Project 4

Durga Pokharel

05/01/2022

## R Markdown

# Loading the data here  
library(haven)  
bank\_loan\_df <- read\_sav("F:/MDS-Private-Study-Materials/First Semester/Statistical Computing with R/Assignments/Data/P4\_bankloan\_5000\_clients.sav")  
bank\_loan\_df$defaulted\_loan<-as.factor(bank\_loan\_df$defaulted\_loan)  
bank\_loan\_df$education\_level<-as.factor(bank\_loan\_df$education\_level)  
str(bank\_loan\_df)

## tibble [5,000 x 9] (S3: tbl\_df/tbl/data.frame)  
## $ age : num [1:5000] 41 30 40 41 57 45 36 39 43 34 ...  
## ..- attr(\*, "label")= chr "Age in years"  
## ..- attr(\*, "format.spss")= chr "F4.0"  
## ..- attr(\*, "display\_width")= int 6  
## $ education\_level : Factor w/ 5 levels "1","2","3","4",..: 3 1 1 1 1 1 1 1 1 3 ...  
## $ current\_employ\_year : num [1:5000] 17 13 15 15 7 0 1 20 12 7 ...  
## ..- attr(\*, "label")= chr "Years with current employer"  
## ..- attr(\*, "format.spss")= chr "F4.0"  
## $ current\_address\_year: num [1:5000] 12 8 14 14 37 13 3 9 11 12 ...  
## ..- attr(\*, "label")= chr "Years at current address"  
## ..- attr(\*, "format.spss")= chr "F4.0"  
## ..- attr(\*, "display\_width")= int 9  
## $ income\_household : num [1:5000] 35.9 46.7 61.8 72 25.6 28.1 19.6 80.5 68.7 33.8 ...  
## ..- attr(\*, "label")= chr "Household income in thousands"  
## ..- attr(\*, "format.spss")= chr "F8.2"  
## ..- attr(\*, "display\_width")= int 10  
## $ debt\_income\_ratio : num [1:5000] 11.9 17.9 10.6 29.7 15.9 ...  
## ..- attr(\*, "label")= chr "Debt to income ratio (x100)"  
## ..- attr(\*, "format.spss")= chr "F8.2"  
## ..- attr(\*, "display\_width")= int 10  
## $ credit\_card\_debt : num [1:5000] 0.504 1.353 3.439 4.166 1.498 ...  
## ..- attr(\*, "label")= chr "Credit card debt in thousands"  
## ..- attr(\*, "format.spss")= chr "F8.2"  
## ..- attr(\*, "display\_width")= int 10  
## $ other\_debts : num [1:5000] 3.77 7 3.14 17.2 2.56 ...  
## ..- attr(\*, "label")= chr "Other debt in thousands"  
## ..- attr(\*, "format.spss")= chr "F8.2"  
## ..- attr(\*, "display\_width")= int 10  
## $ defaulted\_loan : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...  
## - attr(\*, "label")= chr "Bank Loan Default -- Binning"  
## - attr(\*, "notes")= chr [1:7] "DOCUMENT This is a hypothetical data file that concerns a bank's efforts to redu" " ce" "the rate of loan defaults. This file contains financial and demographic" "information on 5000 past customers that the bank will use to create binning rule" ...

This is a hypothetical data file that concerns a bank’s efforts to redu" " ce" “the rate of loan defaults. This file contains financial and demographic” "information on 5000 past customers that the bank will use to create binning rule.

# Train Test Validation

library(caret)

## Warning: package 'caret' was built under R version 4.1.2

## Loading required package: ggplot2

## Loading required package: lattice

# Testing the data into training and testing set

set.seed(1234)  
ind = sample(2,nrow(bank\_loan\_df),replace = T, prob = c(0.8, 0.3))  
train\_data <- bank\_loan\_df[ind==1,]  
test\_data <- bank\_loan\_df[ind==2,]

# Logistic Regression

## Training the logictic model

logic\_model <- train(defaulted\_loan~., data = train\_data, method = "glm", family= "binomial")

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

summary(logic\_model)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.5859 -0.6588 -0.3438 0.1138 3.3020   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.169417 0.268244 -4.360 1.30e-05 \*\*\*  
## age 0.004410 0.008174 0.540 0.5895   
## education\_level2 0.224490 0.108351 2.072 0.0383 \*   
## education\_level3 0.259264 0.153824 1.685 0.0919 .   
## education\_level4 0.250029 0.185073 1.351 0.1767   
## education\_level5 0.018646 0.446741 0.042 0.9667   
## current\_employ\_year -0.182293 0.012469 -14.619 < 2e-16 \*\*\*  
## current\_address\_year -0.092239 0.010140 -9.096 < 2e-16 \*\*\*  
## income\_household -0.003279 0.003835 -0.855 0.3925   
## debt\_income\_ratio 0.099422 0.012702 7.827 4.98e-15 \*\*\*  
## credit\_card\_debt 0.425010 0.043483 9.774 < 2e-16 \*\*\*  
## other\_debts 0.013697 0.030109 0.455 0.6492   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 4124.2 on 3650 degrees of freedom  
## Residual deviance: 2946.7 on 3639 degrees of freedom  
## AIC: 2970.7  
##   
## Number of Fisher Scoring iterations: 6

# Testing the Logistic model

pred1 <- predict(logic\_model, test\_data)

# Confusion Matrix

confusionMatrix(pred1, test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 944 177  
## 1 70 158  
##   
## Accuracy : 0.8169   
## 95% CI : (0.7952, 0.8372)  
## No Information Rate : 0.7517   
## P-Value [Acc > NIR] : 6.180e-09   
##   
## Kappa : 0.4508   
##   
## Mcnemar's Test P-Value : 1.534e-11   
##   
## Sensitivity : 0.9310   
## Specificity : 0.4716   
## Pos Pred Value : 0.8421   
## Neg Pred Value : 0.6930   
## Prevalence : 0.7517   
## Detection Rate : 0.6998   
## Detection Prevalence : 0.8310   
## Balanced Accuracy : 0.7013   
##   
## 'Positive' Class : 0   
##

# KNN Model with train/test validation

knn\_model<-train(defaulted\_loan~.,data = train\_data,  
 method="knn",  
 preProcess = c("center", "scale"),  
 tuneLength = 10  
 )

# Obtain the result

knn\_model$result

## k Accuracy Kappa AccuracySD KappaSD  
## 1 5 0.7454197 0.2900344 0.011038244 0.02396152  
## 2 7 0.7580398 0.3100812 0.011263579 0.02591339  
## 3 9 0.7653001 0.3192384 0.012598023 0.02663501  
## 4 11 0.7699141 0.3216275 0.012504271 0.02653078  
## 5 13 0.7726898 0.3244535 0.013151810 0.03094986  
## 6 15 0.7762592 0.3283860 0.012201179 0.02642366  
## 7 17 0.7777579 0.3261075 0.012592331 0.02979285  
## 8 19 0.7800209 0.3287250 0.010386847 0.02551861  
## 9 21 0.7818723 0.3300842 0.009801227 0.02492881  
## 10 23 0.7831656 0.3305184 0.009523488 0.02541875

# Testing the model

pred2 <- predict(knn\_model, test\_data)

# Confusion Matrix

confusionMatrix(pred2,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 949 218  
## 1 65 117  
##   
## Accuracy : 0.7902   
## 95% CI : (0.7675, 0.8117)  
## No Information Rate : 0.7517   
## P-Value [Acc > NIR] : 0.0004827   
##   
## Kappa : 0.3366   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9359   
## Specificity : 0.3493   
## Pos Pred Value : 0.8132   
## Neg Pred Value : 0.6429   
## Prevalence : 0.7517   
## Detection Rate : 0.7035   
## Detection Prevalence : 0.8651   
## Balanced Accuracy : 0.6426   
##   
## 'Positive' Class : 0   
##

# Fitting Naive Bayes

## Training the model

library(e1071)  
naive\_model <- naiveBayes(defaulted\_loan~., train\_data)  
summary(naive\_model)

## Length Class Mode   
## apriori 2 table numeric   
## tables 8 -none- list   
## levels 2 -none- character  
## isnumeric 8 -none- logical   
## call 4 -none- call

## Testing the model

pred3 <- predict(naive\_model, test\_data)

## Confusion Matrix

confusionMatrix(pred3,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 974 260  
## 1 40 75  
##   
## Accuracy : 0.7776   
## 95% CI : (0.7545, 0.7996)  
## No Information Rate : 0.7517   
## P-Value [Acc > NIR] : 0.01408   
##   
## Kappa : 0.2364   
##   
## Mcnemar's Test P-Value : < 2e-16   
##   
## Sensitivity : 0.9606   
## Specificity : 0.2239   
## Pos Pred Value : 0.7893   
## Neg Pred Value : 0.6522   
## Prevalence : 0.7517   
## Detection Rate : 0.7220   
## Detection Prevalence : 0.9148   
## Balanced Accuracy : 0.5922   
##   
## 'Positive' Class : 0   
##

# Support Vector Machine (SVM) Model

## Training the model

svm\_model <- svm(formula= defaulted\_loan~., data = train\_data, type = "C-classification", kernel= "linear")  
summary(svm\_model)

##   
## Call:  
## svm(formula = defaulted\_loan ~ ., data = train\_data, type = "C-classification",   
## kernel = "linear")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 1   
##   
## Number of Support Vectors: 1614  
##   
## ( 810 804 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## 0 1

## Testing the Model

pred4 <- predict(svm\_model, test\_data)

## Confusion Matrix

confusionMatrix(pred4, test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 960 200  
## 1 54 135  
##   
## Accuracy : 0.8117   
## 95% CI : (0.7898, 0.8322)  
## No Information Rate : 0.7517   
## P-Value [Acc > NIR] : 8.805e-08   
##   
## Kappa : 0.4095   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9467   
## Specificity : 0.4030   
## Pos Pred Value : 0.8276   
## Neg Pred Value : 0.7143   
## Prevalence : 0.7517   
## Detection Rate : 0.7116   
## Detection Prevalence : 0.8599   
## Balanced Accuracy : 0.6749   
##   
## 'Positive' Class : 0   
##

# Decision Tree Model

## Training the model

decision\_tree\_model <-train(defaulted\_loan~.,  
 data = train\_data,  
 method="rpart",  
 parms = list(split = "information"),  
 tuneLength=10  
 )

## Testing the model

pred5 <- predict(decision\_tree\_model, test\_data)

## Confusion Matrix

confusionMatrix(pred5, test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 956 235  
## 1 58 100  
##   
## Accuracy : 0.7828   
## 95% CI : (0.7598, 0.8045)  
## No Information Rate : 0.7517   
## P-Value [Acc > NIR] : 0.004044   
##   
## Kappa : 0.2932   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9428   
## Specificity : 0.2985   
## Pos Pred Value : 0.8027   
## Neg Pred Value : 0.6329   
## Prevalence : 0.7517   
## Detection Rate : 0.7087   
## Detection Prevalence : 0.8829   
## Balanced Accuracy : 0.6207   
##   
## 'Positive' Class : 0   
##

# Artifical Neural Network (ANN) Model

## Training the Model

ann\_model <- train(defaulted\_loan ~ ., data = train\_data,   
method = "nnet",  
preProcess = c("center","scale"),   
maxit = 250, # Maximum number of iterations  
tuneGrid = data.frame(size = 1, decay = 0),  
 # tuneGrid = data.frame(size = 0, decay = 0),skip=TRUE, # Technically, this is log-reg  
metric = "Accuracy")

## # weights: 14  
## initial value 2490.154378   
## iter 10 value 1603.415337  
## iter 20 value 1512.403963  
## iter 30 value 1480.119731  
## iter 40 value 1476.081679  
## iter 50 value 1469.305229  
## iter 60 value 1467.397161  
## iter 70 value 1467.376879  
## iter 80 value 1467.008906  
## iter 90 value 1466.875450  
## final value 1466.875090   
## converged  
## # weights: 14  
## initial value 2407.105942   
## iter 10 value 1621.907412  
## iter 20 value 1558.129266  
## iter 30 value 1530.209595  
## iter 40 value 1475.296367  
## iter 50 value 1455.253543  
## iter 60 value 1454.643391  
## iter 70 value 1452.315447  
## iter 80 value 1452.069503  
## iter 90 value 1452.067419  
## iter 90 value 1452.067410  
## iter 90 value 1452.067409  
## final value 1452.067409   
## converged  
## # weights: 14  
## initial value 3316.269093   
## iter 10 value 2091.326271  
## iter 20 value 1688.424852  
## iter 30 value 1627.575732  
## iter 40 value 1578.208844  
## iter 50 value 1540.242658  
## iter 60 value 1536.818068  
## iter 70 value 1534.901951  
## final value 1534.887159   
## converged  
## # weights: 14  
## initial value 2385.358031   
## iter 10 value 1668.690637  
## iter 20 value 1583.737340  
## iter 30 value 1544.955813  
## iter 40 value 1473.578642  
## iter 50 value 1451.624024  
## iter 60 value 1431.805159  
## iter 70 value 1429.709747  
## iter 80 value 1427.802753  
## iter 90 value 1426.693990  
## iter 100 value 1426.660252  
## final value 1426.659975   
## converged  
## # weights: 14  
## initial value 3201.590239   
## iter 10 value 1735.394117  
## iter 20 value 1639.370673  
## iter 30 value 1557.667764  
## iter 40 value 1520.658633  
## iter 50 value 1505.406878  
## iter 60 value 1500.146505  
## iter 70 value 1500.001251  
## iter 80 value 1498.576661  
## iter 90 value 1498.031963  
## iter 100 value 1498.018686  
## final value 1498.017137   
## converged  
## # weights: 14  
## initial value 2699.729482   
## iter 10 value 1708.185902  
## iter 20 value 1533.672892  
## iter 30 value 1485.389639  
## iter 40 value 1472.897282  
## iter 50 value 1463.951011  
## iter 60 value 1461.864517  
## iter 70 value 1461.852301  
## iter 80 value 1461.295186  
## iter 90 value 1461.191318  
## iter 100 value 1461.140191  
## iter 110 value 1461.060769  
## iter 120 value 1461.027597  
## iter 120 value 1461.027593  
## iter 120 value 1461.027591  
## final value 1461.027591   
## converged  
## # weights: 14  
## initial value 2918.658527   
## iter 10 value 1747.555356  
## iter 20 value 1578.274428  
## iter 30 value 1541.761037  
## iter 40 value 1512.216219  
## iter 50 value 1481.010030  
## iter 60 value 1476.352511  
## iter 70 value 1476.295136  
## iter 80 value 1475.663670  
## iter 90 value 1475.556116  
## iter 100 value 1475.553767  
## iter 110 value 1475.534585  
## final value 1475.534359   
## converged  
## # weights: 14  
## initial value 2414.595151   
## iter 10 value 1568.757814  
## iter 20 value 1502.778270  
## iter 30 value 1472.193138  
## iter 40 value 1466.293800  
## iter 50 value 1462.895500  
## iter 60 value 1461.939982  
## iter 70 value 1461.885035  
## iter 80 value 1461.840147  
## final value 1461.834804   
## converged  
## # weights: 14  
## initial value 2930.006588   
## iter 10 value 1757.770588  
## iter 20 value 1630.089973  
## iter 30 value 1571.955426  
## iter 40 value 1530.350816  
## iter 50 value 1510.110349  
## iter 60 value 1504.706374  
## iter 70 value 1504.643534  
## iter 80 value 1503.452073  
## iter 90 value 1503.105631  
## iter 100 value 1503.094003  
## final value 1503.093402   
## converged  
## # weights: 14  
## initial value 2189.427854   
## iter 10 value 1676.144100  
## iter 20 value 1599.071383  
## iter 30 value 1551.234187  
## iter 40 value 1539.680692  
## iter 50 value 1536.133313  
## final value 1536.132575   
## converged  
## # weights: 14  
## initial value 3830.386414   
## iter 10 value 1732.091410  
## iter 20 value 1554.539280  
## iter 30 value 1476.587233  
## iter 40 value 1452.257887  
## iter 50 value 1440.977816  
## iter 60 value 1436.619308  
## iter 70 value 1436.542765  
## iter 80 value 1435.267041  
## iter 90 value 1434.971108  
## final value 1434.964372   
## converged  
## # weights: 14  
## initial value 2393.249806   
## iter 10 value 1606.742894  
## iter 20 value 1494.230041  
## iter 30 value 1470.391210  
## iter 40 value 1469.559992  
## iter 50 value 1466.959732  
## final value 1466.902721   
## converged  
## # weights: 14  
## initial value 2303.745081   
## iter 10 value 1704.168147  
## iter 20 value 1529.587089  
## iter 30 value 1492.297275  
## iter 40 value 1464.104086  
## iter 50 value 1449.604997  
## iter 60 value 1446.084781  
## iter 70 value 1445.953737  
## iter 80 value 1444.803218  
## iter 90 value 1444.548183  
## iter 100 value 1444.539852  
## iter 110 value 1444.454647  
## iter 120 value 1444.440015  
## iter 120 value 1444.440011  
## final value 1444.439950   
## converged  
## # weights: 14  
## initial value 2803.286864   
## iter 10 value 1732.489256  
## iter 20 value 1533.201667  
## iter 30 value 1447.366296  
## iter 40 value 1426.178074  
## iter 50 value 1414.650132  
## iter 60 value 1410.829602  
## iter 70 value 1410.578650  
## iter 80 value 1410.384875  
## iter 90 value 1410.172837  
## iter 100 value 1410.166008  
## iter 100 value 1410.166003  
## iter 100 value 1410.165995  
## final value 1410.165995   
## converged  
## # weights: 14  
## initial value 2211.136287   
## iter 10 value 1551.419973  
## iter 20 value 1472.810866  
## iter 30 value 1444.043116  
## iter 40 value 1442.292392  
## iter 50 value 1440.049935  
## iter 60 value 1439.845773  
## final value 1439.845706   
## converged  
## # weights: 14  
## initial value 2758.551553   
## iter 10 value 1682.966532  
## iter 20 value 1552.372647  
## iter 30 value 1487.100391  
## iter 40 value 1472.196531  
## iter 50 value 1459.148734  
## iter 60 value 1456.132849  
## iter 70 value 1456.087110  
## iter 80 value 1455.471437  
## iter 90 value 1455.289969  
## iter 100 value 1455.237340  
## iter 110 value 1455.134117  
## iter 120 value 1455.109959  
## iter 120 value 1455.109954  
## final value 1455.109915   
## converged  
## # weights: 14  
## initial value 3230.762626   
## iter 10 value 2014.197184  
## iter 20 value 1616.238525  
## iter 30 value 1560.032833  
## iter 40 value 1545.733807  
## iter 50 value 1492.808071  
## iter 60 value 1460.831971  
## iter 70 value 1455.416018  
## iter 80 value 1448.128086  
## iter 90 value 1446.111942  
## iter 100 value 1446.097170  
## iter 110 value 1445.718095  
## iter 120 value 1445.018042  
## iter 130 value 1444.847994  
## final value 1444.846358   
## converged  
## # weights: 14  
## initial value 2086.395138   
## iter 10 value 1651.981162  
## iter 20 value 1570.275698  
## iter 30 value 1552.487729  
## iter 40 value 1533.896054  
## iter 50 value 1486.347284  
## iter 60 value 1455.570803  
## iter 70 value 1452.722282  
## iter 80 value 1445.433809  
## iter 90 value 1444.580195  
## iter 100 value 1444.564862  
## iter 110 value 1444.157336  
## iter 120 value 1444.042685  
## iter 130 value 1444.041039  
## final value 1444.040972   
## converged  
## # weights: 14  
## initial value 2167.978779   
## iter 10 value 1567.053008  
## iter 20 value 1482.319887  
## iter 30 value 1478.720905  
## iter 40 value 1477.910163  
## iter 50 value 1473.937284  
## iter 60 value 1472.932923  
## iter 70 value 1472.908336  
## iter 80 value 1472.501261  
## iter 90 value 1472.351506  
## final value 1472.350470   
## converged  
## # weights: 14  
## initial value 3352.063487   
## iter 10 value 1611.897544  
## iter 20 value 1542.164826  
## iter 30 value 1480.731131  
## iter 40 value 1464.512410  
## iter 50 value 1460.819301  
## iter 60 value 1456.108430  
## iter 70 value 1455.605860  
## iter 80 value 1455.521161  
## iter 90 value 1455.182616  
## iter 100 value 1455.097729  
## final value 1455.097357   
## converged  
## # weights: 14  
## initial value 2342.129872   
## iter 10 value 1714.695204  
## iter 20 value 1451.318364  
## iter 30 value 1442.729608  
## iter 40 value 1442.590767  
## iter 50 value 1441.556518  
## iter 60 value 1441.294491  
## iter 70 value 1441.291422  
## iter 80 value 1441.201001  
## iter 90 value 1441.165284  
## iter 90 value 1441.165272  
## final value 1441.165244   
## converged  
## # weights: 14  
## initial value 2593.837238   
## iter 10 value 1863.834407  
## iter 20 value 1683.293593  
## iter 30 value 1596.141937  
## iter 40 value 1527.145467  
## iter 50 value 1497.944848  
## iter 60 value 1489.105097  
## iter 70 value 1479.588621  
## iter 80 value 1478.815466  
## iter 90 value 1478.588264  
## iter 100 value 1478.054665  
## iter 110 value 1478.009778  
## iter 120 value 1477.910910  
## iter 130 value 1477.835726  
## iter 140 value 1477.822115  
## final value 1477.822076   
## converged  
## # weights: 14  
## initial value 2151.233944   
## iter 10 value 1521.277087  
## iter 20 value 1500.960138  
## iter 30 value 1473.863542  
## iter 40 value 1472.983454  
## iter 50 value 1471.799061  
## iter 60 value 1470.562451  
## iter 70 value 1470.496968  
## iter 80 value 1470.166746  
## iter 90 value 1469.858236  
## iter 100 value 1469.814168  
## final value 1469.811825   
## converged  
## # weights: 14  
## initial value 2377.280407   
## iter 10 value 1722.209249  
## iter 20 value 1585.170006  
## iter 30 value 1532.813774  
## iter 40 value 1519.437463  
## iter 50 value 1509.323697  
## iter 60 value 1507.368192  
## iter 70 value 1507.289372  
## iter 80 value 1507.064914  
## final value 1507.064734   
## converged  
## # weights: 14  
## initial value 2075.581552   
## iter 10 value 1611.779154  
## iter 20 value 1553.537167  
## iter 30 value 1501.139585  
## iter 40 value 1496.648956  
## iter 50 value 1486.248966  
## iter 60 value 1484.398065  
## iter 70 value 1484.371374  
## iter 80 value 1483.839765  
## iter 90 value 1483.735825  
## iter 100 value 1483.698215  
## iter 110 value 1483.538856  
## iter 120 value 1483.455589  
## iter 120 value 1483.455576  
## final value 1483.455426   
## converged  
## # weights: 14  
## initial value 2235.760938   
## iter 10 value 1605.952447  
## iter 20 value 1554.487605  
## iter 30 value 1491.921467  
## iter 40 value 1482.398128  
## iter 50 value 1475.444177  
## iter 60 value 1473.703199  
## iter 70 value 1473.649950  
## iter 80 value 1473.299082  
## iter 90 value 1473.176866  
## final value 1473.176608   
## converged

## Testing the model

pred6 <- predict(ann\_model, test\_data)

## Confusion Matrix

confusionMatrix(pred6, test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 941 173  
## 1 73 162  
##   
## Accuracy : 0.8176   
## 95% CI : (0.796, 0.8379)  
## No Information Rate : 0.7517   
## P-Value [Acc > NIR] : 4.148e-09   
##   
## Kappa : 0.4573   
##   
## Mcnemar's Test P-Value : 2.754e-10   
##   
## Sensitivity : 0.9280   
## Specificity : 0.4836   
## Pos Pred Value : 0.8447   
## Neg Pred Value : 0.6894   
## Prevalence : 0.7517   
## Detection Rate : 0.6976   
## Detection Prevalence : 0.8258   
## Balanced Accuracy : 0.7058   
##   
## 'Positive' Class : 0   
##

# Leave one Out Validation

## Read the data

library(haven)  
bank\_loan\_df <- read\_sav("F:/MDS-Private-Study-Materials/First Semester/Statistical Computing with R/Assignments/Data/P4\_bankloan\_5000\_clients.sav")  
bank\_loan\_df$defaulted\_loan<-as.factor(bank\_loan\_df$defaulted\_loan)  
bank\_loan\_df$education\_level<-as.factor(bank\_loan\_df$education\_level)

# Logistic Regression With LOOCV Validation

## Training Logistic Regression Model

set.seed(1234)  
library(caret)

ind<-sample(2,nrow(bank\_loan\_df),replace=T,prob = c(0.7,0.3))  
 train\_data<-bank\_loan\_df[ind==1,]  
 test\_data<-bank\_loan\_df[ind==2,]

## Setting Up the Train Control

loocv\_train\_control<-trainControl(method = "LOOCV")

# Logistic Regression With LOOCV Validation

## Training Logistic Regression Model

#logistic\_clf1<-train(defaulted\_loan~.,  
 #data=train\_data,  
 #method="glm",  
 #family="binomial",  
 #trControl=loocv\_train\_control,   
 #verbose=F  
# )

# KNN Model with LOOCV validation

## Training KNN Model

knn\_clf1<-train(defaulted\_loan~.,data = train\_data,  
 method="knn",  
 trControl=loocv\_train\_control  
 )

## Obtain the result

knn\_clf1$result

## k Accuracy Kappa  
## 1 5 0.7636879 0.3087625  
## 2 7 0.7707801 0.3112221  
## 3 9 0.7770213 0.3248772

# Confusion Matrix for Model Evaluation

predicted\_val\_knn1<-predict(knn\_clf1,newdata = test\_data)  
confusionMatrix(predicted\_val\_knn1,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1018 226  
## 1 96 135  
##   
## Accuracy : 0.7817   
## 95% CI : (0.7597, 0.8025)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.009238   
##   
## Kappa : 0.3277   
##   
## Mcnemar's Test P-Value : 6.532e-13   
##   
## Sensitivity : 0.9138   
## Specificity : 0.3740   
## Pos Pred Value : 0.8183   
## Neg Pred Value : 0.5844   
## Prevalence : 0.7553   
## Detection Rate : 0.6902   
## Detection Prevalence : 0.8434   
## Balanced Accuracy : 0.6439   
##   
## 'Positive' Class : 0   
##

# Naïve Bayes classifier

## Training the Model

nb\_clf1<-train(defaulted\_loan~.,  
 data=train\_data,  
 method="naive\_bayes",  
 usepoisson = TRUE,  
 trControl=loocv\_train\_control  
 )

# Making Prediction on Test Data

predicted\_val\_nb1<-predict(nb\_clf1,newdata = test\_data)

# Confusion Matrix for Model Evaluation

confusionMatrix(predicted\_val\_nb1,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1094 308  
## 1 20 53  
##   
## Accuracy : 0.7776   
## 95% CI : (0.7555, 0.7986)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.02363   
##   
## Kappa : 0.1764   
##   
## Mcnemar's Test P-Value : < 2e-16   
##   
## Sensitivity : 0.9820   
## Specificity : 0.1468   
## Pos Pred Value : 0.7803   
## Neg Pred Value : 0.7260   
## Prevalence : 0.7553   
## Detection Rate : 0.7417   
## Detection Prevalence : 0.9505   
## Balanced Accuracy : 0.5644   
##   
## 'Positive' Class : 0   
##

# Support Vector Machine (SVM) Model

#ctrl <- trainControl(method = "LOOCV", savePred=T)  
 #svm\_clf1<-train(defaulted\_loan~.,  
 # data=train\_data,  
 # method="svmLinear",  
 # trControl=ctrl,  
 # )  
 #svm\_clf

# Making the Prediction for test data

#predicted\_val\_svm1<-predict(svm\_clf1,newdata = test\_data)

# Confusion Matrix for Model Evaluation

#confusionMatrix(predicted\_val\_svm1,test\_data$defaulted\_loan)

The Model did not Converge to a solution. Leaving it as is for now. # Decision Tree Model

#dtree\_clf1<-train(defaulted\_loan~.,  
#data = train\_data,  
#method="rpart",  
#parms = list(split = "information"),  
#tuneLength=10,  
# trControl=loocv\_train\_control  
 #)  
#dtree\_clf1

# Making the Prediction for test data

#predicted\_val\_dtree1<-predict(dtree\_clf1,newdata = test\_data)

# Confusion Matrix for Model Evaluation

# confusionMatrix(predicted\_val\_dtree1,test\_data$defaulted\_loan)

# Artifical Neural Network (ANN) Model

## Training the Model

#ann\_clf1 <- train(defaulted\_loan ~ ., data = train\_data,   
# method = "nnet",  
 # preProcess = c("center","scale"),   
# maxit = 250, # Maximum number of iterations  
 # tuneGrid = data.frame(size = 1, decay = 0),  
 # tuneGrid = data.frame(size = 0, decay = 0),skip=TRUE, # Technically, this is log-reg  
 # metric = "Accuracy",  
 # trControl=loocv\_train\_control)

## Making the Predictions for Test data

#predicted\_val\_ann1<-predict(ann\_clf1,newdata = test\_data)

## Confusion Matrix for the Model Evaluation

#confusionMatrix(predicted\_val\_ann1,test\_data$defaulted\_loan)

# K-Fold Cross Validation

## Reading the File

library(haven)  
bank\_loan\_df <- read\_sav("F:/MDS-Private-Study-Materials/First Semester/Statistical Computing with R/Assignments/Data/P4\_bankloan\_5000\_clients.sav")

## Changing the data type of variables

bank\_loan\_df$defaulted\_loan<-as.factor(bank\_loan\_df$defaulted\_loan)  
bank\_loan\_df$education\_level<-as.factor(bank\_loan\_df$education\_level)

## Splitting the data into train and test set

set.seed(1234)  
library(caret)  
ind<-sample(2,nrow(bank\_loan\_df),replace=T,prob = c(0.7,0.3))  
train\_data<-bank\_loan\_df[ind==1,]  
test\_data<-bank\_loan\_df[ind==2,]

## Setting Up the Train Control

cv\_train\_control<-trainControl(method = "cv",number = 10)

## Logistic Regression With Cross Validation

## Training Logistic Regression Model

logistic\_clf1<-train(defaulted\_loan~.,  
 data=train\_data,  
 method="glm",  
 family="binomial",  
 trControl=cv\_train\_control  
 )

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

summary(logistic\_clf1)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.6490 -0.6635 -0.3442 0.1409 3.2833   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.235986 0.272446 -4.537 5.72e-06 \*\*\*  
## age 0.006492 0.008297 0.782 0.4339   
## education\_level2 0.227329 0.110244 2.062 0.0392 \*   
## education\_level3 0.260781 0.156468 1.667 0.0956 .   
## education\_level4 0.285038 0.186776 1.526 0.1270   
## education\_level5 0.020994 0.447370 0.047 0.9626   
## current\_employ\_year -0.182777 0.012678 -14.416 < 2e-16 \*\*\*  
## current\_address\_year -0.094317 0.010300 -9.157 < 2e-16 \*\*\*  
## income\_household -0.002470 0.003879 -0.637 0.5244   
## debt\_income\_ratio 0.099652 0.012885 7.734 1.04e-14 \*\*\*  
## credit\_card\_debt 0.425066 0.044558 9.540 < 2e-16 \*\*\*  
## other\_debts 0.006704 0.030495 0.220 0.8260   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 3994.4 on 3524 degrees of freedom  
## Residual deviance: 2850.2 on 3513 degrees of freedom  
## AIC: 2874.2  
##   
## Number of Fisher Scoring iterations: 6

## Making the Prediction

predicted\_val\_log1<-predict(logistic\_clf1,newdata = test\_data)

## Confusion Matrix for Evaluation

confusionMatrix(predicted\_val\_log1,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1038 191  
## 1 76 170  
##   
## Accuracy : 0.819   
## 95% CI : (0.7984, 0.8383)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 2.487e-09   
##   
## Kappa : 0.4513   
##   
## Mcnemar's Test P-Value : 3.022e-12   
##   
## Sensitivity : 0.9318   
## Specificity : 0.4709   
## Pos Pred Value : 0.8446   
## Neg Pred Value : 0.6911   
## Prevalence : 0.7553   
## Detection Rate : 0.7037   
## Detection Prevalence : 0.8332   
## Balanced Accuracy : 0.7013   
##   
## 'Positive' Class : 0   
##

## KNN Model with Cross validation

## Training KNN Model

knn\_clf1<-train(defaulted\_loan~.,data = train\_data,  
 method="knn",  
 trControl=cv\_train\_control  
 )

## Getting the Result of the Model

knn\_clf1$result

## k Accuracy Kappa AccuracySD KappaSD  
## 1 5 0.7668056 0.3138335 0.01709039 0.03827953  
## 2 7 0.7727611 0.3210782 0.01581367 0.03762291  
## 3 9 0.7744568 0.3184971 0.01934934 0.05507607

## Confusion Matrix for Model Evaluation

predicted\_val\_knn1<-predict(knn\_clf1,newdata = test\_data)  
confusionMatrix(predicted\_val\_knn1,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1019 226  
## 1 95 135  
##   
## Accuracy : 0.7824   
## 95% CI : (0.7604, 0.8032)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.007801   
##   
## Kappa : 0.329   
##   
## Mcnemar's Test P-Value : 3.99e-13   
##   
## Sensitivity : 0.9147   
## Specificity : 0.3740   
## Pos Pred Value : 0.8185   
## Neg Pred Value : 0.5870   
## Prevalence : 0.7553   
## Detection Rate : 0.6908   
## Detection Prevalence : 0.8441   
## Balanced Accuracy : 0.6443   
##   
## 'Positive' Class : 0   
##

# Naïve Bayes classifier

## Training the Model

library(naivebayes)

## Warning: package 'naivebayes' was built under R version 4.1.2

## naivebayes 0.9.7 loaded

nb\_clf1<-train(defaulted\_loan~.,  
 data=train\_data,  
 method="naive\_bayes",  
 usepoisson = TRUE,  
 trControl=cv\_train\_control  
 )  
summary(nb\_clf1)

##   
## ================================== Naive Bayes ==================================   
##   
## - Call: naive\_bayes.default(x = x, y = y, laplace = param$laplace, usekernel = TRUE, usepoisson = TRUE, adjust = param$adjust)   
## - Laplace: 0   
## - Classes: 2   
## - Samples: 3525   
## - Features: 11   
## - Conditional distributions:   
## - KDE: 11  
## - Prior probabilities:   
## - 0: 0.7461  
## - 1: 0.2539  
##   
## ---------------------------------------------------------------------------------

## Making Prediction on Test Data

predicted\_val\_nb1<-predict(nb\_clf1,newdata = test\_data)

## Confusion Matrix for Model Evaluation

confusionMatrix(predicted\_val\_nb1,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1094 308  
## 1 20 53  
##   
## Accuracy : 0.7776   
## 95% CI : (0.7555, 0.7986)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.02363   
##   
## Kappa : 0.1764   
##   
## Mcnemar's Test P-Value : < 2e-16   
##   
## Sensitivity : 0.9820   
## Specificity : 0.1468   
## Pos Pred Value : 0.7803   
## Neg Pred Value : 0.7260   
## Prevalence : 0.7553   
## Detection Rate : 0.7417   
## Detection Prevalence : 0.9505   
## Balanced Accuracy : 0.5644   
##   
## 'Positive' Class : 0   
##

# Support Vector Machine (SVM) Model

## Training the Model

#svm\_clf1<-train(defaulted\_loan~.,  
#data=train\_data,  
#method="svmLinear",  
#trControl=cv\_train\_control,  
 #)  
#svm\_clf1

# Making the Prediction for test data

# predicted\_val\_svm1<-predict(svm\_clf1,newdata = test\_data)

# Confusion Matrix for Model Evaluation

# confusionMatrix(predicted\_val\_svm1,test\_data$defaulted\_loan)

# Decision Tree Model

#dtree\_clf1<-train(defaulted\_loan~.,  
#data = train\_data,  
#method="rpart",  
#parms = list(split = "information"),  
#tuneLength=10,  
#trControl=cv\_train\_control  
 #)  
 #dtree\_clf1

# Making the Prediction for test data

#predicted\_val\_dtree1<-predict(dtree\_clf1,newdata = test\_data)

# Confusion Matrix for Model Evaluation

#confusionMatrix(predicted\_val\_dtree1,test\_data$defaulted\_loan)

# Artifical Neural Network (ANN) Model

## Training the Model

#ann\_clf1 <- train(defaulted\_loan ~ ., data = train\_data,   
#method = "nnet",  
# preProcess = c("center","scale"),   
#maxit = 250, # Maximum number of iterations  
 #tuneGrid = data.frame(size = 1, decay = 0),  
 # tuneGrid = data.frame(size = 0, decay = 0),skip=TRUE, # Technically, this is log-reg  
#metric = "Accuracy",  
 #trControl=cv\_train\_control)

# Making the Predictions for Test data

# predicted\_val\_ann1<-predict(ann\_clf1,newdata = test\_data)

# Confusion Matrix for the Model Evaluation

# confusionMatrix(predicted\_val\_ann1,test\_data$defaulted\_loan)

# Repeated K-Fold Cross Validation

# Setting Up the Train Control

#rep\_cv\_train\_control<-trainControl(method = "repeatedcv",number = 10,repeats = 3)

## Logistic Regression With Repeated Cross Validation

## Training Logistic Regression Model

#logistic\_clf1<-train(defaulted\_loan~.,  
# data=train\_data,  
# method="glm",  
 #family="binomial",  
 #trControl=rep\_cv\_train\_control  
 #)  
 #summary(logistic\_clf1)

## Making the Prediction

#predicted\_val\_log1<-predict(logistic\_clf1,newdata = test\_data)

## Confusion Matrix for Evaluation

#confusionMatrix(predicted\_val\_log1,test\_data$defaulted\_loan)

## KNN Model with Repeated Cross validation

## Training KNN Model

#knn\_clf1<-train(defaulted\_loan~.,data = train\_data,  
# method="knn",  
 #trControl=rep\_cv\_train\_control  
 #)

## Getting the Result of the Model

#knn\_clf1$result

## Confusion Matrix for Model Evaluation

#predicted\_val\_knn1<-predict(knn\_clf1,newdata = test\_data)  
#confusionMatrix(predicted\_val\_knn1,test\_data$defaulted\_loan)

## Naïve Bayes classifier

## Training the Model

#library(naivebayes)

#nb\_clf1<-train(defaulted\_loan~.,  
 #data=train\_data,  
 #method="naive\_bayes",  
 #usepoisson = TRUE,  
 #trControl=rep\_cv\_train\_control  
 #)  
#summary(nb\_clf1)

## Making Prediction on Test Data

#predicted\_val\_nb1<-predict(nb\_clf1,newdata = test\_data)

## Confusion Matrix for Model Evaluation

#confusionMatrix(predicted\_val\_nb1,test\_data$defaulted\_loan)

## Support Vector Machine (SVM) Model

## Training the model

#svm\_clf1<-train(defaulted\_loan~.,  
 #data=train\_data,  
 #method="svmLinear",  
 #trControl=rep\_cv\_train\_control,  
 #)  
#svm\_clf1

## Making the Prediction for test data

#predicted\_val\_svm1<-predict(svm\_clf1,newdata = test\_data)

## Confusion Matrix for Model Evaluation

#confusionMatrix(predicted\_val\_svm1,test\_data$defaulted\_loan)

## Decision Tree Model

#dtree\_clf1<-train(defaulted\_loan~.,  
 #data = train\_data,  
 #method="rpart",  
 #parms = list(split = "information"),  
 #tuneLength=10,  
 #trControl=rep\_cv\_train\_control  
 #)  
#dtree\_clf1

## Making the Prediction for test data

#predicted\_val\_dtree1<-predict(dtree\_clf1,newdata = test\_data)

## Confusion Matrix for Model Evaluation

# confusionMatrix(predicted\_val\_dtree1,test\_data$defaulted\_loan)

## Artifical Neural Network (ANN) Model

## Training the Model

#ann\_clf1 <- train(defaulted\_loan ~ ., data = train\_data,   
#method = "nnet",  
#preProcess = c("center","scale"),   
#maxit = 250, # Maximum number of iterations  
# tuneGrid = data.frame(size = 1, decay = 0),  
 # tuneGrid = data.frame(size = 0, decay = 0),skip=TRUE, # Technically, this is log-reg  
 #metric = "Accuracy",  
 #trControl=rep\_cv\_train\_control)

## Making the Predictions for Test data

#predicted\_val\_ann1<-predict(ann\_clf1,newdata = test\_data)

## Confusion Matrix for the Model Evaluation

#confusionMatrix(predicted\_val\_ann1,test\_data$defaulted\_loan)

## Bagging, Boosting and Random Forest

## Reading the File

library(haven)  
bank\_loan\_df <- read\_sav("F:/MDS-Private-Study-Materials/First Semester/Statistical Computing with R/Assignments/Data/P4\_bankloan\_5000\_clients.sav")  
bank\_loan\_df$defaulted\_loan<-as.factor(bank\_loan\_df$defaulted\_loan)  
bank\_loan\_df$education\_level<-as.factor(bank\_loan\_df$education\_level)

## Splitting the data into train and test set

set.seed(1234)  
 library(caret)  
## Loading required package: ggplot2  
## Loading required package: lattice  
ind<-sample(2,nrow(bank\_loan\_df),replace=T,prob = c(0.7,0.3))  
 train\_data<-bank\_loan\_df[ind==1,]  
 test\_data<-bank\_loan\_df[ind==2,]

## Bagging Model

## Training the Model

library("ipred")

## Warning: package 'ipred' was built under R version 4.1.2

bag\_dtree\_clf<-bagging(defaulted\_loan~.,  
 data = train\_data,  
 coob=T  
 )  
print(bag\_dtree\_clf)

##   
## Bagging classification trees with 25 bootstrap replications   
##   
## Call: bagging.data.frame(formula = defaulted\_loan ~ ., data = train\_data,   
## coob = T)  
##   
## Out-of-bag estimate of misclassification error: 0.2295

## Making the Prediction

predicted\_bag\_tree<-predict(bag\_dtree\_clf,newdata = test\_data)  
library(caret)  
 confusionMatrix(predicted\_bag\_tree,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 991 191  
## 1 123 170  
##   
## Accuracy : 0.7871   
## 95% CI : (0.7653, 0.8078)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.0021549   
##   
## Kappa : 0.385   
##   
## Mcnemar's Test P-Value : 0.0001562   
##   
## Sensitivity : 0.8896   
## Specificity : 0.4709   
## Pos Pred Value : 0.8384   
## Neg Pred Value : 0.5802   
## Prevalence : 0.7553   
## Detection Rate : 0.6719   
## Detection Prevalence : 0.8014   
## Balanced Accuracy : 0.6803   
##   
## 'Positive' Class : 0   
##

## Random Forest Model

## Training the Model

set.seed(1234)  
library(randomForest)

## Warning: package 'randomForest' was built under R version 4.1.2

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

rf\_clf<-randomForest(defaulted\_loan~.,  
 data = train\_data)  
rf\_clf

##   
## Call:  
## randomForest(formula = defaulted\_loan ~ ., data = train\_data)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 20.88%  
## Confusion matrix:  
## 0 1 class.error  
## 0 2420 210 0.07984791  
## 1 526 369 0.58770950

## Making the Prediction

predicted\_rf<-predict(rf\_clf,newdata = test\_data)  
confusionMatrix(predicted\_rf,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1023 197  
## 1 91 164  
##   
## Accuracy : 0.8047   
## 95% CI : (0.7836, 0.8247)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 3.459e-06   
##   
## Kappa : 0.4137   
##   
## Mcnemar's Test P-Value : 6.125e-10   
##   
## Sensitivity : 0.9183   
## Specificity : 0.4543   
## Pos Pred Value : 0.8385   
## Neg Pred Value : 0.6431   
## Prevalence : 0.7553   
## Detection Rate : 0.6936   
## Detection Prevalence : 0.8271   
## Balanced Accuracy : 0.6863   
##   
## 'Positive' Class : 0   
##

## Extreme Gradient Boosting

## Training the Model

#xglm\_clf<-train(defaulted\_loan~.,  
 #data = train\_data,  
 #method="xgbTree",  
 #verbose=F  
# )

## Making the Prediction

#predicted\_xgb<-predict(xglm\_clf,newdata = test\_data)  
#confusionMatrix(predicted\_xgb,test\_data$defaulted\_loan)