FML Assignment 2

Sai Kiran

2023-02-19

library(class)  
library(dplyr)

##   
## Attaching package: 'dplyr'

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

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

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(tinytex)  
ub=read.csv("C:/Users/panug/Downloads/UniversalBank.csv")

#eliminating unwanted columns like ID and Zip code

ub$ID<-NULL  
ub$ZIP.Code<-NULL  
View(ub)

#converting to a variable factor  
ub$Personal.Loan=as.factor(ub$Personal.Loan)

#using is.na command to check if there are any NA values

head(is.na(ub))

## Age Experience Income Family CCAvg Education Mortgage Personal.Loan  
## [1,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [2,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [3,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [4,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [5,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [6,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## Securities.Account CD.Account Online CreditCard  
## [1,] FALSE FALSE FALSE FALSE  
## [2,] FALSE FALSE FALSE FALSE  
## [3,] FALSE FALSE FALSE FALSE  
## [4,] FALSE FALSE FALSE FALSE  
## [5,] FALSE FALSE FALSE FALSE  
## [6,] FALSE FALSE FALSE FALSE

#transforming education into character

ub$Education=as.character(ub$Education)

#Creating dummy variables

edu.1 <- ifelse(ub$Education==1 ,1,0)  
edu.2 <- ifelse(ub$Education==2 ,1,0)  
edu.3 <- ifelse(ub$Education==3 ,1,0)  
ub.2<-data.frame(Age=ub$Age,Experience=ub$Experience,Income=ub$Income,Family=ub$Family,CCAvg=ub$CCAvg, education\_1=edu.1,edu.2=edu.2,edu.3=edu.3,Personal.Loan=ub$Personal.Loan,Mortgage=ub$Mortgage,Securities.Account=ub$Securities.Account,CD.Account=ub$CD.Account,Online=ub$Online,CreditCard=ub$CreditCard)

#setting up testdata

UB\_test1<-data.frame(Age=40,Experience=10,Income=84,Family=2,CCAvg=2,edu.1=0,edu.2=1,edu.3=0,Mortgage=0,Securities.Account=0,CD.Account=0,Online=1,CreditCard=1)

#dividing data sets for training and testing

set.seed(130)  
ub.dummy<- createDataPartition(ub.2$Personal.Loan,p=.6,list=FALSE,times=1)  
train1.ub <- ub.2[ub.dummy, ]  
valid1.ub<- ub.2[-ub.dummy, ]

#Normalization

ub.norm=preProcess(train1.ub[,-(6:9)],method=c("center","scale"))  
trainNorm.ub =predict(ub.norm,train1.ub)  
validNorm.ub =predict(ub.norm,valid1.ub)  
testNorm.ub =predict(ub.norm,UB\_test1)  
View(trainNorm.ub)

#printing knn algorithm

predicttrain.ub<-trainNorm.ub[,-9]  
trainsample.ub<-trainNorm.ub[,9]  
predictvalid.ub<-validNorm.ub[,-9]  
validsample.ub<-validNorm.ub[,9]  
  
predict.ub<-knn(predicttrain.ub, testNorm.ub, cl=trainsample.ub,k=1)  
predict.ub

## [1] 0  
## Levels: 0 1

predict.uvb <- knn(predicttrain.ub, predictvalid.ub, cl=trainsample.ub, k=1)

#The loan proposal was turned down by the client. The decision is made when the k value is 0.

#printing out the best value of k

set.seed(130)  
grid.ub<-expand.grid(k=seq(1:30))   
model.ub<-train(Personal.Loan~.,data=trainNorm.ub,method="knn",tuneGrid=grid.ub)  
model.ub

## k-Nearest Neighbors   
##   
## 3000 samples  
## 13 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 3000, 3000, 3000, 3000, 3000, 3000, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 1 0.9498389 0.6848818  
## 2 0.9456633 0.6536088  
## 3 0.9457715 0.6453799  
## 4 0.9456939 0.6379842  
## 5 0.9464967 0.6369189  
## 6 0.9468210 0.6342505  
## 7 0.9476230 0.6362095  
## 8 0.9475486 0.6304329  
## 9 0.9474853 0.6264414  
## 10 0.9454230 0.6086942  
## 11 0.9455233 0.6063682  
## 12 0.9445282 0.5965274  
## 13 0.9439058 0.5896361  
## 14 0.9425072 0.5751621  
## 15 0.9412785 0.5625136  
## 16 0.9410684 0.5580477  
## 17 0.9403809 0.5494274  
## 18 0.9392614 0.5384893  
## 19 0.9381366 0.5268213  
## 20 0.9379190 0.5236724  
## 21 0.9371251 0.5153713  
## 22 0.9373413 0.5176735  
## 23 0.9369361 0.5122613  
## 24 0.9363567 0.5059488  
## 25 0.9357750 0.5000855  
## 26 0.9350157 0.4931644  
## 27 0.9346204 0.4881624  
## 28 0.9340405 0.4818989  
## 29 0.9334942 0.4759017  
## 30 0.9328745 0.4683129  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 1.

value\_k<-model.ub$bestTune[[1]]

#confusion matrix - validation dataset

confusionMatrix(predict.uvb,validsample.ub)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1784 64  
## 1 24 128  
##   
## Accuracy : 0.956   
## 95% CI : (0.9461, 0.9646)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7205   
##   
## Mcnemar's Test P-Value : 3.219e-05   
##   
## Sensitivity : 0.9867   
## Specificity : 0.6667   
## Pos Pred Value : 0.9654   
## Neg Pred Value : 0.8421   
## Prevalence : 0.9040   
## Detection Rate : 0.8920   
## Detection Prevalence : 0.9240   
## Balanced Accuracy : 0.8267   
##   
## 'Positive' Class : 0   
##

#50:30:20 Repartition

data\_part\_new <- createDataPartition(ub.2$Personal.Loan,p=0.5, list = F)  
Train\_new <- ub.2[data\_part\_new,]  
Train\_db\_new <- ub.2[-data\_part\_new,]  
  
data\_part\_new\_1 <- createDataPartition(Train\_db\_new$Personal.Loan, p=0.6, list = F)  
validate\_new <- Train\_db\_new[data\_part\_new\_1,]  
test\_new <- Train\_db\_new[-data\_part\_new\_1,]

#Normalization

norm\_new <- preProcess(Train\_new[,-(6:9)], method=c("center","scale"))  
Train\_new\_p <- predict(norm\_new, Train\_new)  
Validate\_new\_p <- predict(norm\_new, validate\_new)  
Test\_new\_p <- predict(norm\_new, test\_new)

#predictors and labels

train\_pre <- Train\_new\_p[,-9]  
validate\_pre <- Validate\_new\_p[,-9]  
test\_pre <- Test\_new\_p[,-9]  
  
train\_l <- Train\_new\_p[,9]  
validate\_l <- Validate\_new\_p[,9]  
test\_l <- Test\_new\_p[,9]

#knn

knn.t <- knn(train\_pre,train\_pre,cl= train\_l, k=value\_k)  
  
knn.v <- knn(train\_pre,validate\_pre,cl=train\_l, k=value\_k)  
  
knn.tes <- knn(train\_pre,test\_pre,cl=train\_l, k=value\_k)  
  
confusionMatrix(knn.t,train\_l)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2260 0  
## 1 0 240  
##   
## Accuracy : 1   
## 95% CI : (0.9985, 1)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.000   
## Specificity : 1.000   
## Pos Pred Value : 1.000   
## Neg Pred Value : 1.000   
## Prevalence : 0.904   
## Detection Rate : 0.904   
## Detection Prevalence : 0.904   
## Balanced Accuracy : 1.000   
##   
## 'Positive' Class : 0   
##

confusionMatrix(knn.v,validate\_l)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1337 49  
## 1 19 95  
##   
## Accuracy : 0.9547   
## 95% CI : (0.9429, 0.9646)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 1.551e-13   
##   
## Kappa : 0.712   
##   
## Mcnemar's Test P-Value : 0.0004368   
##   
## Sensitivity : 0.9860   
## Specificity : 0.6597   
## Pos Pred Value : 0.9646   
## Neg Pred Value : 0.8333   
## Prevalence : 0.9040   
## Detection Rate : 0.8913   
## Detection Prevalence : 0.9240   
## Balanced Accuracy : 0.8229   
##   
## 'Positive' Class : 0   
##

confusionMatrix(knn.tes,test\_l)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 899 34  
## 1 5 62  
##   
## Accuracy : 0.961   
## 95% CI : (0.9471, 0.9721)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 5.695e-12   
##   
## Kappa : 0.7402   
##   
## Mcnemar's Test P-Value : 7.340e-06   
##   
## Sensitivity : 0.9945   
## Specificity : 0.6458   
## Pos Pred Value : 0.9636   
## Neg Pred Value : 0.9254   
## Prevalence : 0.9040   
## Detection Rate : 0.8990   
## Detection Prevalence : 0.9330   
## Balanced Accuracy : 0.8202   
##   
## 'Positive' Class : 0   
##