CREDIT CARD FRAUD DETECTION

by

Gülsev ÜYE KARA

Institute of Science

ACM 514-Data Mining and Knowledge Acquisition

SEMESTER PROJECT

Istanbul,*2021*

Table of Contents

Abstract

Financial fraud is an ever-growing metus in the financial industry. Credit card frauds are easy and friendly targets.

Data mining had played an important role in the detection of credit card fraud in online transactions. Credit card fraud detection, which is a data mining problem, becomes challenging due to two major reasons - first, the profiles of normal and fraudulent behaviors change constantly, and secondly, credit card fraud data sets are highly skewed. The performance of fraud detection in credit card transactions is greatly affected by the sampling approach on the dataset, selection of variables, and detection technique(s) used. This document investigates the performance of ANN, Decision Tree, k-nearest neighbor, and Random Forest on highly skewed credit card fraud data. The sampling techniques of under-sampling and oversampling smote and undersampling & oversampling together usage is carried out on the skewed data. The four techniques are applied to raw and preprocessed data. The work is implemented in R.

The performance of the techniques is evaluated based on the confusion matrix with accuracy, sensitivity, specificity, and AUC, F1 Score.

Introduction

In this project, fraud detection data analysis investigated and implemented methodologies according to CRISP-DM.

This R project's main aim to build a classifier that can detect credit card fraudulent transactions.

During the regression analysis of fraud detection using a variety of machine learning algorithms that will be able to perceive fraud data from non-fraudulent ones.

1- Business Understanding

Fraud detection is a very risky and important issue today. Digital banking and with the increase of online e-commerce shopping habits digital fraud has become an even greater threat to all of us.

There are numerous types of fraud such as account take over and new account fraud. Companies are estimated to spend >$20bn annually on fraud detection according to researches. That kind of massive loss pushes companies to find new solutions to detect, prevent, and eliminate fraud problems. Machine learning is the most promising technological weapon to fight financial fraud.

Data analysis and kind of pattern recognition are the key steps to building a fraud detection model.

2- Data Understanding

The dataset here contains transactions made by credit cards in September 2013 by European cardholders. This dataset from Kaggle and presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly imbalanced, the positive class (frauds) account for 0.172% of all transactions. “creditcard.csv” Excel dataset glanced with used some handy functions to be able to check standard deviation, variance, length, a summary of dataset. There are not any missing values ("NA") in this dataset.

In the fraud detection project, data investigated in **“DataEvaluation.R ”** file.

"creditcard.csv" contains only numerical input variables which are the result of a PCA transformation. On the other hand, due to confidentiality issues, we do not have access to the original features and more background information about the data.

Features V1, V2, … V28 are the principal components obtained with PCA transformation, 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 ‘Amount’ feature 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. That would be my target class during the modeling phase.

Below given screenshots captured from R codes. Screenshots shows us result of functions summary of “creditcard.csv” dataset, variance and standart deviation results of Class and Amount columns.

```
'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 ...
```

Screenshot- Summary of creditcard.csv dataset

```
summary (creditcard$Amount)
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
   0.00     5.60    22.00    88.35    77.17 25691.16
```

```
> summary (creditcard$Class)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.000000 0.000000 0.000000 0.001728 0.000000 1.000000
```

```
> var(creditcard$Amount)
[1] 62560.07
> var(creditcard$class)
[1] 0.001724507
> sd(creditcard$Amount)
[1] 250.1201
> sd(creditcard$class)
[1] 0.04152719
```

Screenshot- R Codes for Summary&var&sd functions

2.1 – Data Visualization

Data visualization has substantial potential for making the fraudulent transaction detection process more efficient and effective.

While this analysis has focused on detecting fraudulent transactions, I believe that interactive visualization may also be useful at other stages of the fraud investigation process. That's why in this section I want to show some charts and graphs to be able to better understand the "creditcard.csv" dataset. Of the 284807 transactions continued in the dataset only 492 being fraudulent as explained data understanding part. This result makes an idea clearly for this project will work with an unbalanced dataset.

- If want to show this data with percentage on 3D Pie Chart result as seen below figure.

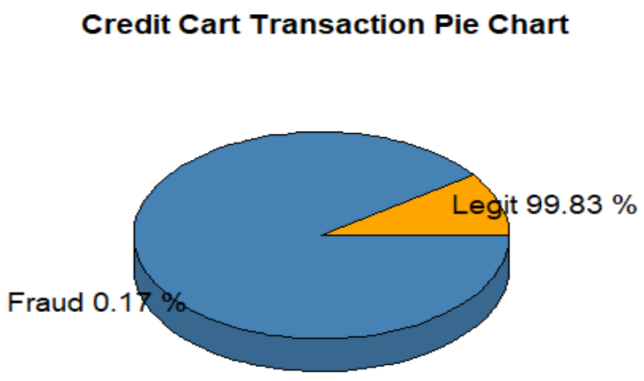


Figure - Credit Card Transactions PieChart

- Box plot figure given above shows us "0" and "1" values amount int the dataset. It is seen clearly there are a lot more variability in the transaction values for non-fraudulent transactions.

Figure- Box Plot Distribution of transaction amount by class

- We clearly see that most of the data features are not correlated via below seen correlation graph. This may be the cause before publishing the dataset, most of the features were presented to a Principal Component Analysis (PCA) algorithm. The features V1 to V28 are most probably the Principal Components resulted after propagating the real features through PCA. We do not know if the numbering of the features reflects the importance of the Principal Components.

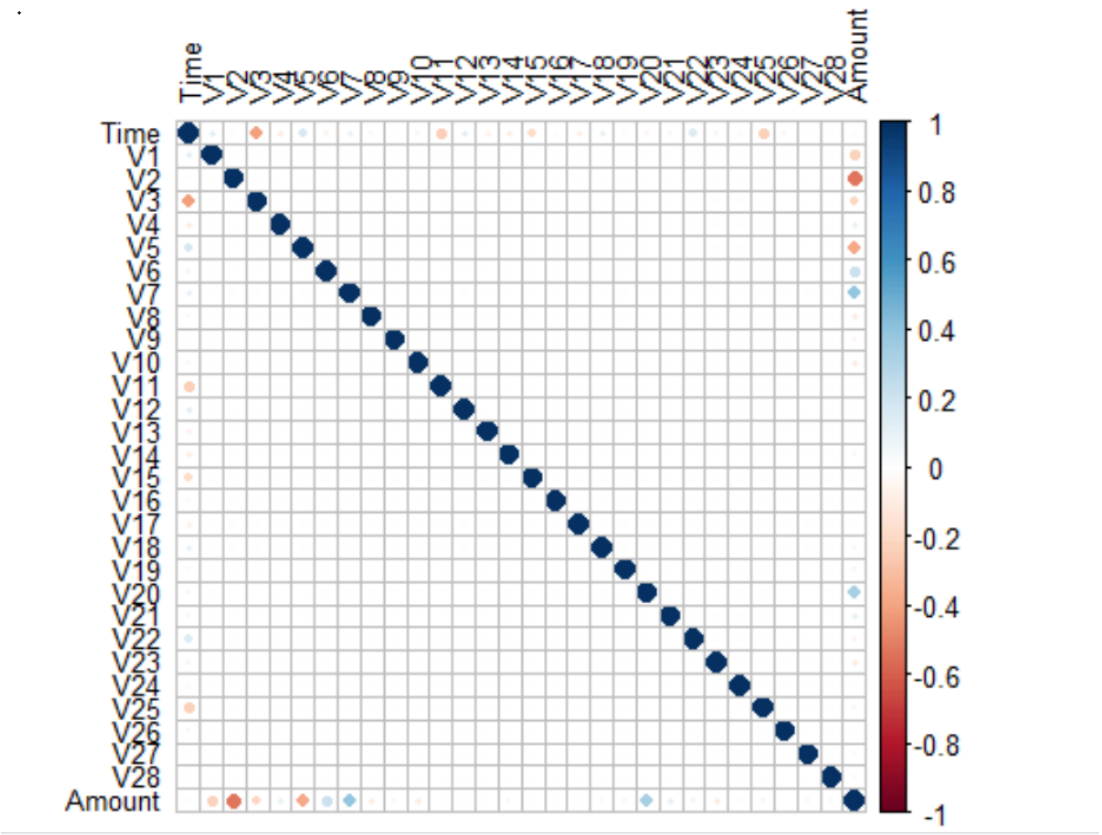


Figure-Correlation Graph

- The below given distribution graph shows us 'Time' feature looks very similar across both types of transactions. One could argue that fraudulent transactions are more uniformly distributed, while normal transactions have a cyclical distribution.

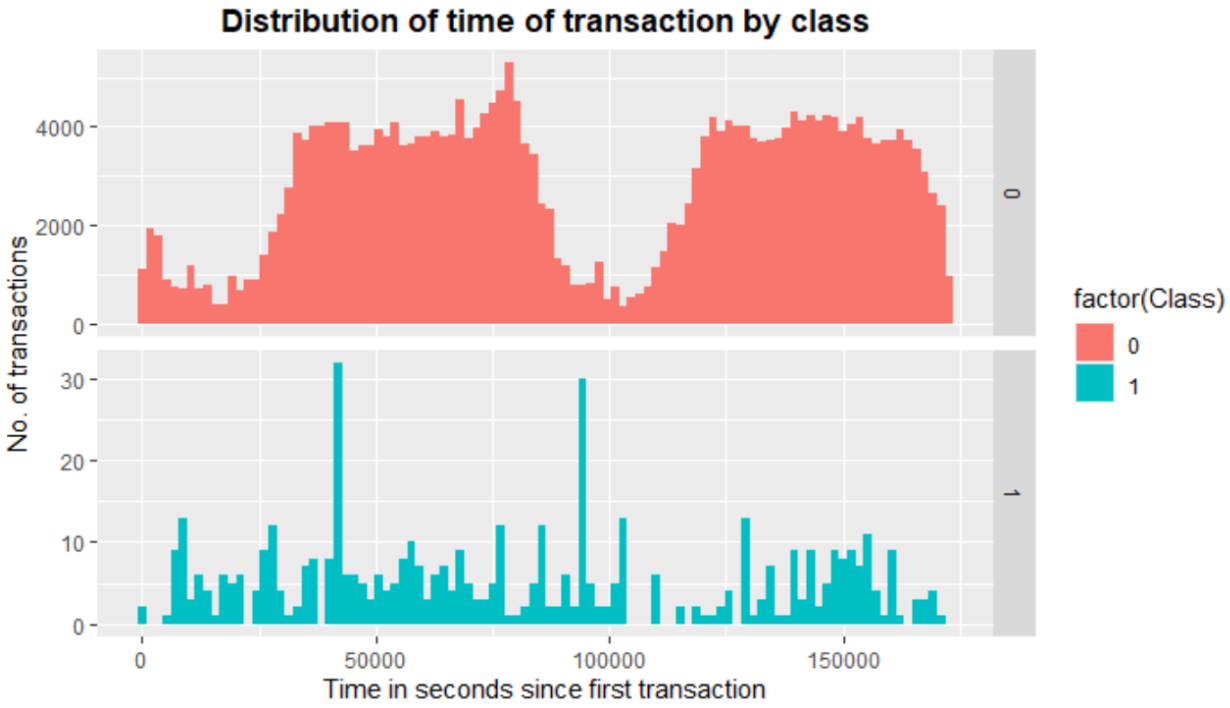


Figure-Distribution Graph of time transaction by class

2.2 - Success Criteria

In this project, we can measure success about Confusion Matrix results and created plots and graphics will guide for understand well results. In this project also can use for measuring the accuracy using the Area Under the Curve (AUC) to support results.

The area under the ROC curve (AUC) results were considered:

- **Excellent** for AUC values between 0.9-1,
- **Good** for AUC values between 0.8-0.9,
- **Fair** for AUC values between 0.7-0.8,
- **Poor** for AUC values between 0.6-0.7
- **Failed** for AUC values between 0.5-0.6.

Confusion matrix accuracy is not meaningful alone for imbalanced classification. That's why when the project should finalize with balanced data and also models should provide us meaningful accuracy, F1-scores,AUC values.

3-Data Preparation

Data prepared for Machine Learning algorithms. In this part, will investigate data test and training partition and what kind of techniques implement for get balanced data . In addition to that, a very critical and important point about these data; Deciding which techniques should use to tackle unbalanced data. You can reach R code of that part via **"Data Preparation.R"** file.

Before to start these steps; in current situation, how is the performance of unbalanced data Iet see here. As seen below attached screenshot accuracy is pretty high but negative prediction value is **"NAN"** . It is not predictable due to **"False Negative**" and **"True Negative"** values are **0**. That is also another problem of the project. For better prediction firstly that dataset need to balance.

```
          Reference
Prediction      0      1
         0 284315    492
         1      0      0

               Accuracy : 0.9983
                 95% CI : (0.9981, 0.9984)
    No Information Rate : 0.9983
    P-Value [Acc > NIR] : 0.512

                  Kappa : 0

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

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.9983
         Neg Pred Value :    NaN
              Precision : 0.9983
                 Recall : 1.0000
                     F1 : 0.9991
             Prevalence : 0.9983
         Detection Rate : 0.9983
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000
```

Figure- Confusion Matrix at the begining

3.1 - Creating Training and Test Dataset

"creditcard.csv" dataset is a very large dataset for my computer. To prevent possible performance problems of the modeling phase; at this step, it is a good idea to work with a specific rows number.

Decided to work with 50000 rows .

```
creditcard <- read.csv("C:/Users/Gulsev/Desktop/ACM514 FINAL PROJECT/creditcard.csv",nrows =50000   ,header=TRUE)
View(creditcard)
row(creditcard)
```

After this process splitted data to create test and training data. Split ratio selected as **80%** for **train** and **20%** will use in **test**.

```
library(caTools)
set.seed(123)
data_sample=sample.split(creditcard$Class,SplitRatio = 0.8)
train_data=subset(creditcard,data_sample==TRUE)
test_data=subset(creditcard,data_sample==FALSE)
```

.

After splitted data as test and training sets latest situation for dimension of test and training sets seen as below captured screenshot.

- "creditcard.csv" dataset has 40000 rows and 31 columns data for **Train.**
- "creditcard.csv" dataset has 10000 rows and 31 columns data for **Test.**
- "creditcard.csv" dataset has levels **"0"→ Legit** and **"1" → Fraud**
- 39882 Legit and 118 Fraud data contain in our **training** dataset for **'Class'** column.
- 9970 Legit and 30 Fraud data contain in our **test** dataset for **'Class'** column.

```
> dim(train_data)
[1] 40000    31
> dim(test_data)
[1] 10000    31
> table(train_data$Class)

    0     1
39882   118
> table(train_data$Class)

    0     1
39882   118
> table(test_data$Class)

   0    1
9970   30
```

Screenshot-Dimension & Table Results in R Code

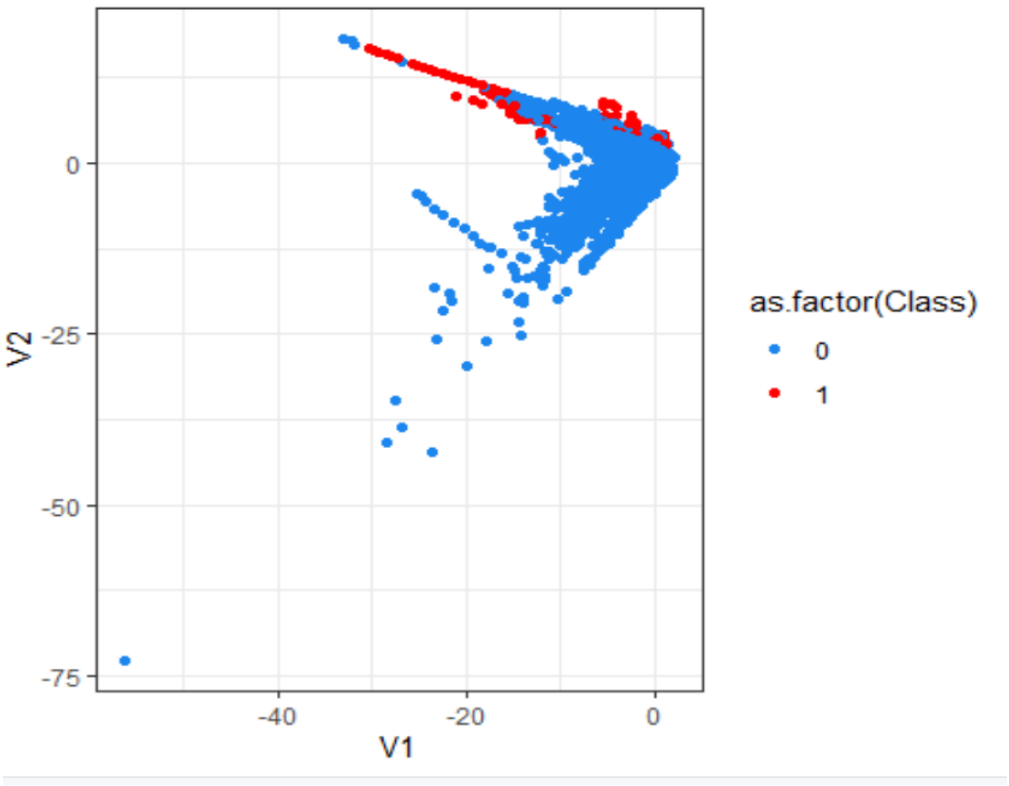


Figure- Scatter Plot of Unbalanced Dataset creditcard

As seen above scatter plot Legit data with blue dots and and Fraud data with red ones. Red dots (Fraud) data has very smaller place in this graph due to inbalanced data stuations.

3.2 – The Approach of Balancing Techniques for Unbalanced Datasets

Standard machine learning algorithms struggle with accuracy on imbalanced data for the following reasons:

- Machine Learning algorithms struggle with accuracy because of the unequal distribution of independent variables. This causes the performance of existing classifiers to get biased towards the majority class.
- The algorithms are accurate driven i.e. they aim to minimize the overall error to which the minority class contributes very little.
- Machine Learning algorithms assume that the data set has balanced class distributions.
- They also assume that errors obtained from different classes have the same cost

The methods to deal with this problem are widely known as **'Sampling Methods'**.

Generally, these methods aim to modify imbalanced data into balanced distribution using some mechanism. The modification occurs by altering the size of the original data set and provide the same proportion of balance.

Below are the methods used in this dataset here to treat the imbalanced dataset:

- Random Oversampling (ROS)
- Random Undersampling &Oversampling Together Usage (ROS& RUS)
- Random Undersampling (RUS)
- Synthetic Data Generation (SMOTE)

3.2.1 Random Oversampling Method (ROS)

Before implementing the Oversampling Method checked Legit (39882) and Fraud (118) data values one more time. According to the training dataset, it is seen that the legit transaction number is 39882 so it is the biggest categoric value ROS will use 39882 for the balanced dataset according to the oversampling method.

This method works with minority class. It replicates the observations from minority class to balance the data. Random oversampling balances the data by randomly oversampling the minority class. Informative oversampling uses a pre-specified criterion and synthetically generates minority class observations.

An **advantage** of using this method is that it leads to **no information loss**. **The disadvantage** of using this method is that, since oversampling simply **adds replicated observations in original data set**, it ends up adding multiple observations of several types, **thus leading to overfitting.**

As seen below screenshot regarding formula 39882 will divide by 0.50. So total dataset value is 79764 after this calculation and that amount dived by two for legit and fraud data.

At the end of this method implementation, the new result of the "oversampling "Class" column is 39882 for each level "0" and "1."

```
> #Random Over Sampling Method (ROS )
> table(train_data$Class)

    0     1
39882   118
> #  0     1
> # 39882   118
> no_legit<-39882
> new_frac_legit<-0.50
> new_n_total<-no_legit/new_frac_legit # = 39882/0.50
> print(new_n_total)
[1] 79764
> #new_n_total: 79764
> library(ROSE)
> oversampling_result<-ovun.sample(Class~.,data=train_data,method="over",N=new_n_total,seed=2019)
> oversampled_credit<-oversampling_result$data
> table(oversampled_credit$Class)

    0     1
39882 39882
>
```

Screenshot- Random Oversampling Method in R Code (ROS)

After Random Oversampling Method implementation drawed a scatter plot for observed the data distribution.

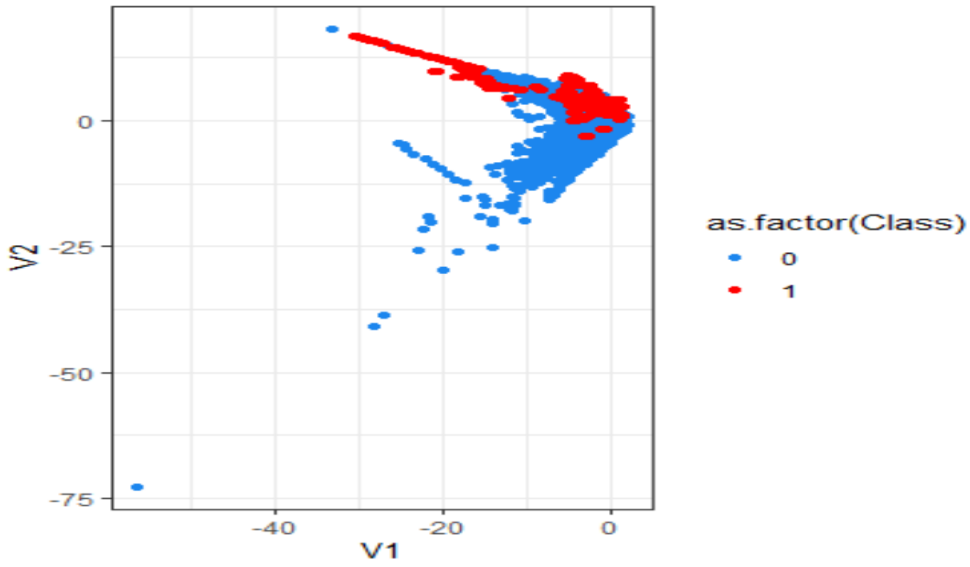


Figure- Scatter Plot After Implement Random Oversampling Method

As seen above scatter plot red dots are getting bigger after applied ROS method if compared result with "Scatter Plot of Unbalanced Dataset creditcard figure".

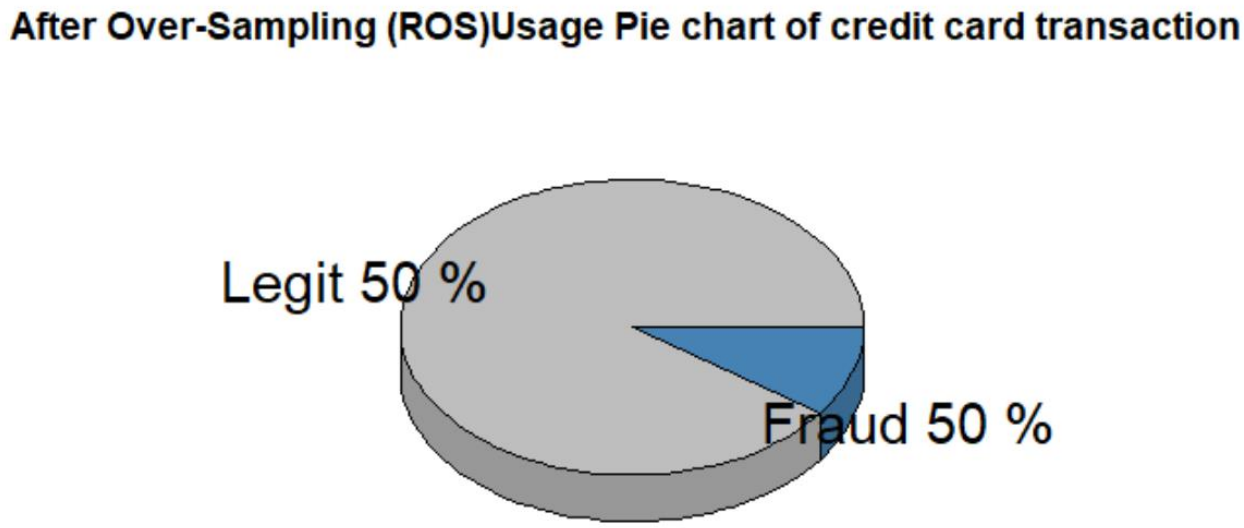


Figure -After Oversampling Usage Pie Chart

As seen above figure Pie chart show us balanced data rates for Level Legit and Fraud datas.

3.2.2 Random Undersampling Method (RUS)

This method reduces the number of observations from majority class to make the data set balanced. **This is best** to use when the **data set is huge** and reducing the number of training samples helps to **improve run time** and storage troubles.

Below screenshots shows implementation of Random Undersampling method.The formula is same with Random Oversampling. The main differences in this method is this time we put the formula fraud value or in other wors smallest cateoric value (fraud:118 observations). "Class" column has 118 observations for each level "0" and "1."

```
> # Random Under- Sampling(RUS)Method
> table(train_data$Class)

    0     1
39882   118
> #Result:39882(0)  118 (1)
> no_fraud<-118
> new_frac_fraud<-0.50
> new_n_total<-no_fraud/new_frac_fraud
> undersampling_result<-ovun.sample(Class~.,data=train_data,method="under",N=new_n_total,seed=123)
> undersampled_credit<-undersampling_result$data
> table(undersampled_credit$Class)

  0   1
118 118
```

Screenshot-Random Undersampling Method R Code

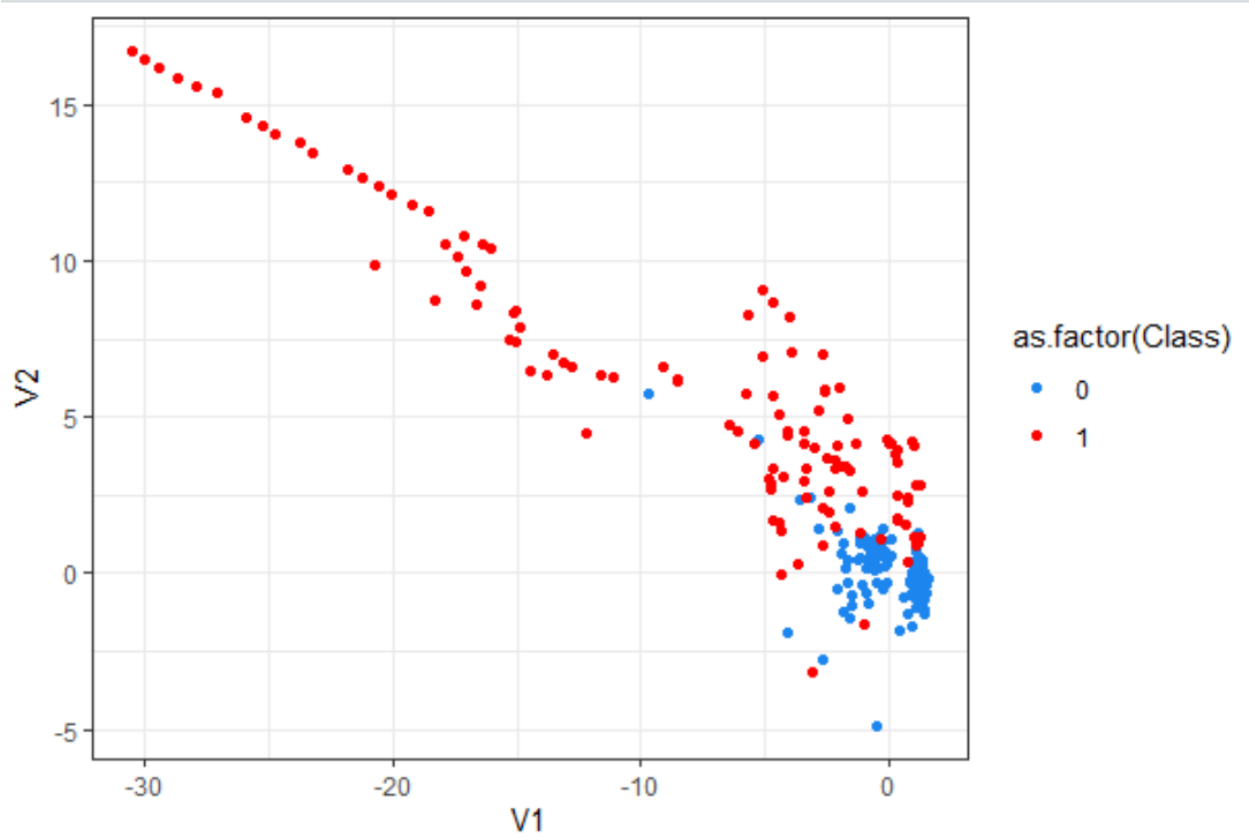


Figure- Scatter Plot Random Undersampling (RUS)

After implemented Random Undersampling Method, it is clearly seen that the red dots which represent us fraud data are scattered over a large area, and blue dots are seen in a small group and more regular situation than the previous ROS method's plot.

3.2.3 ROS & RUS Method Both Usage

This time for Ros and Rus usage after formula Fraud observations value is 19868 and Legit observations value is 20132 . Due to in dataset fraud observations less than legit transactions this method keep it but with different way than individual usage of method ROS and RUS.

```
> ##############################################
> #Performance both ROS AND RUS
> table(train_data$Class)

    0     1
39882   118
> #Result:39882(0)  118 (1)
> new<-nrow(train_data)
> new
[1] 40000
> #new shows 40000
> fraction_fraud_new<-0.50
> sampling_result<-ovun.sample(Class~.,data=train_data,method = 'both',N=new,p=fraction
_fraud_new,seed=2019)
> sampling_credit_result<-sampling_result$data
> table(sampling_credit_result$Class)

    0     1
20132 19868
```

Screenshot: ROS&RUS Method Implementation R Code

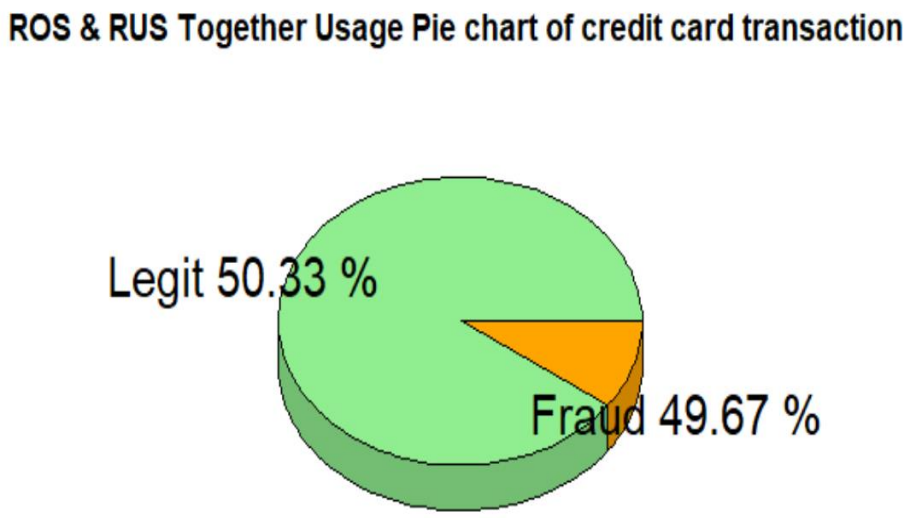


Figure: 3D Pie Chart for ROS&RUS Method Usage Together

As seen above pie chart the observation ratio given . For Fraud (level "1 ") this ratio is 49.67% and for Legit (level "0") observations ratio is 50.33%. This time still these two categoric data has balanced value but this time the balance rate is not exactly 50% like implemented other previous two methods.

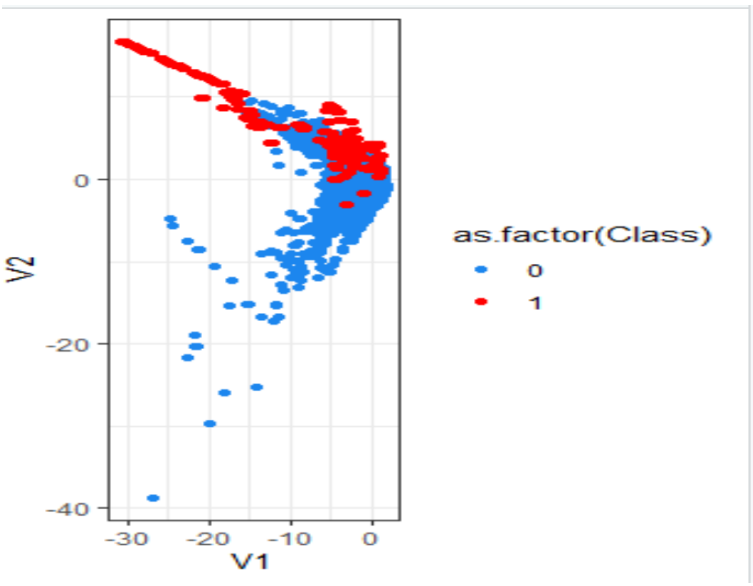


Figure- ROS&RUS Together Implementation Scatter Plot

3.2.4 Smote Method Usage

Instead of replicating and adding the observations from the minority class, it overcomes imbalances by generates artificial data. It is also a type of oversampling technique.

In regards to synthetic data generation, the synthetic minority oversampling technique (SMOTE) is a powerful and widely used method. SMOTE algorithm draws artificial samples by choosing points that lie on the line connecting the rare observation to one of its nearest neighbors in the feature space. ROSE (random over-sampling examples) uses smoothed bootstrapping to draw artificial samples from the feature space neighborhood around the minority class.

Smote method implemented as for that formula given below. Ratio selected in the formula this time Legit- 60% & Fraud-40% .

```
> library(smotefamily)
> table(train_data$Class)

    0     1
39882   118
> table(train_data$Class)

    0     1
39882   118
> #Result:39882(0)  118 (1)
> n0<-39882 #= legit cases
> n1<-118    #=fraud cases
> r0<-0.6
> ntimes<-((1-r0)/r0)*(n0/n1)-1
> smote_output=SMOTE(X=train_data[,-c(1,31)],target=train_data$Class,K=3,dup_size=ntimes)
> credit_smote_result<-smote_output$data
> colnames(credit_smote_result)[30]<-"class"
> prop.table(table(credit_smote_result$Class))

        0         1
0.6003432 0.3996568
```

Screenshot- Smote Method Implementation in R Code

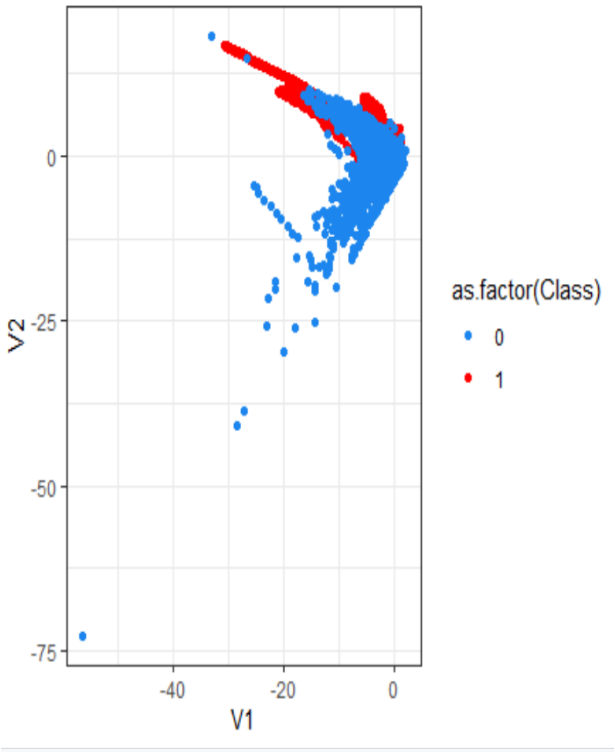


Figure- Scatter Plot of Smote Method

As seen above figure blue dots scattered huge area and this time red dots again in small area but more regular than other plots.

Now the dataset is prepared the next step is to fit the dataset to a model.

4 – Modelling

In this part, ANN, Decision Tree, KNN , Random Forest modeling techniques will show with performance evaluations.

4.1. Artifical Neural Network Modeling (ANN)

You can reach ANN algorithm modeling details via "**ANNAlgorithm_Investigation.R"** file.

- When neuron number increased like 10 for ROS &RUS together ,ROS and Smote methods observed performance problem of computer during modeling execution in R.
- That's why I decided to continue this model with Random Undersampling method for continue with better performance.

Scenario 1: ANN Modeling executed with Random Under Sampling for 10 neuron number and with 80% training -20% test data.

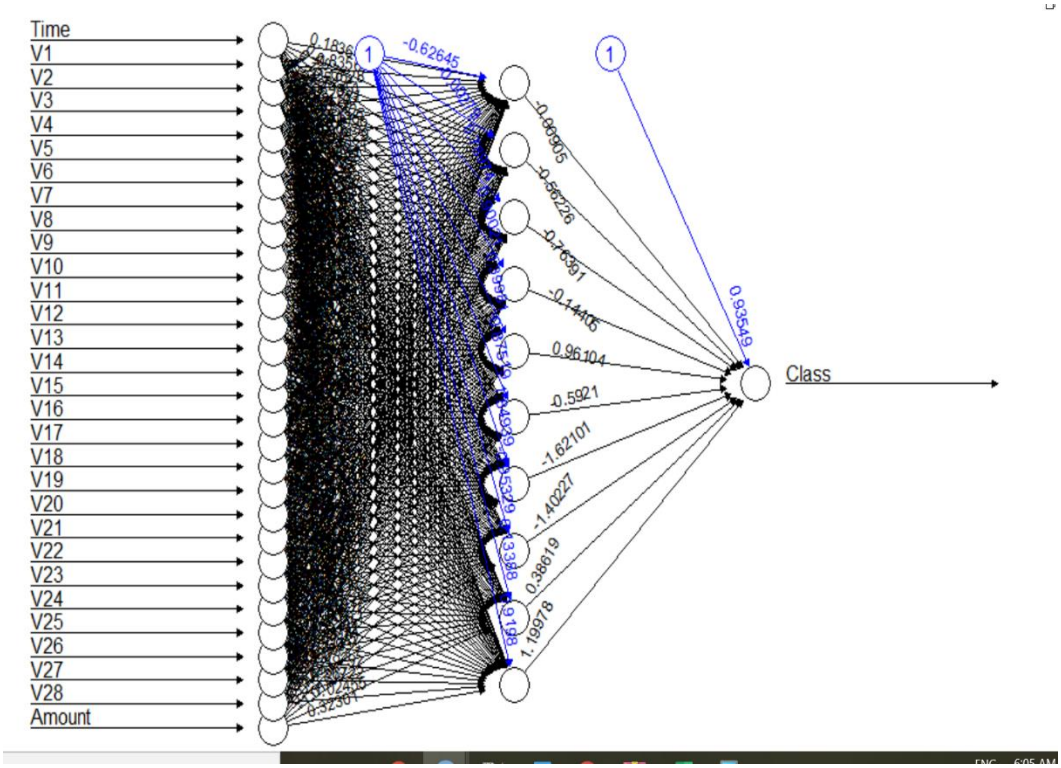


Figure- ANN Pilot for 10 neuron &undersampling method

```
Confusion Matrix and Statistics

          Real/Actual Values
Predictions    0    1
          0   20    0
          1 9950   30

               Accuracy : 0.005
                 95% CI : (0.0037, 0.0066)
    No Information Rate : 0.997
    P-Value [Acc > NIR] : 1

                  Kappa : 0

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

            Sensitivity : 0.002006
            Specificity : 1.000000
         Pos Pred Value : 1.000000
         Neg Pred Value : 0.003006
             Prevalence : 0.997000
         Detection Rate : 0.002000
   Detection Prevalence : 0.002000
      Balanced Accuracy : 0.501003

       'Positive' Class : 0
```

Screenshot- Confusion Matrix ANN Model Result for 10 neuron

As seen above screenshot Confusion Matrix result is terrible for 80%train & 20%test data set for ANN modeling. Accuracy is very very low, Kappa result is not acceptable. On the other hand, when checking the confusion matrix table false-positive observation number is 0. After seen this problem I thought that test data will be not enough for modeling.

Scenario 2: And then I decide to change the split ratio of test & training data to 65% for training and 35% for testing for same sampling model (Random Under Sampling) .
Herby, the ANN model will have more data for testing. After that changing the ANN model re-executed for being able to see the effect of the new test& training ratio on performance.

```
          Real/Actual Values
Predictions     0     1
          0 16876    14
          1   572    38

               Accuracy : 0.9665
                 95% CI : (0.9637, 0.9691)
    No Information Rate : 0.997
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1099

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

            Sensitivity : 0.9672
            Specificity : 0.7308
         Pos Pred Value : 0.9992
         Neg Pred Value : 0.0623
             Prevalence : 0.9970
         Detection Rate : 0.9643
   Detection Prevalence : 0.9651
      Balanced Accuracy : 0.8490

       'Positive' Class : 0
```

Screenshot: Confusion Matrix result for 10 neuron with 65% training - 35% test dataset used model RUS

```
> roc.curve(test_data$Class,predANN_class, plotit = TRUE)
Area under the curve (AUC): 0.849
> F1_Score(predANN_class,test_data$Class,positive = NULL)
[1] 0.9829344
```

Screenshot : AUC & F1 Score values with 65% training - 35% test dataset used model RUS

As seen above screenshot after changed test-training set ratio as explained for same modeling Random Under Sampling with same neuron number accuracy affected very positively also other results increased. In the previous result false-positive observation was 0 , in current situation it seen as 14 observations.

Result : 80% training -20% test data partition is not a good idea for ANN due to poor and incorrect Confusion Matrix calculations.

Scenario 3: Changed neuron number as 5 for training 65% and test 35% data ratio.
In this scenario, I want to observe that what is the **effect of neuron number changing on performance** for the same modeling type Random Undersampling.

```
library(neuralnet)

set.seed(1)

ANN_model <- neuralnet (Class~.,undersampled_credit,hidden = c(5), act.fct = "logistic",linear.output=FALSE)

plot(ANN_model)
```

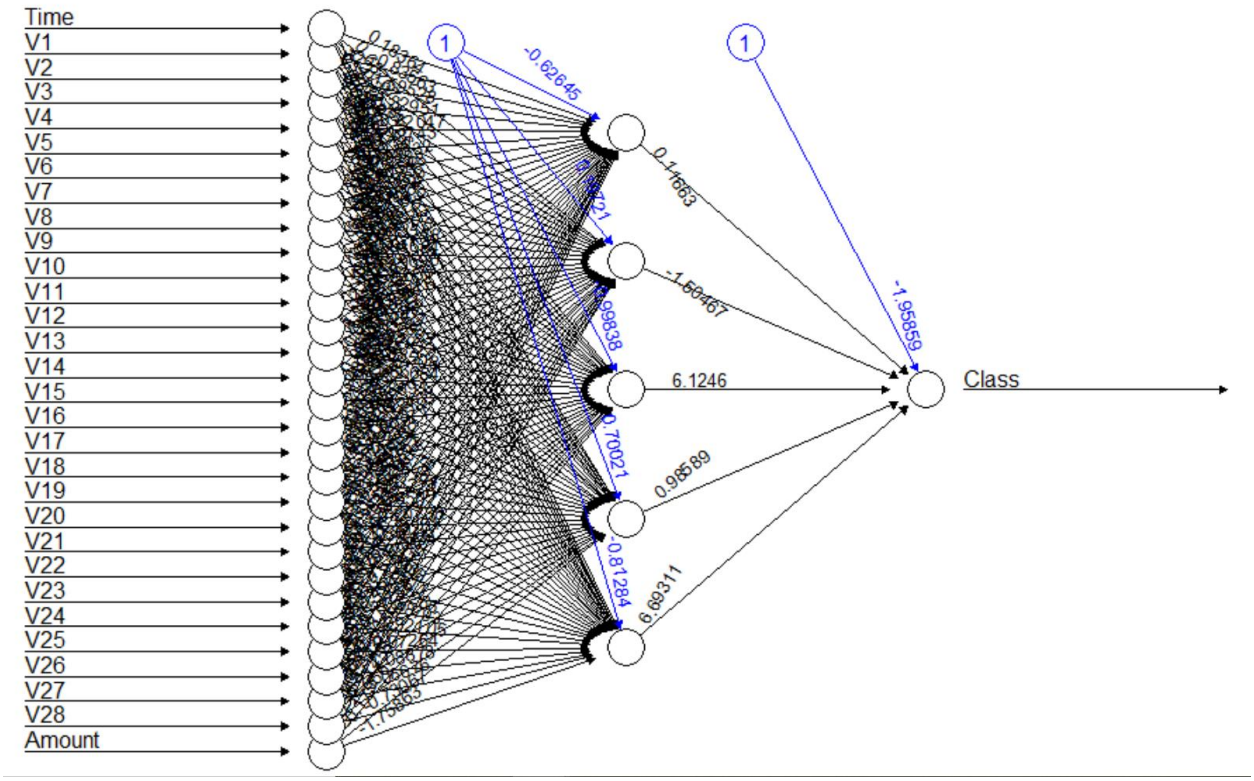


Figure- ANN Graph for 5 neuron number on UnderSampling Model

```
           Real/Actual Values
Predictions     0     1
          0 16882     9
          1   566    43

               Accuracy : 0.9671
                 95% CI : (0.9644, 0.9697)
    No Information Rate : 0.997
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1253

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

            Sensitivity : 0.96756
            Specificity : 0.82692
         Pos Pred Value : 0.99947
         Neg Pred Value : 0.07061
             Prevalence : 0.99703
         Detection Rate : 0.96469
   Detection Prevalence : 0.96520
      Balanced Accuracy : 0.89724

       'Positive' Class : 0
```

```
> roc.curve(test_data$Class,predANN_class, plotit = TRUE)
Area under the curve (AUC): 0.897
> library(MLmetrics)
> F1_Score(predANN_class,test_data$Class,positive = NULL)
[1] 0.9832552
```

Screenshot: Confusion Matrix, AUC ,F1-Score result for 5 neuron with 65% training - 35% test dataset used model RUS

As seen above Confusion Matrix ANN model performance getting better for neuron number 5 instead of 10. Model's accuracy getting high and can be able to predict more observations correctly when compared with previous matrix sample. F1-Score and AUC results also high a little bit more.

4.2. Decision Tree Modeling

Decision Trees to plot the outcomes of a decision. These outcomes are basically a consequence through which we can conclude as to what class the object belongs to. The rpart. plot() function will use the plot it using. Here, specifically will use the recursive parting to plot the decision tree. Test and train data ratio is still same as explained data preparation part .(%80-train- %20 test).

You can reach R code on **"DecisionTreeAlgorithm_Investigation. R"** file.

Scenario 1- Random Undersampling model used for Decision Tree Model algorithm

As seen on summary function we have 236 observations in total left son has 121 observation and right son has 115 .These function gives us all details information for leaves of decision tree.

```
> #credit_smote_result
> summary(decisionTree_model)
Call:
rpart(formula = Class ~ ., data = undersampled_credit, method = "class")
  n= 236

        CP nsplit  rel error     xerror       xstd
1 0.940678      0 1.00000000 1.18644068 0.06395310
2 0.010000      1 0.05932203 0.07627119 0.02493424

Variable importance
V14  V3 V10 V11 V12  V2
 18  17  17  17  16  16

Node number 1: 236 observations,    complexity param=0.940678
  predicted class=0  expected loss=0.5  P(node) =1
    class counts:   118   118
   probabilities: 0.500 0.500
  left son=2 (121 obs) right son=3 (115 obs)
  Primary splits:
      V14 < -1.644088  to the right, improve=104.48280, (0 missing)
      V10 < -1.753934  to the right, improve= 97.36807, (0 missing)
      V11 < 2.782959   to the left,  improve= 96.21538, (0 missing)
      V4  < 1.69639    to the left,  improve= 95.32988, (0 missing)
      V3  < -0.7964642 to the right, improve= 93.49263, (0 missing)
    Surrogate splits:
        V3  < -0.7964642 to the right, agree=0.975, adj=0.948, (0 split)
        V10 < -1.545611  to the right, agree=0.970, adj=0.939, (0 split)
        V11 < 2.711097   to the left,  agree=0.962, adj=0.922, (0 split)
        V12 < -3.5777    to the right, agree=0.945, adj=0.887, (0 split)
        V2  < 1.52308    to the left,  agree=0.941, adj=0.878, (0 split)

  Node number 2: 121 observations
    predicted class=0  expected loss=0.04132231  P(node) =0.5127119
      class counts:   116     5
     probabilities: 0.959 0.041

  Node number 3: 115 observations
    predicted class=1  expected loss=0.0173913  P(node) =0.4872881
      class counts:     2   113
     probabilities: 0.017 0.983
```

Screenshot- Random Undersampling model summary for Decision Tree Model

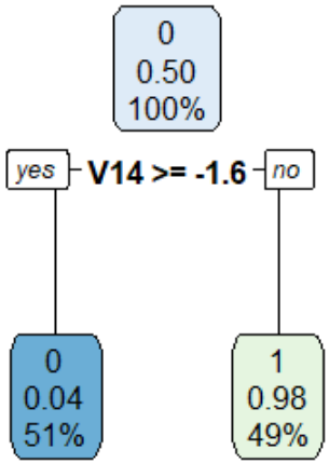
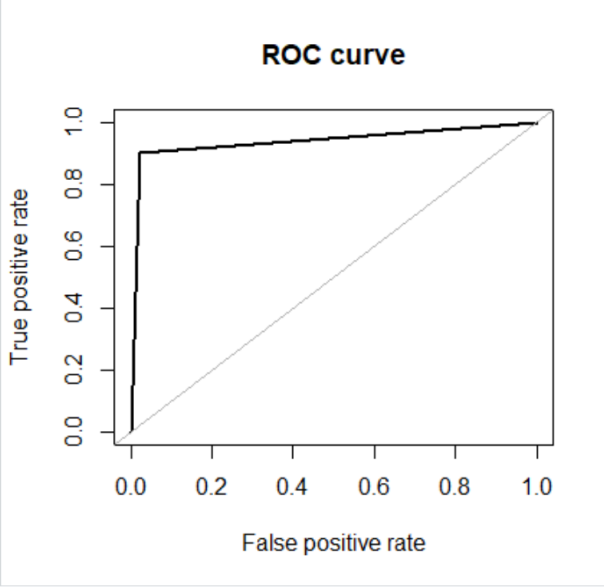


Figure- Decision Tree Graph for Random Under Sampling Model

As seen above figure seen that decision tree graph with 3 nodes and with Level "0"-> Legit and Level "1"-> Fraud possibilities. It will investigate at the next scenario of Decision Tree; what will happen if change sampling method.

```
          Actual Values
Predictions    0    1
          0 9735    3
          1  235   27

               Accuracy : 0.9762
                 95% CI : (0.973, 0.9791)
    No Information Rate : 0.997
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1805

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

            Sensitivity : 0.9764
            Specificity : 0.9000
         Pos Pred Value : 0.9997
         Neg Pred Value : 0.1031
              Precision : 0.9997
                 Recall : 0.9764
                     F1 : 0.9879
             Prevalence : 0.9970
         Detection Rate : 0.9735
   Detection Prevalence : 0.9738
      Balanced Accuracy : 0.9382
```

Screenshot- Confusion Matrix Results for Decision Tree- RUS Method



Screenshot- ROC Curve for Decision Tree Algorithm Random Under Sampling Method

```
> roc.curve(test_data$Class, predicted_value, plotit = TRUE)
Area under the curve (AUC): 0.938
```

```
> F1_Score(predicted_value,test_data$Class,positive = NULL)
[1] 0.9879237
```

Screenshots - F1 -Score and AUC results in R code for Decision Tree

Scenario 2: Smote Method Implementation for Decision Tree

This time Decision tree implemented with smote technique. It is seen that this time node number increase and the nodes observation numbers changed. The below taken screenshots shows us node 10 and node 11 's observation properties and probabilities.

```
  left son=10 (825 obs) right son=11 (997 obs)
  Primary splits:
      Amount < 3.839116   to the right, improve=497.3310, (0 missing)
      V8     < 0.1231969  to the right, improve=491.4122, (0 missing)
      V22    < -0.5209288 to the right, improve=415.4145, (0 missing)
      V4     < 2.318645   to the left,  improve=411.4705, (0 missing)
      V28    < 0.04973175 to the left,  improve=383.2639, (0 missing)
  Surrogate splits:
      V5  < 0.1377616  to the left,  agree=0.853, adj=0.675, (0 split)
      V28 < 0.04973175 to the left,  agree=0.818, adj=0.599, (0 split)
      V8  < 0.1231969  to the right, agree=0.807, adj=0.575, (0 split)
      V22 < -0.5209288 to the right, agree=0.796, adj=0.549, (0 split)
      V6  < -0.1773774 to the right, agree=0.788, adj=0.532, (0 split)

Node number 10: 825 observations
  predicted class=0  expected loss=0.06424242  P(node) =0.01241871
    class counts:   772    53
   probabilities: 0.936 0.064

Node number 11: 997 observations
  predicted class=1  expected loss=0.1935807  P(node) =0.01500783
    class counts:   193   804
   probabilities: 0.194 0.806
```

Screenshot -Summary function result for Decision Tree with Smote Method Implementation

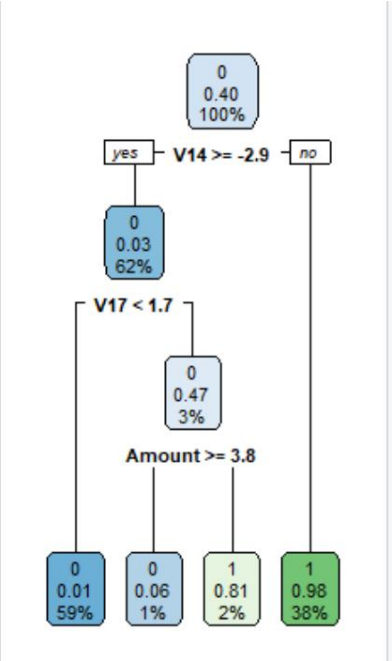


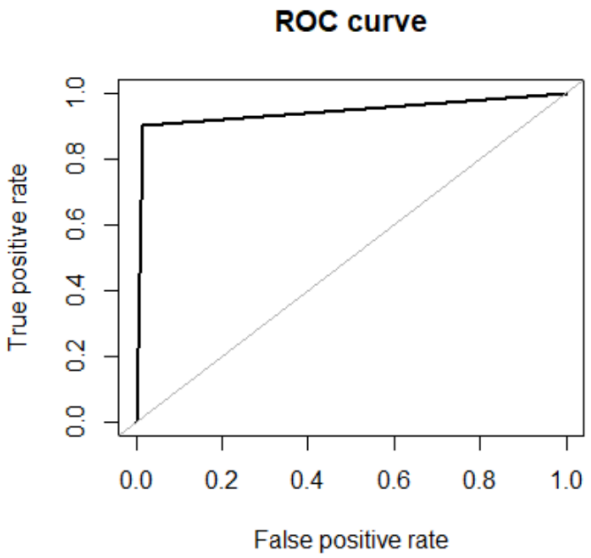
Figure- Decision Tree Graph for Smote Method

As you can see above this time due to changes in sample number due to changed used sampling method as Smote technique. The decision tree node number also changed. This time the decision tree has 11 nodes.

Let us see how will affect the decision tree's performance that new implemented technique.

```
          Actual Values
Predictions    0    1
          0 9817    3
          1  153   27

               Accuracy : 0.9844
                 95% CI : (0.9818, 0.9867)
    No Information Rate : 0.997
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2533

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

            Sensitivity : 0.9847
            Specificity : 0.9000
         Pos Pred Value : 0.9997
         Neg Pred Value : 0.1500
              Precision : 0.9997
                 Recall : 0.9847
                     F1 : 0.9921
             Prevalence : 0.9970
         Detection Rate : 0.9817
   Detection Prevalence : 0.9820
      Balanced Accuracy : 0.9423
```

Screenshot- Confusion Matrix Smote Method Results



Screenshot- ROC Curve for Decision Tree Algorithm Smote Method

```
> F1_Score(predicted_value,test_data$Class,positive = NULL)
[1] 0.9921172
> roc.curve(test_data$Class, predicted_value, plotit = TRUE)
Area under the curve (AUC): 0.942
```

Screenshots - F1 -Score and AUC results in R code for Smote Method Decision Tree

As seen above screenshot Smote Method gives us better result than Random undersampling method.

Smote method has more observation numbers than random undersampling method. That's situation is cause of node number of tree. The node number increased with Smote technique and that made the decision tree structure more consistent. Accuracy and F1 Score and Area under curve results are better than Random Under Sampling method.

Scenario 3 : Tried RUS&ROS model together for Decision Tree

Accuracy : 0.9754

95% CI : (0.9722, 0.9783)

No Information Rate: 0.997

F1- Score: 0.9875114

Scenario 4: Tried Random Oversampling (ROS)Model

Accuracy : 0.9749

95% CI : (0.9716, 0.9779) No Information Rate : 0.997

F1- Score :0.9872544

As a result, we can say that the better option is to use the Smote Method cause to get better accuracy

and F1-score value on the Decision Tree algorithm.

4.3. K-Nearest Neighbors (KNN) Modeling

In this section KNN Model applied to fraud detection data. You can reach R code of that part via **"knnAlgotihm_Investigation "** file.

Scenario 1: Used Repeated K-fold cross-validation method . The process of splitting the data into k-folds can be **repeated a number of times**, this is called repeated **k-fold cross validation.** The following R code uses 10-fold cross validation with 3 repeats. In addition to that Random Undersampling Method also used for sampling.

KNN modeling applied for neighborhood number 3.

```
set.seed(250)
ctrl_knn <- trainControl(method = "repeatedcv", repeats = 10, number = 3)
model_knn <- train(Class~., data = undersampled_credit, method = "knn",preProcess = c("center","scale"),
pred_knn<- predict(model_knn, test_data[,-31])
```

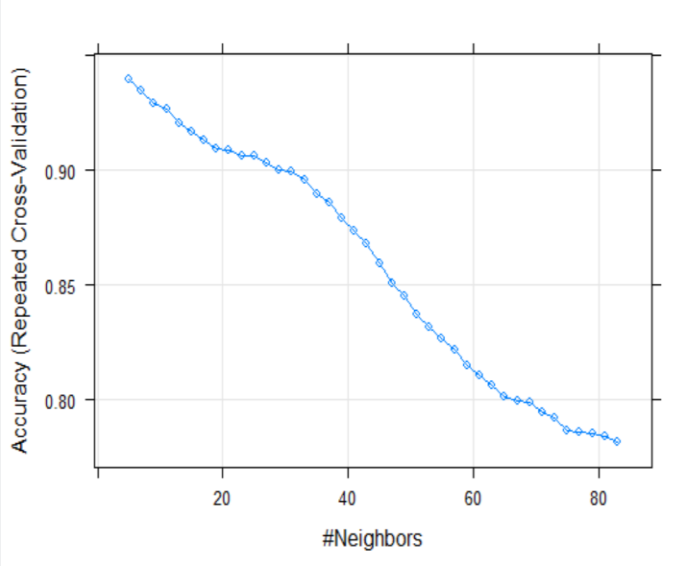


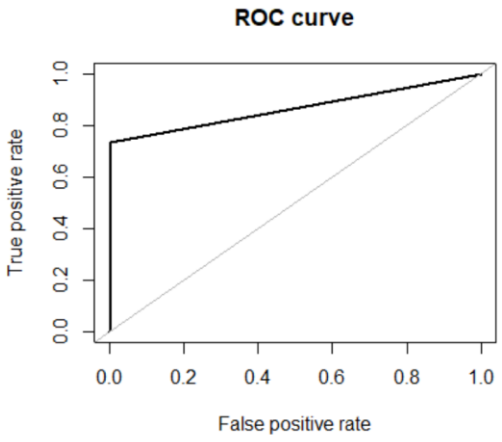
Figure- Accuracy vs Neighbors – KNN Algorithm for Random Undersampling method

```
               Accuracy : 0.9937
                 95% CI : (0.9919, 0.9952)
    No Information Rate : 0.9923
    P-Value [Acc > NIR] : 0.05781

                  Kappa : 0.4087

 Mcnemar's Test P-Value : 6.814e-09

            Sensitivity : 0.2857
            Specificity : 0.9992
         Pos Pred Value : 0.7333
         Neg Pred Value : 0.9945
              Precision : 0.7333
                 Recall : 0.2857
                     F1 : 0.4112
             Prevalence : 0.0077
         Detection Rate : 0.0022
   Detection Prevalence : 0.0030
      Balanced Accuracy : 0.6425

       'Positive' Class : 1
```

Screenshot- KNN Confusion Metrix Result for Random Undersampling Method

As seen above confusion matrix results given. In this results accuracy seen very good but it is not enough for us. In this project we should never forget that working with unbalanced data even implement sampling methods. For that kind of unbalanced data F1- Score also another important measurement for us . F1 score is seen below code that 0.9968. That is engrossing for this modeling .

On the other hand, another measurement is AUC value in Roc curve. Roc curve graph seen as below , AUC value is 0.864. According to AUC criteria it is good but would be better according to determined success criteria of AUC (success criteria explained previous pages for AUC).



```
> roc.curve(test_data$Class, pred_knn, plotit = TRUE)
Area under the curve (AUC): 0.864

> F1_Score(pred_knn,test_data$Class,positive = NULL)
[1] 0.9968331
```

Scenario 2: Smote and Random Over Sampling method implemented KNN.

Due to the performance problem of the computer, it is not going to be my choice this time for the KNN model. Model execution takes so much time.

Scenario 3: Repeats time changed and checked confusion matrix result with Random undersampling method. Result seen that exactly same with first scenario.

```
> set.seed(250)
> ctrl_knn <- trainControl(method = "repeatedcv", repeats = 5, number = 3)
> model_knn <- train(Class~., data = undersampled_credit, method = "knn",preProcess = c("center","scale"),
  trControl = ctrl_knn, tuneLength = 40)
> pred_knn<- predict(model_knn, test_data[,-31])
```

Scenario 4: K value changed as 9 and repeats still 10. In this situation, confusion matrix and AUC results still same with first scenario.

```
set.seed(250)
ctrl_knn <- trainControl(method = "repeatedcv", repeats = 10, number = 9)
model_knn <- train(Class~., data = undersampled_credit, method = "knn",preProcess = c("center","scale
pred_knn<- predict(model_knn, test_data[,-31])
```

To sum up, we can consider found accuracy, F1 Score , AUC results on first scenario for KNN model and it is seen that changing repeat time / k numbers is not changed anything on performance evaluation.

4.4. Random Forest Modeling

You can reach Random Forest Modeling R code via **"RandomForestAlgorithm_Investigation.R"** file.

Scenario 1: Random forest modeling checked with Randon undersampling method for tree number 775 as seen below R code.

```
model_rf <- randomForest(Class~., undersampled_credit, importance = T, nPerm = 4, type = "classification", ntree = 775,
```

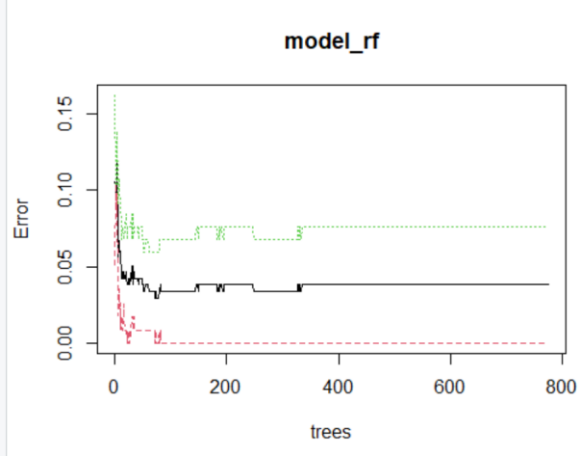
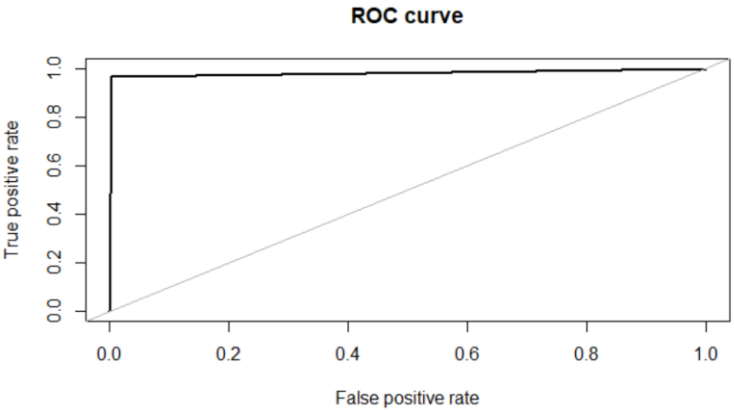


Figure- Random Forest errors vs tree number relation



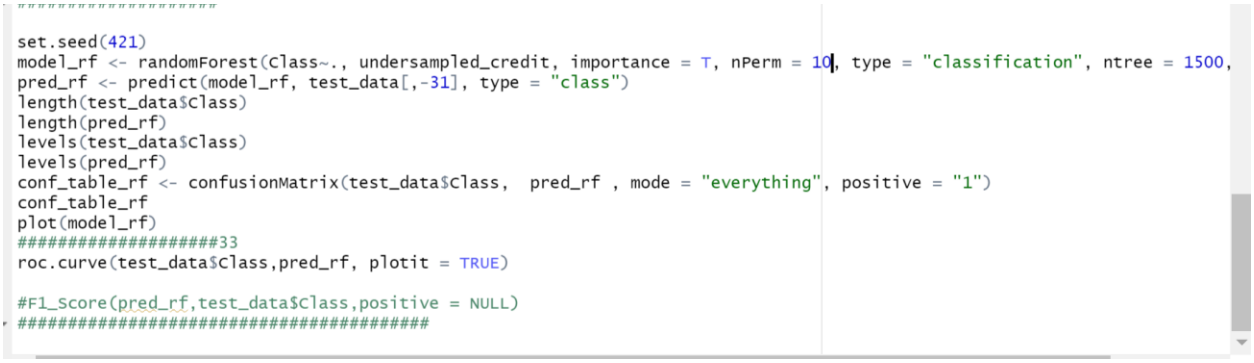

```
Area under the curve (AUC): 0.981
>
```

```
          Reference
Prediction    0    1
         0 9933   37
         1    1   29

               Accuracy : 0.9962
                 95% CI : (0.9948, 0.9973)
    No Information Rate : 0.9934
    P-Value [Acc > NIR] : 0.0001251

                  Kappa : 0.6025

 Mcnemar's Test P-Value : 1.365e-08

            Sensitivity : 0.4394
            Specificity : 0.9999
         Pos Pred Value : 0.9667
         Neg Pred Value : 0.9963
              Precision : 0.9667
                 Recall : 0.4394
                     F1 : 0.6042
             Prevalence : 0.0066
         Detection Rate : 0.0029
   Detection Prevalence : 0.0030
      Balanced Accuracy : 0.7196
```

```
> F1_Score(pred_rf,test_data$Class,positive = NULL)
[1] 0.9980405
```

Screenshot- Random Forest Confusion Matrix, F1 Score performance evaluation for 775 tree number

Scenario 2: Tried with change ntree as 1500 and then changed nPerm value as 10.



```
#####################

set.seed(421)
model_rf <- randomForest(Class~., undersampled_credit, importance = T, nPerm = 10, type = "classification", ntree = 1500,
pred_rf <- predict(model_rf, test_data[,-31], type = "class")
length(test_data$Class)
length(pred_rf)
levels(test_data$Class)
levels(pred_rf)
conf_table_rf <- confusionMatrix(test_data$Class,  pred_rf , mode = "everything", positive = "1")
conf_table_rf
plot(model_rf)
####################33
roc.curve(test_data$Class,pred_rf, plotit = TRUE)

#F1_Score(pred_rf,test_data$Class,positive = NULL)
########################################
```

That values changing has minor effect on F1 score, AUC, also accuracy and sensitivity, specifity on confusion matrix. Seen that the first scenario implemented values gives us better performance for Random Forest.

```
          Reference
Prediction    0    1
         0 9931   39
         1    1   29

               Accuracy : 0.996
                 95% CI : (0.9946, 0.9971)
    No Information Rate : 0.9932
    P-Value [Acc > NIR] : 0.0001604

                  Kappa : 0.5901

 Mcnemar's Test P-Value : 4.909e-09

            Sensitivity : 0.4265
            Specificity : 0.9999
         Pos Pred Value : 0.9667
         Neg Pred Value : 0.9961
              Precision : 0.9667
                 Recall : 0.4265
                     F1 : 0.5918
             Prevalence : 0.0068
         Detection Rate : 0.0029
   Detection Prevalence : 0.0030
      Balanced Accuracy : 0.7132
```

```
> roc.curve(test_data$Class,pred_rf, plotit = TRUE)
Area under the curve (AUC): 0.981
> F1_Score(pred_rf,test_data$Class,positive = NULL)
[1] 0.9979902
```

Screenshot-Confusion matrix, F1 Score, AUC results for Scenario 2

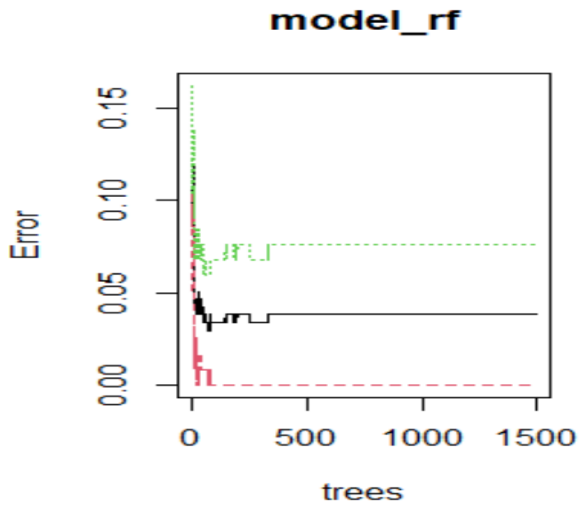


Figure- Random Forest errors vs tree number relation

In this above random forest graph due to trees changed as 1500 in code graphs boundary changes parallel with that number. Error intervals same. It is seen clearly, some trees boundary in between 0-500 gets error after that interval there is no errors for red lines and black lines & green lines error point become a stabile.

```
> varImpPlot(model_rf)
>
```

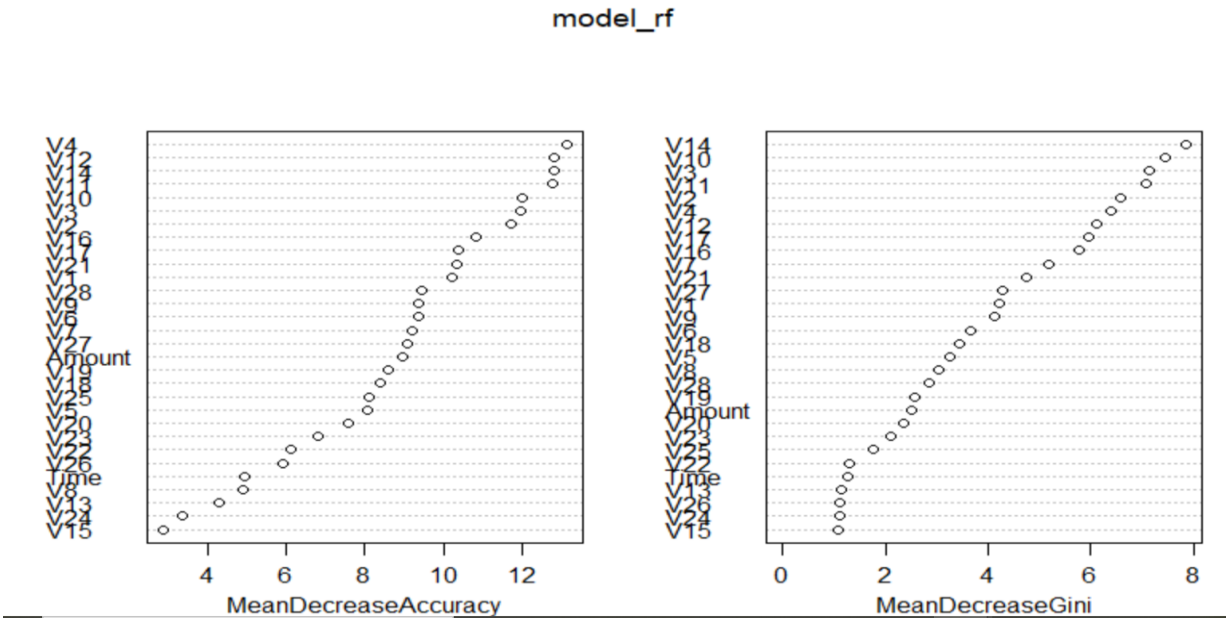


Figure -Random Forest Graph for Random Under Sampling Method

So, a random forest is giving quite accurate results for negative predictions. But ,also it is to be noted that a good number of cases of nonfraud were reported as fraud which will create problems for the organizations as mentioned above. These problems might have been solved if we knew what the real variables actually were. We can consider implemented condition in **Scenario-1** for random forest algorithm.

5- Evaluation

We concluded that the undersampling technique and smote techniques work best on the dataset and achieved significant improvement in model performance over the imbalanced data.

The best score of 0.9962 was achieved using a Random Forest Model though both Decision tree models performed well too.

KNN model also has high accuracy but the AUC value is not good enough than Decision Tree and Random Forest Model. In addition to that, F1 Scores also pretty high for these models.

During the data analysis seen that changing tree numbers, neuron numbers in the model also have some effects on the model.

The most interesting model in this project was working with the ANN model.

ANN model does not show good performance for Train 80% and Test 20% split ratio. It is seen that terrible accuracy. This ratio is not enough to give us good results. That's why for ANN model gets good results with Train 65%-Test 35% split ratio. ANN models AUC result also good but it is not excellent like Decision Tree and Random Forest.

You can check the performance evaluation of each model and used sampling methods, Test - Train Ratio Rates via below given table.

Machine Learning Methods	Accuracy	F1- Score	AUC	Sampling Method Technique	Test-Train Split Ratio
ANN	0.9671	0.9832552	0.897	Random Under Sampling	65% Train- %35 Test
Decision Tree	0.9844	0.9921172	0.942	Smote	80% Train- 20% Test
KNN	0.9937	0.9968331	0.864	Random Under Sampling	80% Train- 20% Test
Random Forest	0.9962	0.9980405	0.981	Random Under Sampling	80% Train- 20% Test

6- FINAL

In this project, I tried to show different methods of dealing with unbalanced datasets like the fraud credit card transaction dataset where the instances of fraudulent cases are few compared to the instances of normal transactions.

On the other hand, argued due to working with unbalanced data, just accuracy is not an appropriate measure of model performance here and used the metric AREA UNDER ROC CURVE and F1 Score to evaluate how different methods of oversampling, undersampling, smote the response variable can lead to better model training.

In addition to that seen that some sampling methods such as Random Over Sampling method and Random Under Sampling & Random Over Sampling together usage method cause performance problem on my computer. That's why working with large data set; working computer performance properties also should consider at the beginning of the project.

This project has demonstrated the importance of sampling methods and the positive effects of modeling using that technique on the imbalanced datasets. On the other hand, used many different data visualization graphs and plots to be able to understand results well and explicate easily result.

I believe at that point, it is very important to make a regression analysis, and finding the correct machine learning strategy in identifying fraud in the early stages will prevent businesses largest losses.

7- References

[1] Awoyemi, John O., et al. “Credit Card Fraud Detection Using Machine Learning Techniques: A Comparative Analysis.” 2017 International Conference on Computing Networking and Informatics (ICCNI), 2017, doi:10.1109/iccni.2017.8123782.

[2] Randhawa, Kuldeep, et al. “Credit Card Fraud Detection Using AdaBoost and Majority Voting.” IEEE Access, vol. 6, 2018, pp. 14277–14284., doi:10.1109/access.2018.2806420.

[3] Melo-Acosta, German E., et al. “Fraud Detection in Big Data Using Supervised and Semi-Supervised Learning Techniques.” 2017 IEEE Colombian Conference on Communications and Computing (COLCOM), 2017, doi:10.1109/colcomcon.2017.8088206.

[4] http://www.rbi.org.in/Circular/CreditCard

[5] https://www.kaggle.com/mlg-ulb/creditcardfraud