CI : (0.9334, 0.9613)

No Information Rate : 0.5717

P-Value [Acc > NIR] : < 2e-16

Kappa : 0.8946

Mcnemar's Test P-Value : 0.01342

Sensitivity : 0.9712

Specificity : 0.9186

Pos Pred Value : 0.9409

Neg Pred Value : 0.9598

Prevalence : 0.5717

Detection Rate : 0.5552

Detection Prevalence : 0.5901

Balanced Accuracy : 0.9449

'Positive' Class : No

On applying the model developed above the the whole dataset, we get below results:

Confusion Matrix and Statistics

Reference

Prediction No Yes

No 276681 35

Yes 7634 457

Accuracy : 0.9731

95% CI : (0.9725, 0.9737)

No Information Rate : 0.9983

P-Value [Acc > NIR] : 1

Kappa : 0.1036

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.97315

Specificity : 0.92886

Pos Pred Value : 0.99987

Neg Pred Value : 0.05648

Prevalence : 0.99827

Detection Rate : 0.97147

Detection Prevalence : 0.97159

Balanced Accuracy : 0.95101

'Positive' Class : No

## Results

I recorded the sensitivity and specificity for the sampling methods. Below is the tabular presentation of the same.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| For SMOTE Sampling: |  | Logistic Reg | CART  Tree | SVM (lin) | SVM (rad) | SVM (lin) e1071 | Neural Net | Bag Tree | Ensemble |
|  |  |  |  |  |  |  |  |  |  |
| Validation Set | Sensitivity | 0.9797 | 0.9695 | 0.9864 | 0.9881 | 0.978 | 0.9729 | 0.9881 | 0.9712 |
| Specificity | 0.914 | 0.8846 | 0.9005 | 0.9163 | 0.9231 | 0.957 | 0.9367 | 0.9186 |
| Full data | Sensitivity | 0.9787 | 0.9716 | 0.9845 | 0.9919 | 0.984 | 0.9658 | 0.9845 | 0.97315 |
| Specificity | 0.9187 | 0.8882 | 0.9065 | 0.9187 | 0.9411 | 0.9736 | 0.9776 | 0.92886 |
|  |  |  |  |  |  |  |  |  |  |

Table 1 : Comparison of Sensitivity& Specificity for all Classification Algorithms for SMOTE sampling

For Under sampling

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Logistic Reg | CART Tree | SVM (lin) | SVM (rad) | SVM (lin) e1071 | Neural Net | Bag Tree | Ensemble |
|  |  |  |  |  |  |  |  |  |  |
| Validation Set | Sensitivity | 0.9739 | 0.9184 | 1 | 0.9864 | 0.966 | 0.9728 | 0.9524 | 0.9592 |
| Specificity | 0.9275 | 0.9252 | 0.898 | 0.9184 | 0.9252 | 0.898 | 0.9048 | 0.9116 |
| Full data | Sensitivity | 0.964 | 0.9025 | 0.9782 | 0.9737 | 0.95602 | 0.9614 | 0.95156 | 0.93556 |
| Specificity | 0.9207 | 0.9411 | 0.9065 | 0.9167 | 0.93496 | 0.92683 | 0.96748 | 0.95325 |

Table 2 : Comparison of Sensitivity& Specificity for all Classification Algorithms for under sampling

**Conclusion**

Bagged tree was the best performing algorithm for both the smote and under sampling methods.

Neural network performed well with smote sampling and its results were almost same as bagged tree for sampling methodology.

## Further work

1. Time variable can be broken down to hours of day this can be studied further and I feel that can be added as a predictor in the dataset, but I have omitted the time variable as I mainly intend to study techniques for handling unbalanced data.
2. As the purpose of this report was to study the unbalanced dataset and study various methodologies to solve this problem of minority representation of one class, tuning of parameters was not done in detail. Tuning was done by the ‘train’ function of the ‘caret’ package. Tuning can be studied further in detail to further increase the accuracy of the algorithms.
3. Anomaly detection can be studied
4. KNN and Random forest were also intended to be studied, but were later left due to high training and predicting times. These algorithms can also be explored.
5. ROSE method and over sampling and sampling with tomek links were left due to time constraint and can be studied.
6. Ensembles to tackle unbalanced dataset can be studied further. In this document , I only ensemble on under and smote samples data.