Credit Card Fraud Detection

Rushit Bhadaniya

7/8/2020

Install & Load Library ——————————————————

```
#install.packages("caret")
#install.packages("ggcorrplot")
#install.packages("ROSE")
#install.packages("smotefamily")
#install.packages("rpart.plot")
#install.packages("e1071")
library(caret)

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

## Loading required package: ggplot2

library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(ggplot2)
library(caTools)
library(ggcorrplot)

## Warning: package 'ggcorrplot' was built under R version 3.6.3

library(ROSE)

## Warning: package 'ROSE' was built under R version 3.6.3

## Loaded ROSE 0.0-3

library(smotefamily)

## Warning: package 'smotefamily' was built under R version 3.6.3
```

```r
library(rpart)
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.6.3

library(e1071)

## Warning: package 'e1071' was built under R version 3.6.3
```

Import Dataset ————————————————————————

```r
credit_card<-
read.csv('D:\\M.Tech\\Credit_Card_fraud_detection\\creditcard_dataset.csv')
```

Analyze Dataset ————————————————————————

```r
#str(dataset) helps in understanding the structure of the data set, data type
of each attribute and number of rows and columns present in the data.
str(credit_card)

## '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 ...
```

```r
credit_card$Class = factor(credit_card$Class, levels = c(0,1))
#Summary() is one of the most important functions that help in summarising each attribute in the dataset. It gives a set of descriptive statistics, depending on the type of variable.
summary(credit_card)
```

```
##       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          Class
##  Min.   :-22.565679   Min.   :-15.43008   Min.   :    0.00   0:284315
##  1st Qu.: -0.070840   1st Qu.: -0.05296   1st Qu.:    5.60   1:   492
##  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
```

```r
#check for missing values
sum(is.na(credit_card))
```

```
## [1] 0
```

```r
#visualisation for credit card transaction...
table(credit_card$Class)
```

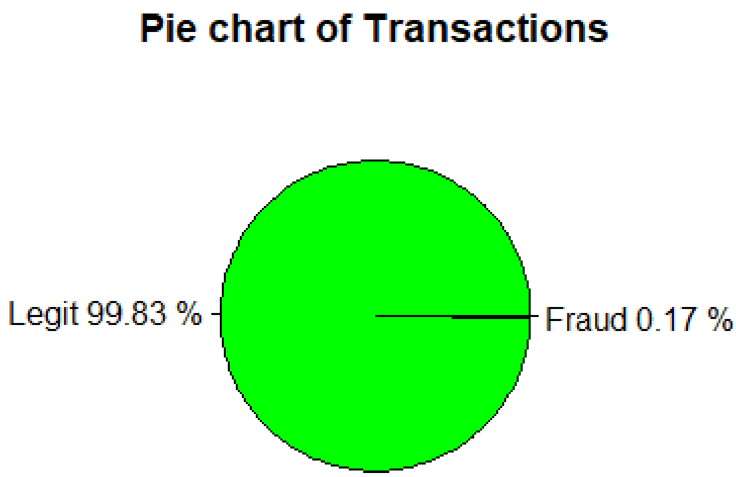
```
##
##      0      1
## 284315    492
```

```r
prop.table(table(credit_card$Class))
```

```
##
##           0           1
## 0.998272514 0.001727486
```

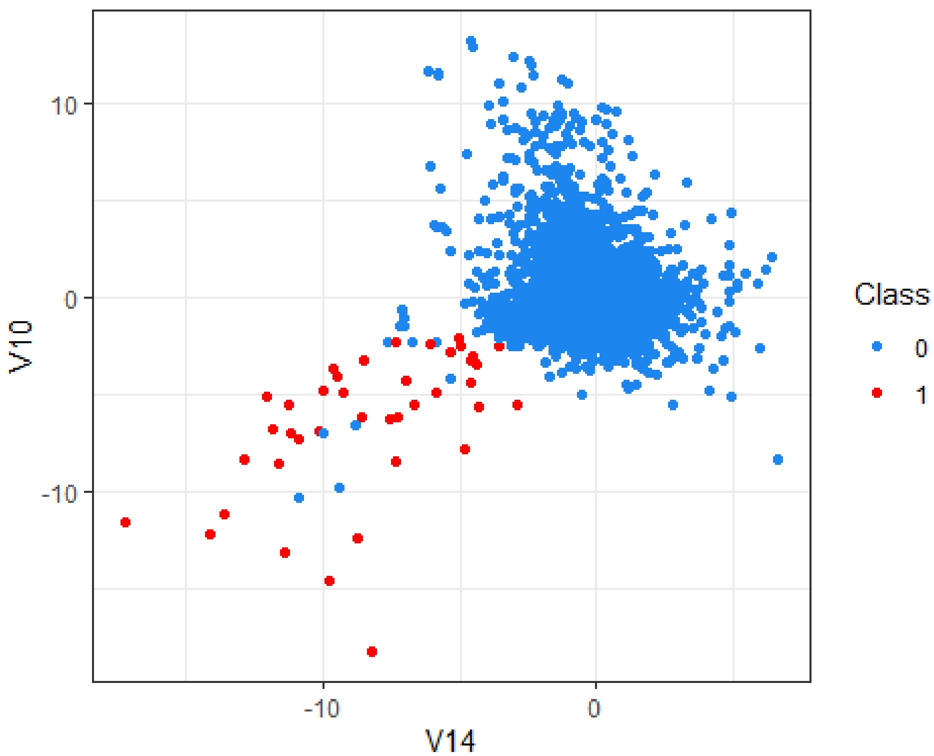
```r
#pie chart for credit card transaction
labels<-c("Legit","Fraud")
labels<-paste(labels,round(100*prop.table(table(credit_card$Class)),2))
labels<-paste(labels,"%")
```

```r
pie(table(credit_card$Class),labels,col = c("green","red"),
                                main = "Pie chart of Transactions" )
```



Comments: Here,
We can see that Data is unbalanced. so, we have to handle this unbalanced data.

```r
set.seed(1)
#here, we use smaple_frac() function to get a smaller fraction from our
dataset.
#here, we are taking small fraction of our dataset to train and build the
mode.so that it is computationally faster.
credit_card <- credit_card %>%sample_frac(0.1)
table(credit_card$Class)

##
##     0     1
## 28437    44

ggplot(data=credit_card,aes(x=V14 ,y=V10,col= Class))+
  geom_point()+
  theme_bw()+
  scale_color_manual(values = c('dodgerblue2','red'))
```



Splitting data into Training and Test data ——————————————

```
#here, we use caTool library to slpit our data

set.seed(123)
data_sample = sample.split(credit_card$Class,SplitRatio = 0.80)

train_data= subset(credit_card,data_sample== TRUE)
test_data= subset(credit_card,data_sample== FALSE)

dim(train_data)

## [1] 22785    31

dim(test_data)

## [1] 5696   31
```

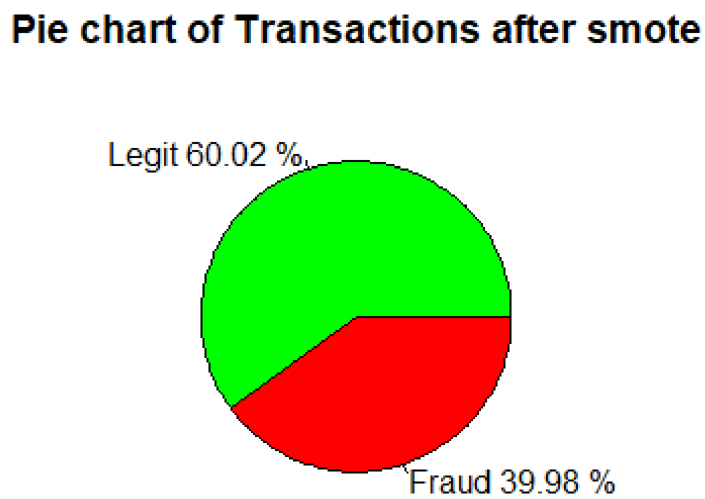
Balance the Imbalanced dataset ————————————————————

- There are various methods available to handle unbalanced data. But here, efficient method is SMOTE method.
- Commented code is for trying different method to see the changes in dataset

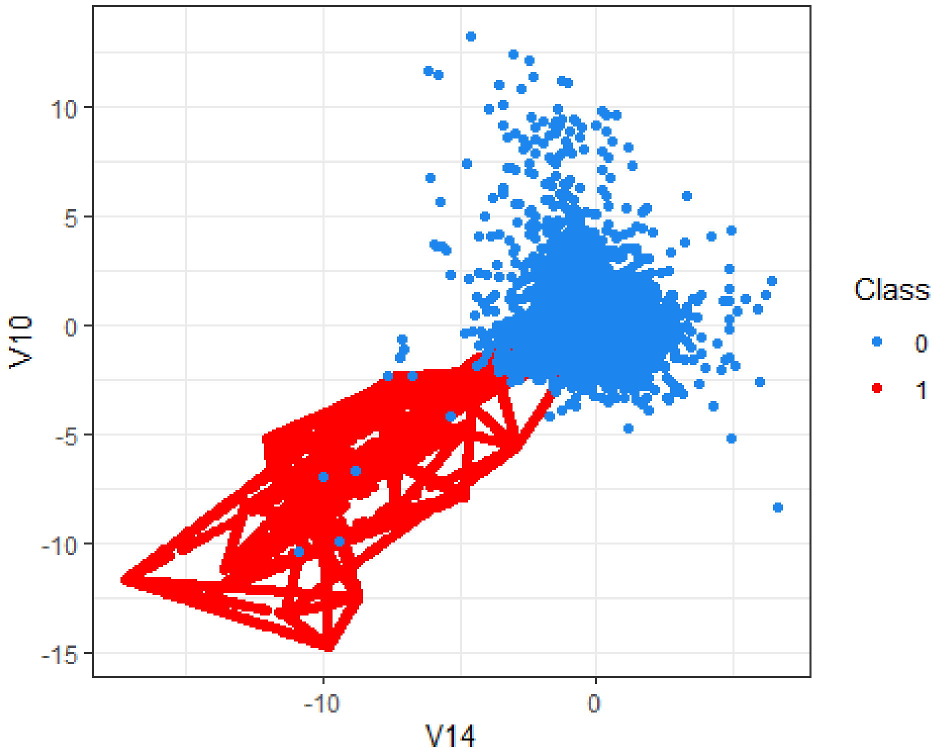
```
# #1. Random over-sampling(Ros)
# table(train_data$Class)
# n_legit<-22750
# new_frac_legit <-0.50 #so,after oversampling class 0 and 1 will be 50-50%
in our data set.
```

```
# new_no_total<- n_legit/new_frac_legit
# #we use ROSE library to for balancing our dataset
#
# oversampling_result <- ovun.sample(class ~ .,
#                                    data=train_data,
#                                    method = "over",
#                                    N=new_no_total,
#                                    Seed=2020)
# oversampling_credit <- oversampling_result
# table(oversampling_credit)
# # this method will copy and paste class 1 data rendomly to oversample the
dataset.
# #So, this method endup creating duplicate data.
#
# #2.Random Under-sampling (RUS)
# table(train_data$Class)
# n_fraud<-35
# new_frac_fraud <-0.50 #so,after under-sampling class 0 and 1 will be 50-50%
in our data set.
# new_no_total<- n_fraud/new_frac_fraud
#
# undersampling_result <- ovun.sample(class ~ .,
#                                    data=train_data,
#                                    method = "under",
#                                    N=new_no_total,
#                                    Seed=2020)
# undersampling_credit <- undersampling_result
# table(undersampling_credit)
# #This method will decrease the number of legitimate cases.
# #Problem here, is that we endup loosing lots of data.
#
# #3.Use Both Method
# table(train_data$Class)
# n_new<-nrow(train_data) #=22785
# new_fraction_fraud <-0.50 #so,after under-sampling class 0 and 1 will be
50-50% in our data set.
#
# sampling_result <- ovun.sample(class ~ .,
#                                     data=train_data,
#                                     method = "both",
#                                     N=n_new,
#                                     p= new_fraction_fraud,
#                                     Seed=2020)
#
# sampling_credit <- sampling_result$data
# table(sampling_credit)

#4. SMOTE method for unbalanced data
#it will generate a new "SMOTEd" data set that addresses the class unbalance
```

```r
problem.
table(train_data$Class)

##
##     0     1
## 22750    35

n_legit<-22750
n_fraud<-35
wanted_ratio<-0.6 #so,after smote 60% will be legit transaction and 40% will
be fraud transaction
#calculate the value for the dup_size parameter of SMOTE
ntimes <- ((1- wanted_ratio)/wanted_ratio)*(n_legit/n_fraud)-1

smote_output = SMOTE(X= train_data[ ,-c(1,31)],
                     target = train_data$Class,
                     K=5,
                     dup_size = ntimes)

credit_smote <- smote_output$data

colnames(credit_smote)[30]<-"Class"
# see the distribution of class column
prop.table(table(credit_smote$Class))

##
##         0         1
## 0.6001847 0.3998153

#pie chart
labels<-c("Legit","Fraud")
labels<-paste(labels,round(100*prop.table(table(credit_smote$Class)),2))
labels<-paste(labels,"%")
pie(table(credit_smote$Class),labels,col = c("green","red"),
    main = "Pie chart of Transactions after smote" )
```

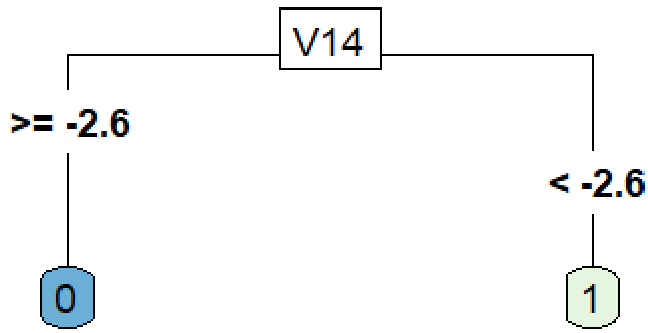



```r
#scatter plot
ggplot(data=credit_smote
       ,aes(x=V14 ,y=V10,col= Class))+
  geom_point()+
  theme_bw()+
  scale_color_manual(values = c('dodgerblue2','red'))
```



Build Decision Tree ———————————————————————-

```
#we use Rpart library to build decision tree

CART_model = rpart(Class ~ .,credit_smote)

rpart.plot(CART_model,extra = 0,type = 5,tweak = 1.2)
```

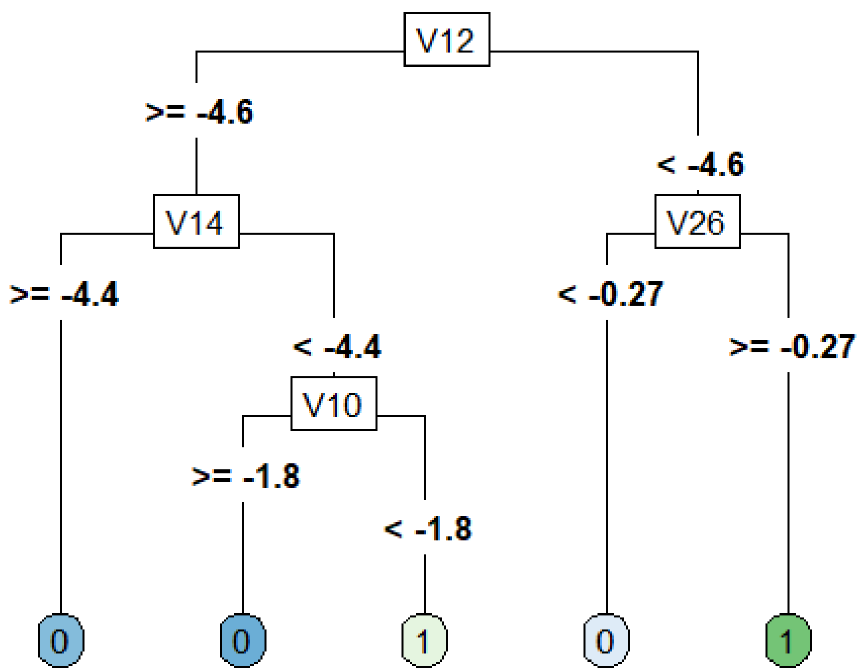



```
#Predict fraud classes
predicted_val <-predict(CART_model,test_data,type = 'class')

#build confusion matrix
confusionMatrix(predicted_val,test_data$Class)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5622    2
##          1   65    7
##
##                Accuracy : 0.9882
##                  95% CI : (0.9851, 0.9909)
##     No Information Rate : 0.9984
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1705
##
##  Mcnemar's Test P-Value : 3.605e-14
##
##             Sensitivity : 0.98857
##             Specificity : 0.77778
##          Pos Pred Value : 0.99964
##          Neg Pred Value : 0.09722
##              Prevalence : 0.99842
```

```
##          Detection Rate : 0.98701
##    Detection Prevalence : 0.98736
##       Balanced Accuracy : 0.88317
##
##        'Positive' Class : 0
##
```

```r
#Let's build same model using original train data--------------------------
CART_model = rpart(Class ~ .,train_data[,-1])

rpart.plot(CART_model,extra = 0,type = 5,tweak = 1.2)
```

```
## Warning: Bad 'data' field in model 'call' (expected a data.frame or a
matrix).
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```




```r
#Predict fraud classes
predicted_val <-predict(CART_model,test_data[-1],type = 'class')

#build confusion matrix
confusionMatrix(predicted_val,test_data$Class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction    0    1
##          0 5686    3
##          1    1    6
##
##                Accuracy : 0.9993
##                  95% CI : (0.9982, 0.9998)
##     No Information Rate : 0.9984
##     P-Value [Acc > NIR] : 0.05483
##
##                   Kappa : 0.7497
##
##  Mcnemar's Test P-Value : 0.61708
##
##             Sensitivity : 0.9998
##             Specificity : 0.6667
##          Pos Pred Value : 0.9995
##          Neg Pred Value : 0.8571
##              Prevalence : 0.9984
##          Detection Rate : 0.9982
##    Detection Prevalence : 0.9988
##       Balanced Accuracy : 0.8332
##
##        'Positive' Class : 0
##
```

```
#Check Accuracy on whole somte data ------------------------------------
CART_model = rpart(Class ~ .,credit_smote)
predicted_val <-predict(CART_model,credit_card[-1],type = 'class')
confusionMatrix(predicted_val,credit_card$Class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 28141     4
##          1   296    40
##
##                Accuracy : 0.9895
##                  95% CI : (0.9882, 0.9906)
##     No Information Rate : 0.9985
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2084
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9896
##             Specificity : 0.9091
##          Pos Pred Value : 0.9999
##          Neg Pred Value : 0.1190
```

```
##               Prevalence : 0.9985
##           Detection Rate : 0.9881
##     Detection Prevalence : 0.9882
##        Balanced Accuracy : 0.9493
##
##         'Positive' Class : 0
##
```

```
# Check Accuracy on whole unbalanced data ---------------------------------
CART_model = rpart(Class ~ .,train_data[,-1])
predicted_val <-predict(CART_model,credit_card[-1],type = 'class')
confusionMatrix(predicted_val,credit_card$Class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 28433     9
##          1     4    35
##
##                Accuracy : 0.9995
##                  95% CI : (0.9992, 0.9998)
##     No Information Rate : 0.9985
##     P-Value [Acc > NIR] : 3.99e-08
##
##                   Kappa : 0.8431
##
##  Mcnemar's Test P-Value : 0.2673
##
##             Sensitivity : 0.9999
##             Specificity : 0.7955
##          Pos Pred Value : 0.9997
##          Neg Pred Value : 0.8974
##              Prevalence : 0.9985
##          Detection Rate : 0.9983
##    Detection Prevalence : 0.9986
##       Balanced Accuracy : 0.8977
##
##        'Positive' Class : 0
##
```

Comments: Here, We can see that we have improved our accuracy by using SMOTE technique.