Use the attached SPSS data containing 5000 cases of bank clients and fit all the supervised classifiers models as described below. Copy all the outputs you will get at/from R/R Studio here and write interpretation for each of the outputs in this file.

Use “defaulted\_loan” variable as dependent variable and use other variables as independent variables to predict by:

1. Fitting logistic regression classifier, KNN classifier, ANN-MLP classifier, Naïve Bayes classifier, SVM classifier and Decision Tree classifier with train/test validation sets, leave-one-out cross validation sets, 10-fold cross validation sets and 3 times repeated 10-fold cross validation sets for this problem and select the best predictive model and find the important variables (variable importance) with relative frequencies and its graph for that model
2. Fitting bagging, random forest and extreme gradient boosting tree models and select the best predictive model for train/test validation sets and find the variable importance with table and graph for that model
3. Who will get the loan as per the best models identified above? Why? Explain with justifications.

**ATTACH THE SCRIPT FILE USED FOR THIS WORK HERE. DO NOT ATTACH ANYTHING ELSE.**

SOLUTION:

QN1:

**Loading the Files**

library(haven)  
 bank\_loan\_df <- read\_sav("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 × 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" ...

# **Train Test Validation**

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

## **Splitting the data into traning and testing set**

set.seed(1234)  
 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,]

## **Logistic Regression With train/test Validation**

### **Training Logistic Regression Model**

logistic\_clf<-train(defaulted\_loan~.,  
 data=train\_data,  
 method="glm",  
 family="binomial"  
 )  
 summary(logistic\_clf)

##   
## 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\_log<-predict(logistic\_clf,newdata = test\_data)

### **Confusion Matrix for Evaluation**

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

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1038 76  
 ## 1 191 170  
 ##   
## Accuracy : 0.819   
## 95% CI : (0.7984, 0.8383)  
 ## No Information Rate : 0.8332   
## P-Value [Acc > NIR] : 0.9322   
##   
## Kappa : 0.4513   
##   
## Mcnemar's Test P-Value : 3.022e-12   
##   
## Sensitivity : 0.8446   
## Specificity : 0.6911   
## Pos Pred Value : 0.9318   
## Neg Pred Value : 0.4709   
## Prevalence : 0.8332   
## Detection Rate : 0.7037   
## Detection Prevalence : 0.7553   
## Balanced Accuracy : 0.7678   
##   
## 'Positive' Class : 0   
##

## **KNN Model with train/test validation**

### **Training KNN Model**

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

### **Getting the Result of the Model**

knn\_clf$result

## k Accuracy Kappa AccuracySD KappaSD  
 ## 1 5 0.7475369 0.2958532 0.010849610 0.02884625  
 ## 2 7 0.7582432 0.3117892 0.008841404 0.02356125  
 ## 3 9 0.7674740 0.3267350 0.008578768 0.02333298  
 ## 4 11 0.7701279 0.3240461 0.009131611 0.02925444  
 ## 5 13 0.7743100 0.3300675 0.009833257 0.02766763  
 ## 6 15 0.7780591 0.3362558 0.008906519 0.02560595  
 ## 7 17 0.7810133 0.3395283 0.011387056 0.03375783  
 ## 8 19 0.7834197 0.3443761 0.010369322 0.02759248  
 ## 9 21 0.7832864 0.3399306 0.008661037 0.02399463  
 ## 10 23 0.7843194 0.3395441 0.010371607 0.02569968

### **Confusion Matrix for Model Evaluation**

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

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1044 70  
 ## 1 234 127  
 ##   
## Accuracy : 0.7939   
## 95% CI : (0.7723, 0.8143)  
 ## No Information Rate : 0.8664   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3414   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8169   
## Specificity : 0.6447   
## Pos Pred Value : 0.9372   
## Neg Pred Value : 0.3518   
## Prevalence : 0.8664   
## Detection Rate : 0.7078   
## Detection Prevalence : 0.7553   
## Balanced Accuracy : 0.7308   
##   
## 'Positive' Class : 0   
##

## **Naïve Bayes classifier**

### **Training the Naïve Bayes classifier**

library(e1071)  
 nb\_clf<-naiveBayes(defaulted\_loan~.,data=train\_data)

### **Summary of the Model**

(nb\_clf)

##   
## Naive Bayes Classifier for Discrete Predictors  
 ##   
## Call:  
 ## naiveBayes.default(x = X, y = Y, laplace = laplace)  
 ##   
## A-priori probabilities:  
 ## Y  
 ## 0 1   
## 0.7460993 0.2539007   
##   
## Conditional probabilities:  
 ## age  
 ## Y [,1] [,2]  
 ## 0 36.22966 7.836511  
 ## 1 32.79218 7.690558  
 ##   
## education\_level  
 ## Y 1 2 3 4 5  
 ## 0 0.566539924 0.268821293 0.100380228 0.055513308 0.008745247  
 ## 1 0.442458101 0.294972067 0.144134078 0.101675978 0.016759777  
 ##   
## current\_employ\_year  
 ## Y [,1] [,2]  
 ## 0 9.775665 7.340640  
 ## 1 5.426816 5.490989  
 ##   
## current\_address\_year  
 ## Y [,1] [,2]  
 ## 0 9.001141 7.038583  
 ## 1 5.698324 5.384183  
 ##   
## income\_household  
 ## Y [,1] [,2]  
 ## 0 48.73426 39.55991  
 ## 1 44.26358 88.72586  
 ##   
## debt\_income\_ratio  
 ## Y [,1] [,2]  
 ## 0 8.59219 5.526728  
 ## 1 14.56893 7.947622  
 ##   
## credit\_card\_debt  
 ## Y [,1] [,2]  
 ## 0 1.286985 1.583750  
 ## 1 2.661458 5.948471  
 ##   
## other\_debts  
 ## Y [,1] [,2]  
 ## 0 2.881188 3.38933  
 ## 1 4.271832 14.45939

### **Making the Prediction in the test data**

predicted\_val\_nb<-predict(nb\_clf,newdata = test\_data)

### **Confusion Matrix for Evaluation**

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

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1071 285  
 ## 1 43 76  
 ##   
## Accuracy : 0.7776   
## 95% CI : (0.7555, 0.7986)  
 ## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.02363   
##   
## Kappa : 0.2223   
##   
## Mcnemar's Test P-Value : < 2e-16   
##   
## Sensitivity : 0.9614   
## Specificity : 0.2105   
## Pos Pred Value : 0.7898   
## Neg Pred Value : 0.6387   
## Prevalence : 0.7553   
## Detection Rate : 0.7261   
## Detection Prevalence : 0.9193   
## Balanced Accuracy : 0.5860   
##   
## 'Positive' Class : 0   
##

## **Support Vector Machine (SVM) Model**

### **Training the Model**

svm\_clf<-train(defaulted\_loan~.,  
 data=train\_data,  
 method="svmLinear"  
 )  
 svm\_clf

## Support Vector Machines with Linear Kernel   
##   
## 3525 samples  
 ## 8 predictor  
 ## 2 classes: '0', '1'   
##   
## No pre-processing  
 ## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 3525, 3525, 3525, 3525, 3525, 3525, ...   
## Resampling results:  
 ##   
## Accuracy Kappa   
## 0.8021072 0.3860155  
 ##   
## Tuning parameter 'C' was held constant at a value of 1

### **Making the Prediction for test data**

predicted\_val\_svm<-predict(svm\_clf,newdata = test\_data)

### **Confusion Matrix for Model Evaluation**

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

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1055 218  
 ## 1 59 143  
 ##   
## Accuracy : 0.8122   
## 95% CI : (0.7913, 0.8318)  
 ## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 9.898e-08   
##   
## Kappa : 0.4032   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9470   
## Specificity : 0.3961   
## Pos Pred Value : 0.8288   
## Neg Pred Value : 0.7079   
## Prevalence : 0.7553   
## Detection Rate : 0.7153   
## Detection Prevalence : 0.8631   
## Balanced Accuracy : 0.6716   
##   
## 'Positive' Class : 0   
##

## **Decision Tree Model**

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

## CART   
##   
## 3525 samples  
 ## 8 predictor  
 ## 2 classes: '0', '1'   
##   
## No pre-processing  
 ## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 3525, 3525, 3525, 3525, 3525, 3525, ...   
## Resampling results across tuning parameters:  
 ##   
## cp Accuracy Kappa   
## 0.002793296 0.7628193 0.3313959  
 ## 0.002979516 0.7639354 0.3327615  
 ## 0.003072626 0.7641848 0.3335925  
 ## 0.003351955 0.7654893 0.3365711  
 ## 0.004469274 0.7724129 0.3390493  
 ## 0.005586592 0.7756679 0.3393153  
 ## 0.006703911 0.7765469 0.3352378  
 ## 0.024581006 0.7714426 0.3095524  
 ## 0.027374302 0.7698968 0.3065844  
 ## 0.060335196 0.7588103 0.2140312  
 ##   
## Accuracy was used to select the optimal model using the largest value.  
 ## The final value used for the model was cp = 0.006703911.

### **Making the Prediction for test data**

predicted\_val\_dtree<-predict(dtree\_clf,newdata = test\_data)

### **Confusion Matrix for Model Evaluation**

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

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1024 214  
 ## 1 90 147  
 ##   
## Accuracy : 0.7939   
## 95% CI : (0.7723, 0.8143)  
 ## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.0002488   
##   
## Kappa : 0.3693   
##   
## Mcnemar's Test P-Value : 1.732e-12   
##   
## Sensitivity : 0.9192   
## Specificity : 0.4072   
## Pos Pred Value : 0.8271   
## Neg Pred Value : 0.6203   
## Prevalence : 0.7553   
## Detection Rate : 0.6942   
## Detection Prevalence : 0.8393   
## Balanced Accuracy : 0.6632   
##   
## 'Positive' Class : 0   
##

## **Artifical Neural Network (ANN) Model**

### **Training the Model**

ann\_clf <- 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 2613.046683   
## iter 10 value 1718.710496  
 ## iter 20 value 1592.438604  
 ## iter 30 value 1536.737176  
 ## iter 40 value 1526.182944  
 ## iter 50 value 1523.444206  
 ## final value 1523.436772   
## converged  
 ..............

### **Making the Predictions for Test data**

predicted\_val\_ann<-predict(ann\_clf,newdata = test\_data)

### **Confusion Matrix for the Model Evaluation**

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

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1036 191  
 ## 1 78 170  
 ##   
## Accuracy : 0.8176   
## 95% CI : (0.797, 0.837)  
 ## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 5.382e-09   
##   
## Kappa : 0.4483   
##   
## Mcnemar's Test P-Value : 8.565e-12   
##   
## Sensitivity : 0.9300   
## Specificity : 0.4709   
## Pos Pred Value : 0.8443   
## Neg Pred Value : 0.6855   
## Prevalence : 0.7553   
## Detection Rate : 0.7024   
## Detection Prevalence : 0.8319   
## Balanced Accuracy : 0.7004   
##   
## 'Positive' Class : 0   
##

# **Leave one Out Validation**

## **Reading the File**

library(haven)  
 bank\_loan\_df <- read\_sav("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)

## 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,]

## **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  
 )  
 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 LOOCV validation**

### **Training KNN Model**

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

### **Getting the Result of the Model**

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**

library(naivebayes)

## naivebayes 0.9.7 loaded

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

*#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

## CART   
##   
## 3525 samples  
 ## 8 predictor  
 ## 2 classes: '0', '1'   
##   
## No pre-processing  
 ## Resampling: Leave-One-Out Cross-Validation   
## Summary of sample sizes: 3524, 3524, 3524, 3524, 3524, 3524, ...   
## Resampling results across tuning parameters:  
 ##   
## cp Accuracy Kappa   
## 0.002793296 0.7926241 0.3538152  
 ## 0.002979516 0.7863830 0.3320428  
 ## 0.003072626 0.7852482 0.3267308  
 ## 0.003351955 0.7900709 0.3357440  
 ## 0.004469274 0.7690780 0.2966642  
 ## 0.005586592 0.7804255 0.3451509  
 ## 0.006703911 0.7790071 0.3422901  
 ## 0.024581006 0.7880851 0.3481796  
 ## 0.027374302 0.7602837 0.2924469  
 ## 0.060335196 0.6669504 -0.1372405  
 ##   
## Accuracy was used to select the optimal model using the largest value.  
 ## The final value used for the model was cp = 0.002793296.

### **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)

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1037 235  
 ## 1 77 126  
 ##   
## Accuracy : 0.7885   
## 95% CI : (0.7667, 0.8091)  
 ## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.001443   
##   
## Kappa : 0.3285   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9309   
## Specificity : 0.3490   
## Pos Pred Value : 0.8153   
## Neg Pred Value : 0.6207   
## Prevalence : 0.7553   
## Detection Rate : 0.7031   
## Detection Prevalence : 0.8624   
## Balanced Accuracy : 0.6400   
##   
## 'Positive' Class : 0   
##

## **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)*

The ANN Also Crashed the R Session for Multiple time so we discard this model for now.

# **K-Fold Cross Validation**

## **Reading the File**

library(haven)  
 bank\_loan\_df <- read\_sav("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)

## 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,]

## **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  
 )  
 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)

## 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

## Support Vector Machines with Linear Kernel   
##   
## 3525 samples  
 ## 8 predictor  
 ## 2 classes: '0', '1'   
##   
## No pre-processing  
 ## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 3173, 3172, 3173, 3173, 3173, 3172, ...   
## Resampling results:  
 ##   
## Accuracy Kappa   
## 0.8033994 0.3865814  
 ##   
## Tuning parameter 'C' was held constant at a value of 1

### **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)

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1055 218  
 ## 1 59 143  
 ##   
## Accuracy : 0.8122   
## 95% CI : (0.7913, 0.8318)  
 ## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 9.898e-08   
##   
## Kappa : 0.4032   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9470   
## Specificity : 0.3961   
## Pos Pred Value : 0.8288   
## Neg Pred Value : 0.7079   
## Prevalence : 0.7553   
## Detection Rate : 0.7153   
## Detection Prevalence : 0.8631   
## Balanced Accuracy : 0.6716   
##   
## 'Positive' Class : 0   
##

## **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

## CART   
##   
## 3525 samples  
 ## 8 predictor  
 ## 2 classes: '0', '1'   
##   
## No pre-processing  
 ## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 3172, 3172, 3173, 3173, 3173, 3172, ...   
## Resampling results across tuning parameters:  
 ##   
## cp Accuracy Kappa   
## 0.002793296 0.7790191 0.3351323  
 ## 0.002979516 0.7798698 0.3382538  
 ## 0.003072626 0.7798698 0.3382538  
 ## 0.003351955 0.7838374 0.3440170  
 ## 0.004469274 0.7832684 0.3427673  
 ## 0.005586592 0.7841183 0.3451674  
 ## 0.006703911 0.7844024 0.3445264  
 ## 0.024581006 0.7753267 0.3038560  
 ## 0.027374302 0.7707877 0.3028661  
 ## 0.060335196 0.7560303 0.1766183  
 ##   
## Accuracy was used to select the optimal model using the largest value.  
 ## The final value used for the model was cp = 0.006703911.

### **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)

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1024 214  
 ## 1 90 147  
 ##   
## Accuracy : 0.7939   
## 95% CI : (0.7723, 0.8143)  
 ## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.0002488   
##   
## Kappa : 0.3693   
##   
## Mcnemar's Test P-Value : 1.732e-12   
##   
## Sensitivity : 0.9192   
## Specificity : 0.4072   
## Pos Pred Value : 0.8271   
## Neg Pred Value : 0.6203   
## Prevalence : 0.7553   
## Detection Rate : 0.6942   
## Detection Prevalence : 0.8393   
## Balanced Accuracy : 0.6632   
##   
## 'Positive' Class : 0   
##

## **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)

## # weights: 14  
 ## initial value 3067.836186   
## iter 10 value 1477.031599  
 ## iter 20 value 1372.641152  
 ## iter 30 value 1319.035272  
 ## iter 40 value 1312.295634  
 ## iter 50 value 1306.681402  
 ## iter 60 value 1305.656208  
 ## iter 70 value 1305.627085  
 ## iter 80 value 1305.322532  
 ## iter 90 value 1305.240048  
 ## final value 1305.238766   
## converged  
 ## # weights: 14  
 ## initial value 1870.856482   
## iter 10 value 1411.886341  
 ## iter 20 value 1373.136030  
 ## iter 30 value 1313.845701  
 ## iter 40 value 1296.538758  
 ## iter 50 value 1283.919622  
 ## iter 60 value 1280.768395  
 ## iter 70 value 1280.693740  
 ## iter 80 value 1279.920838  
 ## iter 90 value 1279.731647  
 ## iter 100 value 1279.729881  
 ## iter 110 value 1279.602928  
 ## iter 120 value 1279.562852  
 ## final value 1279.562718   
…..........................

**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)

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1036 191  
 ## 1 78 170  
 ##   
## Accuracy : 0.8176   
## 95% CI : (0.797, 0.837)  
 ## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 5.382e-09   
##   
## Kappa : 0.4483   
##   
## Mcnemar's Test P-Value : 8.565e-12   
##   
## Sensitivity : 0.9300   
## Specificity : 0.4709   
## Pos Pred Value : 0.8443   
## Neg Pred Value : 0.6855   
## Prevalence : 0.7553   
## Detection Rate : 0.7024   
## Detection Prevalence : 0.8319   
## Balanced Accuracy : 0.7004   
##   
## 'Positive' Class : 0   
##

# **Repeated K-Fold Cross Validation**

## **Reading the File**

library(haven)  
 bank\_loan\_df <- read\_sav("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)

## 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,]

## **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)

##   
## 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 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

## k Accuracy Kappa AccuracySD KappaSD  
 ## 1 5 0.7651051 0.3108629 0.01793387 0.04889046  
 ## 2 7 0.7734288 0.3193413 0.01600585 0.04842754  
 ## 3 9 0.7749418 0.3174315 0.01861342 0.05753687

### **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)

## naivebayes 0.9.7 loaded

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

svm\_clf1

## Support Vector Machines with Linear Kernel   
##   
## 3525 samples  
 ## 8 predictor  
 ## 2 classes: '0', '1'   
##   
## No pre-processing  
 ## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3172, 3172, 3173, 3172, 3172, 3173, ...   
## Resampling results:  
 ##   
## Accuracy Kappa   
## 0.803591 0.3896318  
 ##   
## Tuning parameter 'C' was held constant at a value of 1

### **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)

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1055 218  
 ## 1 59 143  
 ##   
## Accuracy : 0.8122   
## 95% CI : (0.7913, 0.8318)  
 ## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 9.898e-08   
##   
## Kappa : 0.4032   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9470   
## Specificity : 0.3961   
## Pos Pred Value : 0.8288   
## Neg Pred Value : 0.7079   
## Prevalence : 0.7553   
## Detection Rate : 0.7153   
## Detection Prevalence : 0.8631   
## Balanced Accuracy : 0.6716   
##   
## 'Positive' Class : 0   
##

## **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

## CART   
##   
## 3525 samples  
 ## 8 predictor  
 ## 2 classes: '0', '1'   
##   
## No pre-processing  
 ## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3172, 3173, 3173, 3172, 3172, 3173, ...   
## Resampling results across tuning parameters:  
 ##   
## cp Accuracy Kappa   
## 0.002793296 0.7823196 0.3521905  
 ## 0.002979516 0.7835485 0.3489152  
 ## 0.003072626 0.7840206 0.3495299  
 ## 0.003351955 0.7835466 0.3399202  
 ## 0.004469274 0.7850591 0.3386658  
 ## 0.005586592 0.7848705 0.3342402  
 ## 0.006703911 0.7819413 0.3186385  
 ## 0.024581006 0.7692707 0.2886086  
 ## 0.027374302 0.7677572 0.2879412  
 ## 0.060335196 0.7572640 0.2116461  
 ##   
## Accuracy was used to select the optimal model using the largest value.  
 ## The final value used for the model was cp = 0.004469274.

### **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)

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1022 209  
 ## 1 92 152  
 ##   
## Accuracy : 0.7959   
## 95% CI : (0.7744, 0.8162)  
 ## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.0001207   
##   
## Kappa : 0.3801   
##   
## Mcnemar's Test P-Value : 2.292e-11   
##   
## Sensitivity : 0.9174   
## Specificity : 0.4211   
## Pos Pred Value : 0.8302   
## Neg Pred Value : 0.6230   
## Prevalence : 0.7553   
## Detection Rate : 0.6929   
## Detection Prevalence : 0.8346   
## Balanced Accuracy : 0.6692   
##   
## 'Positive' Class : 0   
##

## **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)

## # weights: 14  
 ## initial value 2023.149797   
## iter 10 value 1396.519266  
 ## iter 20 value 1339.721019  
 ## iter 30 value 1291.143105  
 ## iter 40 value 1285.932746  
 ## iter 50 value 1279.022143  
 ## iter 60 value 1277.575386  
 ## iter 70 value 1277.538891  
 ## iter 80 value 1277.170176  
 ## final value 1277.080129   
## converged  
 ## # weights: 14  
 ## initial value 2075.998442

### **Making the Predictions for Test data**

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

### **Confusion Matrix for the Model Evaluation**

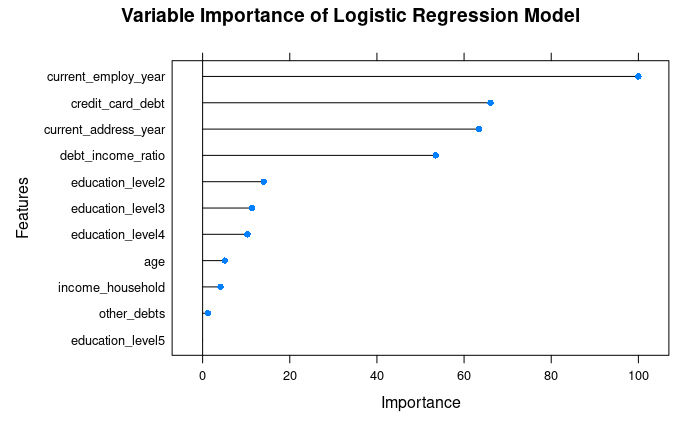
confusionMatrix(predicted\_val\_ann1,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1015 178  
 ## 1 99 183  
 ##   
## Accuracy : 0.8122   
## 95% CI : (0.7913, 0.8318)  
 ## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 9.898e-08   
##   
## Kappa : 0.4514   
##   
## Mcnemar's Test P-Value : 2.778e-06   
##   
## Sensitivity : 0.9111   
## Specificity : 0.5069   
## Pos Pred Value : 0.8508   
## Neg Pred Value : 0.6489   
## Prevalence : 0.7553   
## Detection Rate : 0.6881   
## Detection Prevalence : 0.8088   
## Balanced Accuracy : 0.7090   
##   
## 'Positive' Class : 0   
##

The table below shows accuracy, Sensitivity and Specificity of all the models with different validation methods

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.N** | **Model Name** | **Validation** | **Accuracy** | **Sensitivity** | **Specificity** |
| 1 | Logistic Regression | Train/Test | 0.819 | 0.8446 | 0.6911 |
| 2 | KNN | Train/Test | 0.7939 | 0.8169 | 0.6447 |
| 3 | Naive Bayes Classifier | Train/Test | 0.776 | 0.9614 | 0.2105 |
| 4 | SVM | Train/Test | 0.8122 | 0.947 | 0.3961 |
| 5 | Decision Tree | Train/Test | 0.7939 | 0.9192 | 0.4072 |
| 6 | ANN | Train/Test | 0.8176 | 0.93 | 0.4709 |
| 7 | Logistic Regression | LOOCV | 0.819 | 0.9318 | 0.4709 |
| 8 | KNN | LOOCV | 0.7817 | 0.93138 | 0.374 |
| 9 | Naive Bayes Classifier | LOOCV | 0.7776 | 0.982 | 0.1468 |
| 10 | SVM | LOOCV | 0 | 0 | 0 |
| 11 | Decision Tree | LOOCV | 0.7885 | 0.9309 | 0.349 |
| 12 | ANN | LOOCV | 0 | 0 | 0 |
| 13 | Logistic Regression | K-Fold CV | 0.819 | 0.9318 | 0.4709 |
| 14 | KNN | K-Fold CV | 0.7824 | 0.9147 | 0.374 |
| 15 | Naive Bayes Classifier | K-Fold CV | 0.7776 | 0.982 | 0.1468 |
| 16 | SVM | K-Fold CV | 0.8122 | 0.947 | 0.3961 |
| 17 | Decision Tree | K-Fold CV | 0.7939 | 0.9192 | 0.4072 |
| 18 | ANN | K-Fold CV | 0.8176 | 0.93 | 0.4709 |
| 19 | Logistic Regression | Repeated K-Fold | 0.819 | 0.9318 | 0.4709 |
| 20 | KNN | Repeated K-Fold | 0.7824 | 0.9147 | 0.374 |
| 21 | Naive Bayes Classifier | Repeated K-Fold | 0.7776 | 0.982 | 0.1468 |
| 22 | SVM | Repeated K-Fold | 0.8122 | 0.947 | 0.3961 |
| 23 | Decision Tree | Repeated K-Fold | 0.7959 | 0.9174 | 0.4211 |
| 24 | ANN | Repeated K-Fold | 0.8122 | 0.9111 | 0.5069 |

From the table above we can see the Logistic regression model has highest value of accuracy. So, we choose logistic regression as the best model. Now, among the logistic regression model the value of sensitivity is same so I would go with K-Fold Cross Validation model.



QN2:

# **Bagging, Boosting and Random Forest**

## **Reading the File**

library(haven)  
 bank\_loan\_df <- read\_sav("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)

## 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")  
 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)

## 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)

## Confusion Matrix and Statistics  
 ##   
## Reference  
 ## Prediction 0 1  
 ## 0 1039 199  
 ## 1 75 162  
 ##   
## Accuracy : 0.8142   
## 95% CI : (0.7934, 0.8338)  
 ## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 3.432e-08   
##   
## Kappa : 0.4315   
##   
## Mcnemar's Test P-Value : 1.080e-13   
##   
## Sensitivity : 0.9327   
## Specificity : 0.4488   
## Pos Pred Value : 0.8393   
## Neg Pred Value : 0.6835   
## Prevalence : 0.7553   
## Detection Rate : 0.7044   
## Detection Prevalence : 0.8393   
## Balanced Accuracy : 0.6907   
##   
## 'Positive' Class : 0   
##

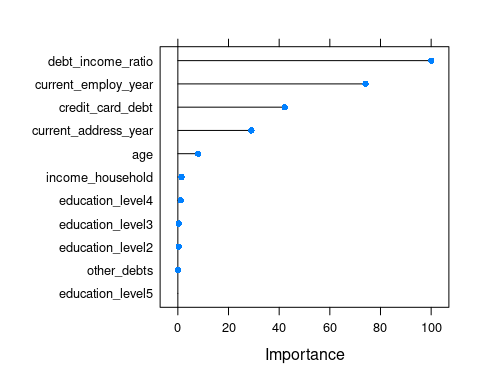
summary(xglm\_clf)

## Length Class Mode   
## handle 1 xgb.Booster.handle externalptr  
 ## raw 57241 -none- raw   
## niter 1 -none- numeric   
## call 6 -none- call   
## params 8 -none- list   
## callbacks 0 -none- list   
## feature\_names 11 -none- character   
## nfeatures 1 -none- numeric   
## xNames 11 -none- character   
## problemType 1 -none- character   
## tuneValue 7 data.frame list   
## obsLevels 2 -none- character   
## param 1 -none- list

varImp(xglm\_clf)

## xgbTree variable importance  
 ##   
## Overall  
 ## debt\_income\_ratio 100.0000  
 ## current\_employ\_year 74.0824  
 ## credit\_card\_debt 42.1879  
 ## current\_address\_year 29.0426  
 ## age 8.0120  
 ## income\_household 1.4524  
 ## education\_level4 1.1738  
 ## education\_level3 0.3552  
 ## education\_level2 0.3359  
 ## other\_debts 0.1395  
 ## education\_level5 0.0000

plot(varImp(xglm\_clf))



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.N** | **Model** | **Accuracy** | **Sensitivity** | **Specificity** |
| 1 | Bagging | 0.7871 | 0.8896 | 0.4709 |
| 2 | Random Forest | 0.8047 | 0.9183 | 0.4543 |
| 3 | Extreme Boosting | 0.8142 | 0.9255 | 0.8273 |

The table below shows the evaluation metrices on different models

From the table we can see that Extreme Boosting algorithm gives the highest accuracy so we choose this to be the best model in our case.

QN3:

From the variable importance we can see that the variables like debt\_income\_ratio, current\_employ\_year ,credit\_card\_debt ,current\_address\_year

And age is very important for the model. Based on these value the model determines if the person will get loan or not.