Assignment\_2

Srilaya Valmeekam

2023-02-20

#Installing necessary packages  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(ggplot2)  
library(ISLR)  
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(class)

#Importing the dataset  
library(readr)  
UniBank <- read.csv("~/Downloads/UniversalBank.csv")

#Executing a K-NN classification with all attributes except for ID and ZIP code.  
UniBank$ID <- NULL  
UniBank$ZIP.Code <- NULL  
summary(UniBank)

## Age Experience Income Family   
## Min. :23.00 Min. :-3.0 Min. : 8.00 Min. :1.000   
## 1st Qu.:35.00 1st Qu.:10.0 1st Qu.: 39.00 1st Qu.:1.000   
## Median :45.00 Median :20.0 Median : 64.00 Median :2.000   
## Mean :45.34 Mean :20.1 Mean : 73.77 Mean :2.396   
## 3rd Qu.:55.00 3rd Qu.:30.0 3rd Qu.: 98.00 3rd Qu.:3.000   
## Max. :67.00 Max. :43.0 Max. :224.00 Max. :4.000   
## CCAvg Education Mortgage Personal.Loan   
## Min. : 0.000 Min. :1.000 Min. : 0.0 Min. :0.000   
## 1st Qu.: 0.700 1st Qu.:1.000 1st Qu.: 0.0 1st Qu.:0.000   
## Median : 1.500 Median :2.000 Median : 0.0 Median :0.000   
## Mean : 1.938 Mean :1.881 Mean : 56.5 Mean :0.096   
## 3rd Qu.: 2.500 3rd Qu.:3.000 3rd Qu.:101.0 3rd Qu.:0.000   
## Max. :10.000 Max. :3.000 Max. :635.0 Max. :1.000   
## Securities.Account CD.Account Online CreditCard   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.000   
## Median :0.0000 Median :0.0000 Median :1.0000 Median :0.000   
## Mean :0.1044 Mean :0.0604 Mean :0.5968 Mean :0.294   
## 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.:1.0000 3rd Qu.:1.000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.000

UniBank$Personal.Loan = as.factor(UniBank$Personal.Loan)

#Creating dummy variables  
education\_1 <- ifelse(UniBank$Education==1 ,1,0)  
  
education\_2 <- ifelse(UniBank$Education==2 ,1,0)  
  
education\_3 <- ifelse(UniBank$Education==3 ,1,0)

#Removing and combining education variable  
unibank<-data.frame(Age=UniBank$Age,Experience=UniBank$Experience,Income=UniBank$Income,Family=UniBank$Family,CCAvg=UniBank$CCAvg, education\_1=education\_1,education\_2=education\_2,education\_3=education\_3,Personal.Loan=UniBank$Personal.Loan,Mortgage=UniBank$Mortgage,Securities.Account=UniBank$Securities.Account,CD.Account=UniBank$CD.Account,Online=UniBank$Online,CreditCard=UniBank$CreditCard)  
head(unibank)

## Age Experience Income Family CCAvg education\_1 education\_2 education\_3  
## 1 25 1 49 4 1.6 1 0 0  
## 2 45 19 34 3 1.5 1 0 0  
## 3 39 15 11 1 1.0 1 0 0  
## 4 35 9 100 1 2.7 0 1 0  
## 5 35 8 45 4 1.0 0 1 0  
## 6 37 13 29 4 0.4 0 1 0  
## Personal.Loan Mortgage Securities.Account CD.Account Online CreditCard  
## 1 0 0 1 0 0 0  
## 2 0 0 1 0 0 0  
## 3 0 0 0 0 0 0  
## 4 0 0 0 0 0 0  
## 5 0 0 0 0 0 1  
## 6 0 155 0 0 1 0

#separating into validation and training  
Model.normalise <- preProcess(UniBank[, -8],method = c("center", "scale"))  
summary(UniBank)

## Age Experience Income Family   
## Min. :23.00 Min. :-3.0 Min. : 8.00 Min. :1.000   
## 1st Qu.:35.00 1st Qu.:10.0 1st Qu.: 39.00 1st Qu.:1.000   
## Median :45.00 Median :20.0 Median : 64.00 Median :2.000   
## Mean :45.34 Mean :20.1 Mean : 73.77 Mean :2.396   
## 3rd Qu.:55.00 3rd Qu.:30.0 3rd Qu.: 98.00 3rd Qu.:3.000   
## Max. :67.00 Max. :43.0 Max. :224.00 Max. :4.000   
## CCAvg Education Mortgage Personal.Loan  
## Min. : 0.000 Min. :1.000 Min. : 0.0 0:4520   
## 1st Qu.: 0.700 1st Qu.:1.000 1st Qu.: 0.0 1: 480   
## Median : 1.500 Median :2.000 Median : 0.0   
## Mean : 1.938 Mean :1.881 Mean : 56.5   
## 3rd Qu.: 2.500 3rd Qu.:3.000 3rd Qu.:101.0   
## Max. :10.000 Max. :3.000 Max. :635.0   
## Securities.Account CD.Account Online CreditCard   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.000   
## Median :0.0000 Median :0.0000 Median :1.0000 Median :0.000   
## Mean :0.1044 Mean :0.0604 Mean :0.5968 Mean :0.294   
## 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.:1.0000 3rd Qu.:1.000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.000

UniBank.normalise <- predict(Model.normalise,UniBank)  
summary(UniBank.normalise)

## Age Experience Income Family   
## Min. :-1.94871 Min. :-2.014710 Min. :-1.4288 Min. :-1.2167   
## 1st Qu.:-0.90188 1st Qu.:-0.881116 1st Qu.:-0.7554 1st Qu.:-1.2167   
## Median :-0.02952 Median :-0.009121 Median :-0.2123 Median :-0.3454   
## Mean : 0.00000 Mean : 0.000000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.84284 3rd Qu.: 0.862874 3rd Qu.: 0.5263 3rd Qu.: 0.5259   
## Max. : 1.88967 Max. : 1.996468 Max. : 3.2634 Max. : 1.3973   
## CCAvg Education Mortgage Personal.Loan  
## Min. :-1.1089 Min. :-1.0490 Min. :-0.5555 0:4520   
## 1st Qu.:-0.7083 1st Qu.:-1.0490 1st Qu.:-0.5555 1: 480   
## Median :-0.2506 Median : 0.1417 Median :-0.5555   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.3216 3rd Qu.: 1.3324 3rd Qu.: 0.4375   
## Max. : 4.6131 Max. : 1.3324 Max. : 5.6875   
## Securities.Account CD.Account Online CreditCard   
## Min. :-0.3414 Min. :-0.2535 Min. :-1.2165 Min. :-0.6452   
## 1st Qu.:-0.3414 1st Qu.:-0.2535 1st Qu.:-1.2165 1st Qu.:-0.6452   
## Median :-0.3414 Median :-0.2535 Median : 0.8219 Median :-0.6452   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.:-0.3414 3rd Qu.:-0.2535 3rd Qu.: 0.8219 3rd Qu.: 1.5495   
## Max. : 2.9286 Max. : 3.9438 Max. : 0.8219 Max. : 1.5495

Index\_Train <- createDataPartition(UniBank$Personal.Loan, p = 0.6, list = FALSE)  
Train = UniBank.normalise[Index\_Train,]  
validation = UniBank.normalise[-Index\_Train,]

#Q1  
#Data prediction  
library(FNN)

##   
## Attaching package: 'FNN'

## The following objects are masked from 'package:class':  
##   
## knn, knn.cv

to\_Predict = data.frame(Age = 40, Experience = 10, Income = 84, Family = 2,  
 CCAvg = 2, Education = 1, Mortgage = 0, Securities.Account =  
 0, CD.Account = 0, Online = 1, CreditCard = 1)  
print(to\_Predict)

## Age Experience Income Family CCAvg Education Mortgage Securities.Account  
## 1 40 10 84 2 2 1 0 0  
## CD.Account Online CreditCard  
## 1 0 1 1

Predict.Normalise <- predict(Model.normalise,to\_Predict)  
Predictions <- knn(train= as.data.frame(Train[,1:7,9:12]),  
 test = as.data.frame(Predict.Normalise[,1:7,9:12]),  
 cl= Train$Personal.Loan,  
 k=1)

## Warning in drop && !has.j: 'length(x) = 4 > 1' in coercion to 'logical(1)'

## Warning in drop && length(y) == 1L: 'length(x) = 4 > 1' in coercion to  
## 'logical(1)'

## Warning in drop && !mdrop: 'length(x) = 4 > 1' in coercion to 'logical(1)'

## Warning in drop && !has.j: 'length(x) = 4 > 1' in coercion to 'logical(1)'

## Warning in drop && length(y) == 1L: 'length(x) = 4 > 1' in coercion to  
## 'logical(1)'

## Warning in drop && !mdrop: 'length(x) = 4 > 1' in coercion to 'logical(1)'

#Q2  
#Value of K that gives largest accuracy  
set.seed(123)  
UniBank <- trainControl(method= "repeatedcv", number = 3, repeats = 2)  
searchGrid = expand.grid(k=1:10)  
knn.model = train(Personal.Loan~., data = Train, method = 'knn', tuneGrid = searchGrid,trControl = UniBank)  
knn.model

## k-Nearest Neighbors   
##   
## 3000 samples  
## 11 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold, repeated 2 times)   
## Summary of sample sizes: 2000, 2000, 2000, 2000, 2000, 2000, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 1 0.9480000 0.6701386  
## 2 0.9471667 0.6658123  
## 3 0.9543333 0.6900162  
## 4 0.9495000 0.6488627  
## 5 0.9498333 0.6398443  
## 6 0.9496667 0.6422238  
## 7 0.9483333 0.6198590  
## 8 0.9466667 0.6068889  
## 9 0.9456667 0.5983958  
## 10 0.9456667 0.5952577  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 3.

#The value of k is 3.This is the value that balances between overfitting and ignoring the predictor information

#Q3  
#Finding Confusion matrix using the K value  
UniBank\_prediction <- predict(knn.model,validation)  
confusionMatrix(UniBank\_prediction,validation$Personal.Loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1791 58  
## 1 17 134  
##   
## Accuracy : 0.9625   
## 95% CI : (0.9532, 0.9704)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7612   
##   
## Mcnemar's Test P-Value : 3.86e-06   
##   
## Sensitivity : 0.9906   
## Specificity : 0.6979   
## Pos Pred Value : 0.9686   
## Neg Pred Value : 0.8874   
## Prevalence : 0.9040   
## Detection Rate : 0.8955   
## Detection Prevalence : 0.9245   
## Balanced Accuracy : 0.8443   
##   
## 'Positive' Class : 0   
##

#This matrix has a 94.5% accuracy.  
#This the confusion matrix for the validation data that results from using the best k.

#Q4  
#Classifying and predicting the customer  
ForPredictNorm = data.frame(Age = 40, Experience = 10, Income = 84, Family = 2,  
 CCAvg = 2, Education = 1, Mortgage = 0,  
 Securities.Account =0, CD.Account = 0, Online = 1,  
 CreditCard = 1)  
ForPredictNorm = predict(Model.normalise, ForPredictNorm)  
predict(knn.model, ForPredictNorm)

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

#It results in level 0,1

#Q5  
#Partitioning the data into 3 parts  
#Creating Training, Test, and validation sets from the data collection.  
Train\_size = 0.5 #training(50%)  
Train\_Index = createDataPartition(UniBank.normalise$Personal.Loan, p = 0.5, list = FALSE)  
Train = UniBank.normalise[Train\_Index,]  
  
valid\_size = 0.3 #validation(30%)  
Validation\_Index = createDataPartition(UniBank.normalise$Personal.Loan, p = 0.3, list = FALSE)  
validation = UniBank.normalise[Validation\_Index,]  
  
Test\_size = 0.2 #Test Data(20%)  
Test\_Index = createDataPartition(UniBank.normalise$Personal.Loan, p = 0.2, list = FALSE)  
Test = UniBank.normalise[Test\_Index,]  
  
  
Trainingknn <- knn(train = Train[,-8], test = Train[,-8], cl = Train[,8], k =3)  
Validknn <- knn(train = Train[,-8], test = validation[,-8], cl = Train[,8], k =3)  
Testingknn <- knn(train = Train[,-8], test = Test[,-8], cl = Train[,8], k =3)  
  
confusionMatrix(Trainingknn, Train[,8])

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2255 63  
## 1 5 177  
##   
## Accuracy : 0.9728   
## 95% CI : (0.9656, 0.9788)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8243   
##   
## Mcnemar's Test P-Value : 4.77e-12   
##   
## Sensitivity : 0.9978   
## Specificity : 0.7375   
## Pos Pred Value : 0.9728   
## Neg Pred Value : 0.9725   
## Prevalence : 0.9040   
## Detection Rate : 0.9020   
## Detection Prevalence : 0.9272   
## Balanced Accuracy : 0.8676   
##   
## 'Positive' Class : 0   
##

confusionMatrix(Validknn, validation[,8])

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1351 46  
## 1 5 98  
##   
## Accuracy : 0.966   
## 95% CI : (0.9555, 0.9746)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7756   
##   
## Mcnemar's Test P-Value : 2.13e-08   
##   
## Sensitivity : 0.9963   
## Specificity : 0.6806   
## Pos Pred Value : 0.9671   
## Neg Pred Value : 0.9515   
## Prevalence : 0.9040   
## Detection Rate : 0.9007   
## Detection Prevalence : 0.9313   
## Balanced Accuracy : 0.8384   
##   
## 'Positive' Class : 0   
##

confusionMatrix(Testingknn, Test[,8])

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 896 24  
## 1 8 72  
##   
## Accuracy : 0.968   
## 95% CI : (0.9551, 0.978)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 3.349e-15   
##   
## Kappa : 0.8008   
##   
## Mcnemar's Test P-Value : 0.00801   
##   
## Sensitivity : 0.9912   
## Specificity : 0.7500   
## Pos Pred Value : 0.9739   
## Neg Pred Value : 0.9000   
## Prevalence : 0.9040   
## Detection Rate : 0.8960   
## Detection Prevalence : 0.9200   
## Balanced Accuracy : 0.8706   
##   
## 'Positive' Class : 0   
##