1. load data in R
2. Data Exploration ## data preview

Frauddata$type = as.character(Frauddata$type)  
Frauddata$nameDest = as.character(Frauddata$nameDest)  
Frauddata$nameOrig = as.numeric(Frauddata$nameOrig)  
  
head(Frauddata)

## step type amount nameOrig oldbalanceOrg newbalanceOrig nameDest  
## 1 1 PAYMENT 9839.64 757870 170136 160296.36 M1979787155  
## 2 1 PAYMENT 1864.28 2188999 21249 19384.72 M2044282225  
## 3 1 TRANSFER 181.00 1002157 181 0.00 C553264065  
## 4 1 CASH\_OUT 181.00 5828263 181 0.00 C38997010  
## 5 1 PAYMENT 11668.14 3445982 41554 29885.86 M1230701703  
## 6 1 PAYMENT 7817.71 6026526 53860 46042.29 M573487274  
## oldbalanceDest newbalanceDest isFraud isFlaggedFraud  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 1 0  
## 4 21182 0 1 0  
## 5 0 0 0 0  
## 6 0 0 0 0

str(Frauddata)

## 'data.frame': 6362620 obs. of 11 variables:  
## $ step : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ type : chr "PAYMENT" "PAYMENT" "TRANSFER" "CASH\_OUT" ...  
## $ amount : num 9840 1864 181 181 11668 ...  
## $ nameOrig : num 757870 2188999 1002157 5828263 3445982 ...  
## $ oldbalanceOrg : num 170136 21249 181 181 41554 ...  
## $ newbalanceOrig: num 160296 19385 0 0 29886 ...  
## $ nameDest : chr "M1979787155" "M2044282225" "C553264065" "C38997010" ...  
## $ oldbalanceDest: num 0 0 0 21182 0 ...  
## $ newbalanceDest: num 0 0 0 0 0 ...  
## $ isFraud : int 0 0 1 1 0 0 0 0 0 0 ...  
## $ isFlaggedFraud: int 0 0 0 0 0 0 0 0 0 0 ...

## check missing value

sum(is.na(Frauddata$step))

## [1] 0

sum(is.na(Frauddata$type))

## [1] 0

sum(is.na(Frauddata$amount))

## [1] 0

sum(is.na(Frauddata$nameOrig))

## [1] 0

sum(is.na(Frauddata$oldbalanceOrg))

## [1] 0

sum(is.na(Frauddata$newbalanceOrig))

## [1] 0

sum(is.na(Frauddata$nameDest))

## [1] 0

sum(is.na(Frauddata$oldbalanceDest))

## [1] 0

sum(is.na(Frauddata$newbalanceDest))

## [1] 0

sum(is.na(Frauddata$isFraud))

## [1] 0

sum(is.na(Frauddata$isFlaggedFraud))

## [1] 0

## how many fraudulent/non-fraudulent transactions

length(Frauddata$isFraud[Frauddata$isFraud == 1])

## [1] 8213

length(Frauddata$isFraud[Frauddata$isFraud == 0])

## [1] 6354407

## Check how many distinct items in name

length(unique(Frauddata$nameOrig))

## [1] 6353307

length(unique(Frauddata$nameDest))

s

## check how many rows flageed as fraud

length(Frauddata$isFlaggedFraud[Frauddata$isFlaggedFraud == 1])

## [1] 16

## test normality

library(nortest)  
ad.test(Frauddata$amount)

##   
## Anderson-Darling normality test  
##   
## data: Frauddata$amount  
## A = 1307700, p-value < 2.2e-16

ad.test(Frauddata$newbalanceDest)

##   
## Anderson-Darling normality test  
##   
## data: Frauddata$newbalanceDest  
## A = 1197600, p-value < 2.2e-16

ad.test(Frauddata$newbalanceOrig)

##   
## Anderson-Darling normality test  
##   
## data: Frauddata$newbalanceOrig  
## A = 1692700, p-value < 2.2e-16

ad.test(Frauddata$oldbalanceDest)

##   
## Anderson-Darling normality test  
##   
## data: Frauddata$oldbalanceDest  
## A = 1234300, p-value < 2.2e-16

ad.test(Frauddata$oldbalanceOrg)

##   
## Anderson-Darling normality test  
##   
## data: Frauddata$oldbalanceOrg  
## A = 1711600, p-value < 2.2e-16

##amount,oldbalanceOrg,newbalanceOrig,newbalanceDest,oldbalanceDest are not normally distributed

## summary of data

summary(Frauddata)

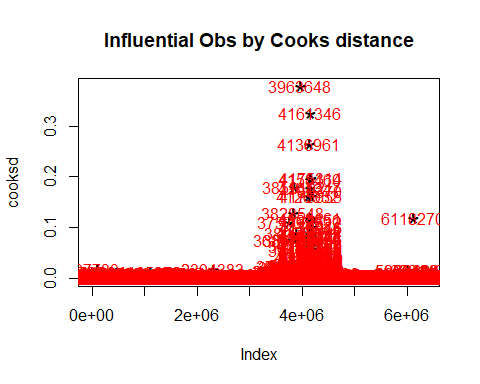
## step type amount nameOrig   
## Min. : 1.0 Length:6362620 Min. : 0 Min. : 1   
## 1st Qu.:156.0 Class :character 1st Qu.: 13390 1st Qu.:1588333   
## Median :239.0 Mode :character Median : 74872 Median :3176674   
## Mean :243.4 Mean : 179862 Mean :3176679   
## 3rd Qu.:335.0 3rd Qu.: 208721 3rd Qu.:4765049   
## Max. :743.0 Max. :92445517 Max. :6353307   
## oldbalanceOrg newbalanceOrig nameDest   
## Min. : 0 Min. : 0 Length:6362620   
## 1st Qu.: 0 1st Qu.: 0 Class :character   
## Median : 14208 Median : 0 Mode :character   
## Mean : 833883 Mean : 855114   
## 3rd Qu.: 107315 3rd Qu.: 144258   
## Max. :59585040 Max. :49585040   
## oldbalanceDest newbalanceDest isFraud   
## Min. : 0 Min. : 0 Min. :0.000000   
## 1st Qu.: 0 1st Qu.: 0 1st Qu.:0.000000   
## Median : 132706 Median : 214661 Median :0.000000   
## Mean : 1100702 Mean : 1224996 Mean :0.001291   
## 3rd Qu.: 943037 3rd Qu.: 1111909 3rd Qu.:0.000000   
## Max. :356015889 Max. :356179279 Max. :1.000000   
## isFlaggedFraud   
## Min. :0.0e+00   
## 1st Qu.:0.0e+00   
## Median :0.0e+00   
## Mean :2.5e-06   
## 3rd Qu.:0.0e+00   
## Max. :1.0e+00

## check outliers

## cook’s distance

## multivariate model (ratio is 0.000001729)

cooksd = cooks.distance(lm(isFraud~Frauddata$amount+Frauddata$oldbalanceOrg+Frauddata$newbalanceOrig+Frauddata$oldbalanceDest+Frauddata$newbalanceDest, data = Frauddata))  
  
plot(cooksd, pch="\*", cex=2, main="Influential Obs by Cooks distance")  
abline(h = 4\*mean(cooksd, na.rm=T), col="red")   
text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>4\*mean(cooksd, na.rm=T),names(cooksd),""), col="red")



influential <- as.numeric(names(cooksd)[(cooksd > 4\*mean(cooksd, na.rm=T))])  
head(Frauddata[influential, ])

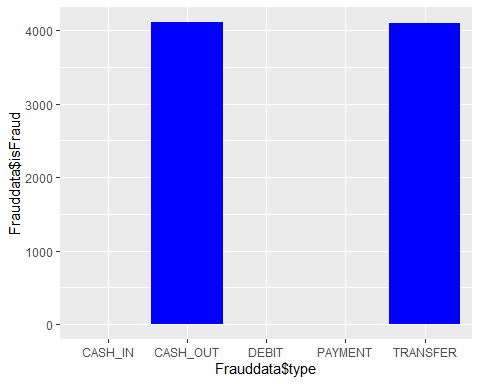
## step type amount nameOrig oldbalanceOrg newbalanceOrig  
## 3 1 TRANSFER 181.00 1002157 181 0  
## 4 1 CASH\_OUT 181.00 5828263 181 0  
## 49 1 CASH\_OUT 5346.89 4749572 0 0  
## 85 1 TRANSFER 379856.23 1477100 0 0  
## 89 1 TRANSFER 761507.39 4421347 0 0  
## 90 1 TRANSFER 1429051.47 1708443 0 0  
## nameDest oldbalanceDest newbalanceDest isFraud isFlaggedFraud  
## 3 C553264065 0 0 1 0  
## 4 C38997010 21182 0 1 0  
## 49 C248609774 652637 6453431 0 0  
## 85 C1590550415 900180 19169205 0 0  
## 89 C1590550415 1280036 19169205 0 0  
## 90 C1590550415 2041544 19169205 0 0

## Histograms

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.5.3

ggplot(Frauddata,aes(x=Frauddata$type, y =Frauddata$isFraud))+geom\_bar(stat = "identity", fill = "blue")



length(Frauddata$type[Frauddata$type=="CASH\_OUT"])

## [1] 2237500

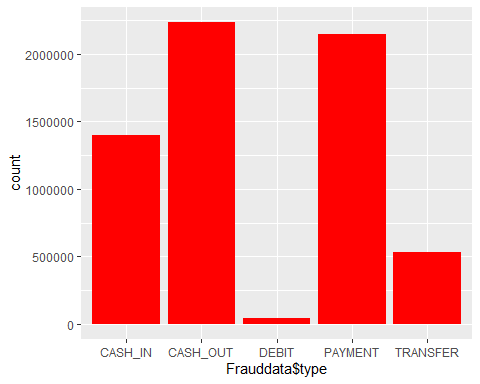
length(Frauddata$type[Frauddata$type== "TRANSFER"])

## [1] 532909

unique(Frauddata$type)

## [1] "PAYMENT" "TRANSFER" "CASH\_OUT" "DEBIT" "CASH\_IN"

ggplot(Frauddata,aes(Frauddata$type))+geom\_bar(fill = "red")



## correlation(spearman non para since numeric variables are not normal)

cor(data.frame(Frauddata$amount,Frauddata$oldbalanceOrg,Frauddata$newbalanceOrig, Frauddata$oldbalanceDest,Frauddata$newbalanceDest))

## Frauddata.amount Frauddata.oldbalanceOrg  
## Frauddata.amount 1.000000000 -0.002762475  
## Frauddata.oldbalanceOrg -0.002762475 1.000000000  
## Frauddata.newbalanceOrig -0.007860925 0.998802763  
## Frauddata.oldbalanceDest 0.294137450 0.066242501  
## Frauddata.newbalanceDest 0.459304267 0.042028619  
## Frauddata.newbalanceOrig Frauddata.oldbalanceDest  
## Frauddata.amount -0.007860925 0.29413745  
## Frauddata.oldbalanceOrg 0.998802763 0.06624250  
## Frauddata.newbalanceOrig 1.000000000 0.06781152  
## Frauddata.oldbalanceDest 0.067811518 1.00000000  
## Frauddata.newbalanceDest 0.041837497 0.97656851  
## Frauddata.newbalanceDest  
## Frauddata.amount 0.45930427  
## Frauddata.oldbalanceOrg 0.04202862  
## Frauddata.newbalanceOrig 0.04183750  
## Frauddata.oldbalanceDest 0.97656851  
## Frauddata.newbalanceDest 1.00000000

* **Logistic regression (information Gain)**

Frauddata$type = as.character(Frauddata$type)  
Frauddata$nameDest = as.character(Frauddata$nameDest)  
Frauddata$nameOrig = as.numeric(Frauddata$nameOrig)

start\_time <- Sys.time()  
  
library(caret)

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.5.3

library(FSelectorRcpp)

## Warning: package 'FSelectorRcpp' was built under R version 3.5.3

library(FSelector)

## Warning: package 'FSelector' was built under R version 3.5.3

library(ROSE)

## Warning: package 'ROSE' was built under R version 3.5.3

## Loaded ROSE 0.0-3

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.5.3

##   
## 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(Metrics)

## Warning: package 'Metrics' was built under R version 3.5.3

##   
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':  
##   
## precision, recall

library(cvAUC)

## Warning: package 'cvAUC' was built under R version 3.5.3

## Loading required package: ROCR

## Warning: package 'ROCR' was built under R version 3.5.3

## Loading required package: gplots

## Warning: package 'gplots' was built under R version 3.5.3

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

## Loading required package: data.table

## Warning: package 'data.table' was built under R version 3.5.3

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

##

## cvAUC version: 1.1.0

## Notice to cvAUC users: Major speed improvements in version 1.1.0

##

validateaccruacy=list()  
testaccruacy = list()  
validaterecall = list()  
testrecall = list()  
validateprecision = list()  
testreprecision = list()  
validateF1 = list()  
testF1 = list()  
MSErrorvalidate = list()  
MSErrortest = list()  
aucvalidate = list()  
auctest = list()  
  
folds = cut(seq(1,nrow(Frauddata)),breaks = 10, labels = F)  
for(f in 1:10){  
 ##Random Shuffle to avoid overfitting  
 Frauddata = Frauddata[sample(nrow(Frauddata)),]  
 Frauddata = Frauddata[sample(nrow(Frauddata)),]  
 Frauddata = Frauddata[sample(nrow(Frauddata)),]  
 Frauddata = Frauddata[sample(nrow(Frauddata)),]  
 Frauddata = Frauddata[sample(nrow(Frauddata)),]  
 ## spliting data  
   
 testIndexes = which(folds == f, arr.ind =T)  
 train = Frauddata[-testIndexes,]  
 testset = Frauddata[testIndexes,]  
 data\_train = sample(nrow(train), floor(nrow(train)\*0.8))  
 trainset = train[data\_train,]  
 validateset = train[-data\_train,]  
 ## performing information gain feature selection  
 x = information\_gain(isFraud ~ ., trainset)  
 to\_formula(cut\_attrs(attrs = x), "Species")  
 variable = cut\_attrs(attrs = x, k = 0.7)  
 variable = c(variable[-1],"isFraud")  
 trainset = trainset[variable]  
   
 ##Over and under sampling  
 trainsetunder = data.frame(ovun.sample(isFraud~.,data = trainset, method = "under", seed =1)$data)  
 trainsetboth = data.frame(ovun.sample(isFraud~.,data = trainset, method = "both", seed =1)$data)  
 trainsetunder$isFraud = as.factor(trainsetunder$isFraud)  
 trainsetboth$isFraud = as.factor(trainsetboth$isFraud)  
   
 ##logstic regression   
   
  
 logisticboth = glm(formula = isFraud~.,family = "binomial",data = trainsetboth)  
 probabilities <- logisticboth %>% predict(validateset, type = "response")  
 predicted.validate = ifelse(probabilities>0.5,1,0)  
 cmvalidate = as.matrix(table(Actual = validateset$isFraud, Predicted =predicted.validate ))  
 cmvalidate  
   
 n1 = sum(cmvalidate)  
 diag1 = diag(cmvalidate)  
 accuracyvalidate = sum(diag1) / n1   
 precisionvalidate = (cmvalidate[2,2])/(cmvalidate[2,2]+cmvalidate[1,2])  
 recallvalidate = (cmvalidate[2,2])/(cmvalidate[2,2]+cmvalidate[2,1])  
 recallvalidate  
 f1validate = 2 \* precisionvalidate \* recallvalidate / (precisionvalidate + recallvalidate)   
 data.frame(precisionvalidate, recallvalidate, f1validate)  
 auc1 = AUC(predicted.validate,validateset$isFraud )  
 ##store result  
   
 validateaccruacy= append(validateaccruacy,accuracyvalidate)  
 validaterecall= append(validaterecall,recallvalidate)  
 validateprecision = append(validateprecision,precisionvalidate)  
 validateF1= append(validateF1,f1validate)  
 MSErrorvalidate = append(MSErrorvalidate,mse(actual = validateset$isFraud, predict = predicted.validate))  
 aucvalidate = append(aucvalidate, auc1)  
   
  
 ##  
 probabilitiestest <- logisticboth %>% predict(testset, type = "response")  
 predicted.test = ifelse(probabilitiestest>0.5,1,0)  
 mean(predicted.test == testset$isFraud)  
 cm = as.matrix(table(Actual = testset$isFraud, Predicted =predicted.test ))  
  
 n = sum(cm)  
 nc = nrow(cm)  
 diag = diag(cm)  
 accuracy = sum(diag) / n   
 precision = (cm[2,2])/(cm[2,2]+cm[1,2])  
 recall = (cm[2,2])/(cm[2,2]+cm[2,1])  
 f1 = 2 \* precision \* recall / (precision + recall)   
 data.frame(precision, recall, f1)  
 auc2 = AUC(predicted.test,testset$isFraud )  
   
 ##store result  
   
 testaccruacy= append(testaccruacy,accuracy)  
 testrecall=append(testrecall,recall)  
 testreprecision=append(testreprecision,precision)  
 testF1= append(testF1,f1)  
 MSErrortest = append(MSErrortest,mse(actual = testset$isFraud, predict = predicted.test) )  
 auctest = append(auctest, auc2)  
   
 }

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 1.296812 hours

##Validate Result  
 mean(as.numeric(validateaccruacy))

## [1] 0.8061388

mean(as.numeric(validaterecall))

## [1] 0.7683645

mean(as.numeric(validateF1))

## [1] 0.04843643

mean(as.numeric(MSErrorvalidate))

## [1] 0.1938612

mean(as.numeric(aucvalidate))

## [1] 0.7872763

matrix(validateaccruacy)

## [,1]   
## [1,] 0.7040633  
## [2,] 0.9881897  
## [3,] 0.6858711  
## [4,] 0.9874266  
## [5,] 0.6510226  
## [6,] 0.9853144  
## [7,] 0.7024698  
## [8,] 0.9836912  
## [9,] 0.682905   
## [10,] 0.6904342

matrix(validaterecall)

## [,1]   
## [1,] 0.9967742  
## [2,] 0.6961057  
## [3,] 0.9973369  
## [4,] 0.6803559  
## [5,] 0.0244373  
## [6,] 0.650904   
## [7,] 0.9986329  
## [8,] 0.6452933  
## [9,] 0.9958735  
## [10,] 0.997931

matrix(validateF1)

## [,1]   
## [1,] 0.009034612   
## [2,] 0.1289284   
## [3,] 0.008258927   
## [4,] 0.1213083   
## [5,] 0.0001901188  
## [6,] 0.1001552   
## [7,] 0.008502219   
## [8,] 0.09197861   
## [9,] 0.007911358   
## [10,] 0.008096668

matrix(MSErrorvalidate)

## [,1]   
## [1,] 0.2959367   
## [2,] 0.01181029  
## [3,] 0.3141289   
## [4,] 0.01257343  
## [5,] 0.3489774   
## [6,] 0.01468559  
## [7,] 0.2975302   
## [8,] 0.01630879  
## [9,] 0.317095   
## [10,] 0.3095658

matrix(aucvalidate)

## [,1]   
## [1,] 0.8502204  
## [2,] 0.8423313  
## [3,] 0.8413995  
## [4,] 0.8340874  
## [5,] 0.3381559  
## [6,] 0.8183194  
## [7,] 0.850362   
## [8,] 0.8147091  
## [9,] 0.8391903  
## [10,] 0.8439877

##test Result  
 mean(as.numeric(testaccruacy))

## [1] 0.806337

mean(as.numeric(testrecall))

## [1] 0.7640862

mean(as.numeric(testF1))

## [1] 0.049051

mean(as.numeric(MSErrortest))

## [1] 0.193663

mean(as.numeric(auctest))

## [1] 0.7852362

matrix(testaccruacy)

## [,1]   
## [1,] 0.7037007  
## [2,] 0.988456   
## [3,] 0.6861214  
## [4,] 0.9873244  
## [5,] 0.6516341  
## [6,] 0.9850423  
## [7,] 0.7035325  
## [8,] 0.9837866  
## [9,] 0.6834244  
## [10,] 0.6903477

matrix(testrecall)

## [,1]   
## [1,] 0.9976247   
## [2,] 0.6821026   
## [3,] 0.995116   
## [4,] 0.6743064   
## [5,] 0.02405063  
## [6,] 0.652019   
## [7,] 0.9952607   
## [8,] 0.6310795   
## [9,] 0.9964455   
## [10,] 0.9928571

matrix(testF1)

## [,1]   
## [1,] 0.008832622   
## [2,] 0.1292235   
## [3,] 0.008095799   
## [4,] 0.1217467   
## [5,] 0.0001714105  
## [6,] 0.1034385   
## [7,] 0.008827656   
## [8,] 0.09349736   
## [9,] 0.008281349   
## [10,] 0.008395072

matrix(MSErrortest)

## [,1]   
## [1,] 0.2962993   
## [2,] 0.01154399  
## [3,] 0.3138786   
## [4,] 0.0126756   
## [5,] 0.3483659   
## [6,] 0.01495767  
## [7,] 0.2964675   
## [8,] 0.01621345  
## [9,] 0.3165756   
## [10,] 0.3096523

matrix(auctest)

## [,1]   
## [1,] 0.8504679  
## [2,] 0.8354719  
## [3,] 0.8404196  
## [4,] 0.8310196  
## [5,] 0.3382325  
## [6,] 0.8187513  
## [7,] 0.8492028  
## [8,] 0.807667   
## [9,] 0.8397271  
## [10,] 0.8414025

* **Logistic regression (stepwise selection)**
* start\_time <- Sys.time()  
    
  library(caret)
* ## Warning: package 'caret' was built under R version 3.5.3
* ## Loading required package: lattice
* ## Loading required package: ggplot2
* ## Warning: package 'ggplot2' was built under R version 3.5.3
* library(ROSE)
* ## Warning: package 'ROSE' was built under R version 3.5.3
* ## Loaded ROSE 0.0-3
* library(dplyr)
* ## Warning: package 'dplyr' was built under R version 3.5.3
* ##   
  ## 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(MASS)
* ##   
  ## Attaching package: 'MASS'
* ## The following object is masked from 'package:dplyr':  
  ##   
  ## select
* library(Metrics)
* ## Warning: package 'Metrics' was built under R version 3.5.3
* ##   
  ## Attaching package: 'Metrics'
* ## The following objects are masked from 'package:caret':  
  ##   
  ## precision, recall
* library(cvAUC)
* ## Warning: package 'cvAUC' was built under R version 3.5.3
* ## Loading required package: ROCR
* ## Warning: package 'ROCR' was built under R version 3.5.3
* ## Loading required package: gplots
* ## Warning: package 'gplots' was built under R version 3.5.3
* ##   
  ## Attaching package: 'gplots'
* ## The following object is masked from 'package:stats':  
  ##   
  ## lowess
* ## Loading required package: data.table
* ## Warning: package 'data.table' was built under R version 3.5.3
* ##   
  ## Attaching package: 'data.table'
* ## The following objects are masked from 'package:dplyr':  
  ##   
  ## between, first, last
* ##
* ## cvAUC version: 1.1.0
* ## Notice to cvAUC users: Major speed improvements in version 1.1.0
* ##
* validateaccruacy=list()  
  testaccruacy = list()  
  validaterecall = list()  
  testrecall = list()  
  validateprecision = list()  
  testreprecision = list()  
  validateF1 = list()  
  testF1 = list()  
  MSErrorvalidate = list()  
  MSErrortest = list()  
  aucvalidate = list()  
  auctest = list()  
    
  folds = cut(seq(1,nrow(Frauddata)),breaks = 10, labels = F)  
  for(f in 1:10){  
   ##random shuffle avoide overfit  
   Frauddata = Frauddata[sample(nrow(Frauddata)),]  
   Frauddata = Frauddata[sample(nrow(Frauddata)),]  
   Frauddata = Frauddata[sample(nrow(Frauddata)),]  
   Frauddata = Frauddata[sample(nrow(Frauddata)),]  
   Frauddata = Frauddata[sample(nrow(Frauddata)),]  
     
   testIndexes = which(folds == f, arr.ind =T)  
   ## spliting data  
   train = Frauddata[-testIndexes,]  
   testset = Frauddata[testIndexes,]  
   data\_train = sample(nrow(train), floor(nrow(train)\*0.8))  
   trainset = train[data\_train,]  
   trainset$isFraud = as.factor(trainset$isFraud)  
   validateset = train[-data\_train,]  
   ## performing forward and backward   
   stepMod <- step(lm(isFraud ~ . , data= Frauddatadirection = "both")

shortlistedVars <- names(unlist(stepMod[[1]]))

shortlistedVars <- shortlistedVars[!shortlistedVars %in% "(Intercept)"]   
 trainset = trainset[shortlistedVars]  
   
 ##Over and under sampling  
   
 trainsetboth = data.frame(ovun.sample(isFraud~.,data = trainset, method = "both", seed =1)$data)  
 trainsetboth$isFraud = as.factor(trainsetboth$isFraud)  
   
 ##logstic regression   
   
  
 logisticboth = glm(formula = isFraud~.,family = "binomial",data = trainsetboth)  
 logisticboth  
 probabilities <- logisticboth %>% predict(validateset, type = "response")  
 predicted.validate = ifelse(probabilities>0.5,1,0)  
 mean(predicted.validate == validateset$isFraud)  
 cmvalidate = as.matrix(table(Actual = validateset$isFraud, Predicted =predicted.validate ))  
  
 n1 = sum(cmvalidate)  
 nc1 = nrow(cmvalidate)  
 diag1 = diag(cmvalidate)  
 accuracyvalidate = sum(diag1) / n1   
 precisionvalidate = (cmvalidate[2,2])/(cmvalidate[2,2]+cmvalidate[1,2])  
 recallvalidate = (cmvalidate[2,2])/(cmvalidate[2,2]+cmvalidate[2,1])  
 f1validate = 2 \* precisionvalidate \* recallvalidate / (precisionvalidate + recallvalidate)   
 auc1 = AUC(predicted.validate,validateset$isFraud )  
 ##store result  
   
 validateaccruacy= append(validateaccruacy,accuracyvalidate)  
 validaterecall= append(validaterecall,recallvalidate)  
 validateprecision = append(validateprecision,precisionvalidate)  
 validateF1= append(validateF1,f1validate)  
 MSErrorvalidate = append(MSErrorvalidate,mse(actual = validateset$isFraud, predict = predicted.validate))  
 aucvalidate = append(aucvalidate, auc1)  
   
 ##  
 probabilitiestest <- logisticboth %>% predict(testset, type = "response")  
 predicted.test = ifelse(probabilitiestest>0.5,1,0)  
 mean(predicted.test == testset$isFraud)  
 cm = as.matrix(table(Actual = testset$isFraud, Predicted =predicted.test ))  
 cm  
 n = sum(cm)  
 nc = nrow(cm)  
 diag = diag(cm)  
 accuracy = sum(diag) / n   
 precision = (cm[2,2])/(cm[2,2]+cm[1,2])  
 recall = (cm[2,2])/(cm[2,2]+cm[2,1])  
 f1 = 2 \* precision \* recall / (precision + recall)   
   
 auc2 = AUC(predicted.test,testset$isFraud )  
 ##store result  
   
 testaccruacy= append(testaccruacy,accuracy)  
 testrecall=append(testrecall,recall)  
 testreprecision=append(testreprecision,precision)  
 testF1= append(testF1,f1)  
 MSErrortest = append(MSErrortest,mse(actual = testset$isFraud, predict = predicted.test) )  
 auctest = append(auctest, auc2)  
 }

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

end\_time <- Sys.time()

end\_time - start\_time

## Time difference of 1.127227 hours

##Validate Result  
 mean(as.numeric(validateaccruacy))

## [1] 0.6869895

mean(as.numeric(validaterecall))

## [1] 0.9958083

mean(as.numeric(validateF1))

## [1] 0.008306414

mean(as.numeric(MSErrorvalidate))

## [1] 0.3130105

mean(as.numeric(aucvalidate))

## [1] 0.8411958

matrix(validateaccruacy)

## [,1]   
## [1,] 0.6989676  
## [2,] 0.6936632  
## [3,] 0.6999088  
## [4,] 0.6908787  
## [5,] 0.686567   
## [6,] 0.694697   
## [7,] 0.6521027  
## [8,] 0.6708179  
## [9,] 0.6923456  
## [10,] 0.6899461

matrix(validaterecall)

## [,1]   
## [1,] 0.9947575  
## [2,] 0.989425   
## [3,] 0.996008   
## [4,] 0.9967763  
## [5,] 0.9965398  
## [6,] 0.9973369  
## [7,] 0.995935   
## [8,] 0.9980658  
## [9,] 0.9986737  
## [10,] 0.9945652

matrix(validateF1)

## [,1]   
## [1,] 0.008729155  
## [2,] 0.008461619  
## [3,] 0.008636206  
## [4,] 0.008658154  
## [5,] 0.007959187  
## [6,] 0.008495652  
## [7,] 0.007324784  
## [8,] 0.008145246  
## [9,] 0.008475912  
## [10,] 0.008178223

matrix(MSErrorvalidate)

## [,1]   
## [1,] 0.3010324  
## [2,] 0.3063368  
## [3,] 0.3000912  
## [4,] 0.3091213  
## [5,] 0.313433   
## [6,] 0.305303   
## [7,] 0.3478973  
## [8,] 0.3291821  
## [9,] 0.3076544  
## [10,] 0.3100539

matrix(aucvalidate)

## [,1]   
## [1,] 0.8466652  
## [2,] 0.8413485  
## [3,] 0.8477639  
## [4,] 0.8436201  
## [5,] 0.8413576  
## [6,] 0.8458182  
## [7,] 0.823797   
## [8,] 0.83422   
## [9,] 0.8453077  
## [10,] 0.8420597

##test Result  
 mean(as.numeric(testaccruacy))

## [1] 0.686758

mean(as.numeric(testrecall))

## [1] 0.9949231

mean(as.numeric(testF1))

## [1] 0.008233015

mean(as.numeric(MSErrortest))

## [1] 0.313242

mean(as.numeric(auctest))

## [1] 0.8406391

matrix(testaccruacy)

## [,1]   
## [1,] 0.6996898  
## [2,] 0.6932883  
## [3,] 0.700089   
## [4,] 0.6908805  
## [5,] 0.6859596  
## [6,] 0.6948663  
## [7,] 0.6514926  
## [8,] 0.6702129  
## [9,] 0.6910549  
## [10,] 0.6900459

matrix(testrecall)

## [,1]   
## [1,] 0.9963235  
## [2,] 0.990099   
## [3,] 0.9964953  
## [4,] 0.996129   
## [5,] 0.9976663  
## [6,] 0.9917355  
## [7,] 0.9976959  
## [8,] 0.9962917  
## [9,] 0.9927971  
## [10,] 0.9939976

matrix(testF1)

## [,1]   
## [1,] 0.008437899  
## [2,] 0.008132189  
## [3,] 0.008861049  
## [4,] 0.007789129  
## [5,] 0.008485426  
## [6,] 0.008579088  
## [7,] 0.007750342  
## [8,] 0.007623804  
## [9,] 0.008344096  
## [10,] 0.008327132

matrix(MSErrortest)

## [,1]   
## [1,] 0.3003102  
## [2,] 0.3067117  
## [3,] 0.299911   
## [4,] 0.3091195  
## [5,] 0.3140404  
## [6,] 0.3051337  
## [7,] 0.3485074  
## [8,] 0.3297871  
## [9,] 0.3089451  
## [10,] 0.3099541

matrix(auctest)

## [,1]   
## [1,] 0.8478162  
## [2,] 0.841505   
## [3,] 0.8480925  
## [4,] 0.8433186  
## [5,] 0.8416027  
## [6,] 0.843103   
## [7,] 0.8243578  
## [8,] 0.8330447  
## [9,] 0.8417283  
## [10,] 0.8418225

* **Decision tree (information Gain)**
* library(caret)
* ## Loading required package: lattice
* ## Loading required package: ggplot2
* library(FSelectorRcpp)
* library(ROSE)
* ## Loaded ROSE 0.0-3
* 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(Metrics)
* ##
* ## Attaching package: 'Metrics'
* ## The following objects are masked from 'package:caret':
* ##
* ## precision, recall
* library(cvAUC)
* ## Loading required package: ROCR
* ## Loading required package: gplots
* ##
* ## Attaching package: 'gplots'
* ## The following object is masked from 'package:stats':
* ##
* ## lowess
* ## Loading required package: data.table
* ##
* ## Attaching package: 'data.table'
* ## The following objects are masked from 'package:dplyr':
* ##
* ## between, first, last
* ##
* ## cvAUC version: 1.1.0
* ## Notice to cvAUC users: Major speed improvements in version 1.1.0
* ##
* validateaccruacy=list()
* testaccruacy = list()
* validaterecall = list()
* testrecall = list()
* validateprecision = list()
* testreprecision = list()
* validateF1 = list()
* testF1 = list()
* MSErrorvalidate = list()
* MSErrortest = list()
* aucvalidate = list()
* auctest = list()
* folds = cut(seq(1,nrow(Frauddata)),breaks = 10, labels = F)
* for(f in 1:10){
* Frauddata = Frauddata[sample(nrow(Frauddata)),]
* Frauddata = Frauddata[sample(nrow(Frauddata)),]
* Frauddata = Frauddata[sample(nrow(Frauddata)),]
* Frauddata = Frauddata[sample(nrow(Frauddata)),]
* Frauddata = Frauddata[sample(nrow(Frauddata)),]

* testIndexes = which(folds == f, arr.ind =T)
* ## spliting data
* train = Frauddata[-testIndexes,]
* testset = Frauddata[testIndexes,]
* data\_train = sample(nrow(train), floor(nrow(train)\*0.8))
* trainset = train[data\_train,]
* trainset$isFraud = as.factor(trainset$isFraud)
* validateset = train[-data\_train,]
* ## performing forward and backward
* x = information\_gain(isFraud ~ ., trainset)
* to\_formula(cut\_attrs(attrs = x), "Species")
* variable = cut\_attrs(attrs = x, k = 0.7)
* variable = c(variable[-1],"isFraud")
* trainset = trainset[variable]
* ##Over and under sampling
* trainsetboth = data.frame(ovun.sample(isFraud~.,data = trainset, method = "both", seed =1)$data)
* trainsetboth$isFraud = as.factor(trainsetboth$isFraud)
* ##Decision tree
* library(e1071)
* library(rpart)
* fit <- rpart(isFraud~., data = trainsetboth, method = 'class')
* fit
* Vpredict\_decisiontree = predict(fit,validateset, type = 'class')
* table\_validate = table(Actual = validateset$isFraud, Predicted = Vpredict\_decisiontree)
* cmvalidate = as.matrix(table\_validate)
* confusionMatrix(table\_validate)
* diag1 = diag(cmvalidate)
* n1 = sum(cmvalidate)
* accuracyvalidate = sum(diag1) / n1
* precisionvalidate = (cmvalidate[2,2])/(cmvalidate[2,2]+cmvalidate[1,2])
* recallvalidate = (cmvalidate[2,2])/(cmvalidate[2,2]+cmvalidate[2,1])
* f1validate = 2 \* precisionvalidate \* recallvalidate / (precisionvalidate + recallvalidate)
* auc1 = AUC(as.numeric(Vpredict\_decisiontree),as.numeric(validateset$isFraud) )
* ##store result
* validateaccruacy= append(validateaccruacy,accuracyvalidate)
* validaterecall= append(validaterecall,recallvalidate)
* validateprecision = append(validateprecision,precisionvalidate)
* validateF1= append(validateF1,f1validate)
* MSErrorvalidate = append(MSErrorvalidate,mse(actual = as.numeric(validateset$isFraud), predict =as.numeric( Vpredict\_decisiontree)))
* aucvalidate = append(aucvalidate, auc1)
* ##
* predict\_decisiontree = predict(fit,testset, type = 'class')
* table\_test = table(Actual =testset$isFraud,Predicted =predict\_decisiontree)
* cm = as.matrix(table\_test)
* confusionMatrix(table\_test)
* cm
* n = sum(cm)
* diag = diag(cm)
* accuracy = sum(diag) / n
* precision = (cm[2,2])/(cm[2,2]+cm[1,2])
* recall = (cm[2,2])/(cm[2,2]+cm[2,1])
* f1 = 2 \* precision \* recall / (precision + recall)
* data.frame(precision, recall, f1)
* auc2 = AUC(as.numeric(predict\_decisiontree),as.numeric(testset$isFraud) )
* ##store result
* testaccruacy= append(testaccruacy,accuracy)
* testrecall=append(testrecall,recall)
* testreprecision=append(testreprecision,precision)
* testF1= append(testF1,f1)
* MSErrortest = append(MSErrortest,mse(actual = as.numeric(testset$isFraud), predict = as.numeric(predict\_decisiontree) ))
* auctest = append(auctest, auc2)
* }
* end\_time <- Sys.time()
* end\_time - start\_time
* ## Time difference of 53.96558 mins
* mean(as.numeric(validateaccruacy))
* ## [1] 0.9639509
* mean(as.numeric(validaterecall))
* ## [1] 0.9922424
* mean(as.numeric(validateF1))
* ## [1] 0.06715547
* mean(as.numeric(MSErrorvalidate))
* ## [1] 1.108107
* mean(as.numeric(aucvalidate))
* ## [1] 0.9780781
* matrix(validateaccruacy)
* ## [,1]
* ## [1,] 0.9644294
* ## [2,] 0.9638959
* ## [3,] 0.9643403
* ## [4,] 0.9644163
* ## [5,] 0.9640723
* ## [6,] 0.9627041
* ## [7,] 0.9639815
* ## [8,] 0.9637091
* ## [9,] 0.9636942
* ## [10,] 0.9642661
* matrix(validaterecall)
* ## [,1]
* ## [1,] 0.9911202
* ## [2,] 0.9938734
* ## [3,] 0.9940984
* ## [4,] 0.9934938
* ## [5,] 0.9917145
* ## [6,] 0.9939516
* ## [7,] 0.992691
* ## [8,] 0.9892256
* ## [9,] 0.9897331
* ## [10,] 0.9925221
* matrix(validateF1)
* ## [,1]
* ## [1,] 0.06649863
* ## [2,] 0.06596038
* ## [3,] 0.06911014
* ## [4,] 0.06971489
* ## [5,] 0.07031338
* ## [6,] 0.06476616
* ## [7,] 0.06754221
* ## [8,] 0.06602099
* ## [9,] 0.06502968
* ## [10,] 0.06659824
* matrix(MSErrorvalidate)
* ## [,1]
* ## [1,] 1.106666
* ## [2,] 1.108281
* ## [3,] 1.106948
* ## [4,] 1.106716
* ## [5,] 1.107738
* ## [6,] 1.111856
* ## [7,] 1.108017
* ## [8,] 1.108817
* ## [9,] 1.108865
* ## [10,] 1.107163
* matrix(aucvalidate)
* ## [,1]
* ## [1,] 0.9777577
* ## [2,] 0.9788654
* ## [3,] 0.9791995
* ## [4,] 0.9789355
* ## [5,] 0.9778744
* ## [6,] 0.9783075
* ## [7,] 0.9783174
* ## [8,] 0.9764508
* ## [9,] 0.976697
* ## [10,] 0.9783759
* mean(as.numeric(testaccruacy))
* ## [1] 0.9640892
* mean(as.numeric(testrecall))
* ## [1] 0.9934629
* mean(as.numeric(testF1))
* ## [1] 0.06536096
* mean(as.numeric(MSErrortest))
* ## [1] 1.107699
* mean(as.numeric(auctest))
* ## [1] 0.9787575
* matrix(testaccruacy)
* ## [,1]
* ## [1,] 0.9645099
* ## [2,] 0.9637083
* ## [3,] 0.9641296
* ## [4,] 0.9647237
* ## [5,] 0.9642993
* ## [6,] 0.9631488
* ## [7,] 0.964106
* ## [8,] 0.9642066
* ## [9,] 0.9638907
* ## [10,] 0.9641688
* matrix(testrecall)
* ## [,1]
* ## [1,] 0.9916766
* ## [2,] 0.9935733
* ## [3,] 0.9933599
* ## [4,] 0.9936789
* ## [5,] 0.992629
* ## [6,] 0.9986702
* ## [7,] 0.9914634
* ## [8,] 0.9914634
* ## [9,] 0.9930955
* ## [10,] 0.9950187
* matrix(testF1)
* ## [,1]
* ## [1,] 0.06878634
* ## [2,] 0.06275115
* ## [3,] 0.06151569
* ## [4,] 0.06545364
* ## [5,] 0.06641733
* ## [6,] 0.06020281
* ## [7,] 0.06646501
* ## [8,] 0.06663934
* ## [9,] 0.06987571
* ## [10,] 0.06550254
* matrix(MSErrortest)
* ## [,1]
* ## [1,] 1.106426
* ## [2,] 1.108844
* ## [3,] 1.10758
* ## [4,] 1.105798
* ## [5,] 1.107064
* ## [6,] 1.110547
* ## [7,] 1.107638
* ## [8,] 1.107336
* ## [9,] 1.10829
* ## [10,] 1.107468
* matrix(auctest)
* ## [,1]
* ## [1,] 0.9780753
* ## [2,] 0.9786225
* ## [3,] 0.9787274
* ## [4,] 0.9791832
* ## [5,] 0.978446
* ## [6,] 0.9808885
* ## [7,] 0.977767
* ## [8,] 0.9778174
* ## [9,] 0.9784731
* ## [10,] 0.9795743
* **Decision tree (stepwise selection)**

start\_time <- Sys.time()  
  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(FSelectorRcpp)  
library(ROSE)

## Loaded ROSE 0.0-3

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(Metrics)

##   
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':  
##   
## precision, recall

library(cvAUC)

## Loading required package: ROCR

## Loading required package: gplots

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

## Loading required package: data.table

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

##

## cvAUC version: 1.1.0

## Notice to cvAUC users: Major speed improvements in version 1.1.0

##

validateaccruacy=list()  
testaccruacy = list()  
validaterecall = list()  
testrecall = list()  
validateprecision = list()  
testreprecision = list()  
validateF1 = list()  
testF1 = list()  
MSErrorvalidate = list()  
MSErrortest = list()  
aucvalidate = list()  
auctest = list()  
  
folds = cut(seq(1,nrow(Frauddata)),breaks = 10, labels = F)  
for(f in 1:10){  
 testIndexes = which(folds == f, arr.ind =T)  
  
 ## spliting data  
 train = Frauddata[-testIndexes,]  
 testset = Frauddata[testIndexes,]  
 data\_train = sample(nrow(train), floor(nrow(train)\*0.8))  
 trainset = train[data\_train,]  
 trainset$isFraud = as.factor(trainset$isFraud)  
 validateset = train[-data\_train,]  
 ## performing forward and backward   
stepMod <- step(lm(isFraud ~ . , data= Frauddatadirection = "both")

shortlistedVars <- names(unlist(stepMod[[1]]))

shortlistedVars <- shortlistedVars[!shortlistedVars %in% "(Intercept)"]  
   
 trainset = trainset[c(shortlistedVars)]  
   
 ##Over and under sampling  
   
 trainsetboth = data.frame(ovun.sample(isFraud~.,data = trainset, method = "both", seed =1)$data)  
 trainsetboth$isFraud = as.factor(trainsetboth$isFraud)  
 ##Decision tree  
 library(e1071)  
 library(rpart)  
 fit <- rpart(isFraud~., data = trainsetboth, method = 'class')  
 fit  
 Vpredict\_decisiontree = predict(fit,validateset, type = 'class')  
 table\_validate = table(Actual = validateset$isFraud, Predicted = Vpredict\_decisiontree)  
 cmvalidate = as.matrix(table\_validate)  
  
 n1 = sum(cmvalidate)  
 confusionMatrix(table\_validate)  
 diag1 = diag(cmvalidate)  
 accuracyvalidate = sum(diag1) / n1   
 precisionvalidate = (cmvalidate[2,2])/(cmvalidate[2,2]+cmvalidate[1,2])  
 recallvalidate = (cmvalidate[2,2])/(cmvalidate[2,2]+cmvalidate[2,1])  
  
 f1validate = 2 \* precisionvalidate \* recallvalidate / (precisionvalidate + recallvalidate)   
 data.frame(precisionvalidate, recallvalidate, f1validate)  
 auc1 = AUC(as.numeric(Vpredict\_decisiontree),as.numeric(validateset$isFraud))  
 ##store result  
   
 validateaccruacy= append(validateaccruacy,accuracyvalidate)  
 validaterecall= append(validaterecall,recallvalidate)  
 validateprecision = append(validateprecision,precisionvalidate)  
 validateF1= append(validateF1,f1validate)  
   
 MSErrorvalidate = append(MSErrorvalidate,mse(actual = as.numeric(validateset$isFraud), predict =as.numeric( Vpredict\_decisiontree)))  
 aucvalidate = append(aucvalidate, auc1)  
   
 ##  
   
   
   
 predict\_decisiontree = predict(fit,testset, type = 'class')  
 table\_test = table(Actual =testset$isFraud,Predicted =predict\_decisiontree)  
 cm = as.matrix(table\_test)  
 confusionMatrix(table\_test)  
  
 n = sum(cm)  
 diag = diag(cm)  
 accuracy = sum(diag) / n   
 precision = (cm[2,2])/(cm[2,2]+cm[1,2])  
 recall = (cm[2,2])/(cm[2,2]+cm[2,1])  
 f1 = 2 \* precision \* recall / (precision + recall)   
 auc2 = AUC(as.numeric(predict\_decisiontree),as.numeric(testset$isFraud) )  
 ##store result  
   
 testaccruacy= append(testaccruacy,accuracy)  
 testrecall=append(testrecall,recall)  
 testreprecision=append(testreprecision,precision)  
 testF1= append(testF1,f1)  
 MSErrortest = append(MSErrortest,mse(actual = as.numeric(testset$isFraud), predict = as.numeric(predict\_decisiontree) ))  
 auctest = append(auctest, auc2)  
   
 }  
end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 40.97556 mins

mean(as.numeric(validateaccruacy))

## [1] 0.9640785

mean(as.numeric(validaterecall))

## [1] 0.992821

mean(as.numeric(validateF1))

## [1] 0.06688673

mean(as.numeric(MSErrorvalidate))

## [1] 1.107728

mean(as.numeric(aucvalidate))

## [1] 0.9784309

matrix(validateaccruacy)

## [,1]   
## [1,] 0.9642766  
## [2,] 0.9642862  
## [3,] 0.9643133  
## [4,] 0.9639867  
## [5,] 0.9640365  
## [6,] 0.9638199  
## [7,] 0.9641395  
## [8,] 0.9643316  
## [9,] 0.9634166  
## [10,] 0.9641779

matrix(validaterecall)

## [,1]   
## [1,] 0.9974522  
## [2,] 0.9919532  
## [3,] 0.9932015  
## [4,] 0.9917197  
## [5,] 0.994152   
## [6,] 0.9919204  
## [7,] 0.9908854  
## [8,] 0.9925558  
## [9,] 0.995481   
## [10,] 0.9888889

matrix(validateF1)

## [,1]   
## [1,] 0.07110909  
## [2,] 0.06218187  
## [3,] 0.07290462  
## [4,] 0.07019996  
## [5,] 0.06915567  
## [6,] 0.0715246   
## [7,] 0.06900304  
## [8,] 0.07264472  
## [9,] 0.06856076  
## [10,] 0.04158296

matrix(MSErrorvalidate)

## [,1]   
## [1,] 1.107156  
## [2,] 1.107103  
## [3,] 1.107022  
## [4,] 1.107994  
## [5,] 1.107859  
## [6,] 1.108495  
## [7,] 1.107533  
## [8,] 1.106963  
## [9,] 1.109726  
## [10,] 1.107431

matrix(aucvalidate)

## [,1]   
## [1,] 0.9808416  
## [2,] 0.9781032  
## [3,] 0.9787369  
## [4,] 0.9778342  
## [5,] 0.979074   
## [6,] 0.9778504  
## [7,] 0.9774945  
## [8,] 0.9784238  
## [9,] 0.979427   
## [10,] 0.9765237

mean(as.numeric(testaccruacy))

## [1] 0.963902

mean(as.numeric(testrecall))

## [1] 0.9901335

mean(as.numeric(testF1))

## [1] 0.06060939

mean(as.numeric(MSErrortest))

## [1] 1.108249

mean(as.numeric(auctest))

## [1] 0.9769998

matrix(testaccruacy)

## [,1]   
## [1,] 0.9620659  
## [2,] 0.9636471  
## [3,] 0.9635653  
## [4,] 0.9651071  
## [5,] 0.964128   
## [6,] 0.9642081  
## [7,] 0.9646278  
## [8,] 0.9643527  
## [9,] 0.9644329  
## [10,] 0.9628848

matrix(testrecall)

## [,1]   
## [1,] 0.9530026  
## [2,] 0.9939499  
## [3,] 0.9943182  
## [4,] 0.9950739  
## [5,] 0.9900249  
## [6,] 0.9918367  
## [7,] 1   
## [8,] 0.9936709  
## [9,] 0.9979592  
## [10,] 0.9914984

matrix(testF1)

## [,1]   
## [1,] 0.02935736  
## [2,] 0.09044436  
## [3,] 0.02931078  
## [4,] 0.03511669  
## [5,] 0.03361843  
## [6,] 0.04093493  
## [7,] 0.03878022  
## [8,] 0.02694238  
## [9,] 0.04142664  
## [10,] 0.2401622

matrix(MSErrortest)

## [,1]   
## [1,] 1.113689  
## [2,] 1.109015  
## [3,] 1.109291  
## [4,] 1.104666  
## [5,] 1.107591  
## [6,] 1.10735   
## [7,] 1.106117  
## [8,] 1.106929  
## [9,] 1.106695  
## [10,] 1.111144

matrix(auctest)

* ## [,1]   
  ## [1,] 0.957537   
  ## [2,] 0.9787709  
  ## [3,] 0.9789332  
  ## [4,] 0.9800809  
  ## [5,] 0.9770683  
  ## [6,] 0.9780118  
  ## [7,] 0.9823013  
  ## [8,] 0.9790045  
  ## [9,] 0.9811831  
  ## [10,] 0.9771065
* **Naïve Bayes**

start\_time <- Sys.time()  
  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(FSelectorRcpp)  
library(ROSE)

## Loaded ROSE 0.0-3

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(Metrics)

##   
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':  
##   
## precision, recall

library(cvAUC)

## Loading required package: ROCR

## Loading required package: gplots

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

## Loading required package: data.table

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

##

## cvAUC version: 1.1.0

## Notice to cvAUC users: Major speed improvements in version 1.1.0

##

library(class)  
  
validateaccruacy=list()  
testaccruacy = list()  
validaterecall = list()  
testrecall = list()  
validateprecision = list()  
testreprecision = list()  
validateF1 = list()  
testF1 = list()  
MSErrorvalidate = list()  
MSErrortest = list()  
aucvalidate = list()  
auctest = list()  
  
folds = cut(seq(1,nrow(Frauddata)),breaks = 10, labels = F)  
for(f in 1:10){  
testIndexes = which(folds == f, arr.ind =T)  
  
 ## spliting data  
 train = Frauddata[-testIndexes,]  
 testset = Frauddata[testIndexes,]  
 data\_train = sample(nrow(train), floor(nrow(train)\*0.8))  
 trainset = train[data\_train,]  
 trainset$isFraud = as.factor(trainset$isFraud)  
 validateset = train[-data\_train,]  
  
  
  
 ##Over and under sampling  
   
 trainsetboth = data.frame(ovun.sample(isFraud~.,data = trainset, method = "both", seed =1)$data)  
 trainsetboth$isFraud = as.factor(trainsetboth$isFraud)  
   
   
   
   
 #Navie Bayes  
 library(e1071)  
 NBclassfier=naiveBayes(isFraud~step+amount+type+nameOrig+oldbalanceOrg+newbalanceOrig+oldbalanceDest+newbalanceDest+isFlaggedFraud, data=trainsetboth)  
 NBclassfier  
 ##  
Vpredict\_Bayes = predict(NBclassfier,newdata = validateset, type = "class")  
 table\_validate = table(Actual = validateset$isFraud, Predicted = Vpredict\_Bayes)  
 cmvalidate = as.matrix(table\_validate)  
 confusionMatrix(table\_validate)  
 diag1 = diag(cmvalidate)  
 n1 = sum(cmvalidate)  
 accuracyvalidate = sum(diag1) / n1   
 precisionvalidate = (cmvalidate[2,2])/(cmvalidate[2,2]+cmvalidate[1,2])  
 recallvalidate = (cmvalidate[2,2])/(cmvalidate[2,2]+cmvalidate[2,1])  
 accuracyvalidate  
 f1validate = 2 \* precisionvalidate \* recallvalidate / (precisionvalidate + recallvalidate)   
 data.frame(precisionvalidate, recallvalidate, f1validate)  
 auc1 = AUC(as.numeric(Vpredict\_Bayes),as.numeric(validateset$isFraud) )  
 ##store result  
  
 validateaccruacy= append(validateaccruacy,accuracyvalidate)  
 validaterecall= append(validaterecall,recallvalidate)  
 validateprecision = append(validateprecision,precisionvalidate)  
 validateF1= append(validateF1,f1validate)  
 MSErrorvalidate = append(MSErrorvalidate,mse(actual = as.numeric(validateset$isFraud), predict =as.numeric( Vpredict\_Bayes)))  
 aucvalidate = append(aucvalidate, auc1)  
 mean(as.numeric(aucvalidate))  
   
   
 predict\_Bayes = predict(NBclassfier,testset, type = "class")  
 table\_test = table(Actual =testset$isFraud,Predicted =predict\_Bayes)  
 cm = as.matrix(table\_test)  
 confusionMatrix(table\_test)  
  
 n = sum(cm)  
 diag = diag(cm)  
 accuracy = sum(diag) / n   
 precision = (cm[2,2])/(cm[2,2]+cm[1,2])  
 recall = (cm[2,2])/(cm[2,2]+cm[2,1])  
 f1 = 2 \* precision \* recall / (precision + recall)   
 data.frame(precision, recall, f1)  
 auc2 = AUC(as.numeric(predict\_Bayes),as.numeric(testset$isFraud) )  
 ##store result  
   
 testaccruacy= append(testaccruacy,accuracy)  
 testrecall=append(testrecall,recall)  
 testreprecision=append(testreprecision,precision)  
 testF1= append(testF1,f1)  
 MSErrortest = append(MSErrortest,mse(actual = as.numeric(testset$isFraud), predict = as.numeric(predict\_Bayes) ))  
 auctest = append(auctest, auc2)  
 }

## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion

end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 1.449594 hours

mean(as.numeric(validateaccruacy))

## [1] 0.9868576

mean(as.numeric(validaterecall))

## [1] 0.2767851

mean(as.numeric(validateF1))

## [1] 0.05451788

mean(as.numeric(MSErrorvalidate))

## [1] 1.035694

mean(as.numeric(aucvalidate))

## [1] 0.6322805

matrix(validateaccruacy)

## [,1]   
## [1,] 0.9874432  
## [2,] 0.9898042  
## [3,] 0.9886018  
## [4,] 0.9910379  
## [5,] 0.9884438  
## [6,] 0.9882796  
## [7,] 0.9871157  
## [8,] 0.9871218  
## [9,] 0.9872615  
## [10,] 0.9734666

matrix(validaterecall)

## [,1]   
## [1,] 0.2940447  
## [2,] 0.2604317  
## [3,] 0.2474293  
## [4,] 0.2574189  
## [5,] 0.2425016  
## [6,] 0.2392132  
## [7,] 0.3229974  
## [8,] 0.2647995  
## [9,] 0.2819525  
## [10,] 0.3570621

matrix(validateF1)

## [,1]   
## [1,] 0.06184356  
## [2,] 0.05838239  
## [3,] 0.05570023  
## [4,] 0.06775658  
## [5,] 0.05430511  
## [6,] 0.05318474  
## [7,] 0.06346789  
## [8,] 0.05339837  
## [9,] 0.05676602  
## [10,] 0.02037395

matrix(MSErrorvalidate)

## [,1]   
## [1,] 1.033696  
## [2,] 1.026997  
## [3,] 1.030105  
## [4,] 1.023128  
## [5,] 1.030523  
## [6,] 1.030973  
## [7,] 1.034993  
## [8,] 1.034601  
## [9,] 1.034311  
## [10,] 1.077613

matrix(aucvalidate)

## [,1]   
## [1,] 0.6412326  
## [2,] 0.6255611  
## [3,] 0.6185197  
## [4,] 0.6246931  
## [5,] 0.6159837  
## [6,] 0.6142625  
## [7,] 0.655506   
## [8,] 0.6264568  
## [9,] 0.6350871  
## [10,] 0.6655027

mean(as.numeric(testaccruacy))

## [1] 0.9883528

mean(as.numeric(testrecall))

## [1] 0.2177543

mean(as.numeric(testF1))

## [1] 0.06060759

mean(as.numeric(MSErrortest))

## [1] 1.031086

mean(as.numeric(auctest))

## [1] 0.6035278

matrix(testaccruacy)

## [,1]   
## [1,] 0.9931506  
## [2,] 0.995648   
## [3,] 0.9963034  
## [4,] 0.9964417  
## [5,] 0.9941879  
## [6,] 0.9955207  
## [7,] 0.975235   
## [8,] 0.9739463  
## [9,] 0.9824915  
## [10,] 0.9806023

matrix(testrecall)

## [,1]   
## [1,] 0.1331593  
## [2,] 0.1910112  
## [3,] 0.1988636  
## [4,] 0.2019704  
## [5,] 0.1895262  
## [6,] 0.2571429  
## [7,] 0.2709251  
## [8,] 0.1740506  
## [9,] 0.255102   
## [10,] 0.3057917

matrix(testF1)

## [,1]   
## [1,] 0.02286996   
## [2,] 0.1376518   
## [3,] 0.05617978   
## [4,] 0.0675453   
## [5,] 0.03948052   
## [6,] 0.08123791   
## [7,] 0.01537212   
## [8,] 0.006591958  
## [9,] 0.02194908   
## [10,] 0.1571975

matrix(MSErrortest)

## [,1]   
## [1,] 1.018461  
## [2,] 1.007172  
## [3,] 1.009317  
## [4,] 1.008638  
## [5,] 1.015393  
## [6,] 1.011149  
## [7,] 1.072214  
## [8,] 1.07652   
## [9,] 1.050231  
## [10,] 1.041766

matrix(auctest)

## [,1]   
## [1,] 0.5634139  
## [2,] 0.5940625  
## [3,] 0.5978042  
## [4,] 0.5994597  
## [5,] 0.5921108  
## [6,] 0.6266163  
## [7,] 0.6233315  
## [8,] 0.5741972  
## [9,] 0.6190771  
## [10,] 0.6452049