assignment 2

Rishitha Reddy Muddasani

2023-02-19

```
#loading the packages
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
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)
library(FNN)
```

```
##
## Attaching package: 'FNN'
```

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

```
# Importing the dataset.
RR <- read.csv("~/Downloads/UniversalBank.csv")
```

```
#Performing a K-NN classification with all attributes except ID and ZIP code.
RR$ID <- NULL
RR$ZIP.Code <- NULL
summary(RR)
```

```
##       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
```

```
RR$Personal.Loan =  as.factor(RR$Personal.Loan)
```

```
#Creating dummy  variables
education_1 <- ifelse(RR$Education==1 ,1,0)
education_2 <- ifelse(RR$Education==2 ,1,0)
education_3 <- ifelse(RR$Education==3 ,1,0)
unibank<-data.frame(Age=RR$Age,Experience=RR$Experience,Income=RR$Income,Family=RR$Family,CCAvg=RR$CCAv
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
```

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

```
##       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
```

```
RR.normalise <- predict(Model.normalise,RR)
summary(RR.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(RR$Personal.Loan, p = 0.6, list = FALSE)
Train = RR.normalise[Index_Train,]
validation = RR.normalise[-Index_Train,]
```

```
#QUESTION-1  - Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Education_1 = 0, Educatio
#Prediction of data
library(FNN)
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)'
```

```
#QUESTION 2 - What is a choice of k that balances between overfitting and ignoring the predictor inform
set.seed(123)
RR <- 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 = RR)
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  0.9520000  0.6887687
##    2  0.9483333  0.6662532
##    3  0.9496667  0.6509650
##    4  0.9480000  0.6354258
##    5  0.9495000  0.6385828
##    6  0.9485000  0.6305158
##    7  0.9455000  0.5984761
##    8  0.9451667  0.5962477
##    9  0.9435000  0.5776461
##   10  0.9431667  0.5710782
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

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

```
#QUESTION 3- Show the confusion matrix for the validation data that results from using the best k.
RR_prediction <- predict(knn.model,validation)
confusionMatrix(RR_prediction,validation$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1782   58
##          1   26  134
```

```
##
##                Accuracy : 0.958
##                  95% CI : (0.9483, 0.9664)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7385
##
##  Mcnemar's Test P-Value : 0.0007186
##
##             Sensitivity : 0.9856
##             Specificity : 0.6979
##          Pos Pred Value : 0.9685
##          Neg Pred Value : 0.8375
##              Prevalence : 0.9040
##          Detection Rate : 0.8910
##    Detection Prevalence : 0.9200
##       Balanced Accuracy : 0.8418
##
##        'Positive' Class : 0
##
```

```
#This matrix has a 95.9% accuracy.
#This the confusion matrix for the validation data that results from using the best k.
```

```
#QUESTION 4 - Consider the following customer: Age = 40, Experience = 10, Income = 84, Family = 2, CCAv
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
```

```
#QUESTION 5 - Repartition the data, this time into training, validation, and test sets (50% : 30% : 20%
#Creating Training, Test, and validation sets from the data collection.
Train_size = 0.5 #training(50%)
Train_Index = createDataPartition(RR.normalise$Personal.Loan, p = 0.5, list = FALSE)
Train = RR.normalise[Train_Index,]
valid_size = 0.3 #validation(30%)
Validation_Index = createDataPartition(RR.normalise$Personal.Loan, p = 0.3, list = FALSE)
validation = RR.normalise[Validation_Index,]
Test_size = 0.2 #Test Data(20%)
Test_Index = createDataPartition(RR.normalise$Personal.Loan, p = 0.2, list = FALSE)
Test = RR.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 2256   55
##          1    4  185
##
##                Accuracy : 0.9764
##                  95% CI : (0.9697, 0.982)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8498
##
##  Mcnemar's Test P-Value : 7.543e-11
##
##             Sensitivity : 0.9982
##             Specificity : 0.7708
##          Pos Pred Value : 0.9762
##          Neg Pred Value : 0.9788
##              Prevalence : 0.9040
##          Detection Rate : 0.9024
##    Detection Prevalence : 0.9244
##       Balanced Accuracy : 0.8845
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(Validknn, validation[,8])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1352   51
##          1    4   93
##
##                Accuracy : 0.9633
##                  95% CI : (0.9525, 0.9723)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7527
##
##  Mcnemar's Test P-Value : 5.552e-10
##
##             Sensitivity : 0.9971
##             Specificity : 0.6458
##          Pos Pred Value : 0.9636
##          Neg Pred Value : 0.9588
##              Prevalence : 0.9040
##          Detection Rate : 0.9013
##    Detection Prevalence : 0.9353
##       Balanced Accuracy : 0.8214
##
##        'Positive' Class : 0
##
```

```r
confusionMatrix(Testingknn, Test[,8])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 902  27
##          1   2  69
##
##                Accuracy : 0.971
##                  95% CI : (0.9586, 0.9805)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8109
##
##  Mcnemar's Test P-Value : 8.324e-06
##
##             Sensitivity : 0.9978
##             Specificity : 0.7188
##          Pos Pred Value : 0.9709
##          Neg Pred Value : 0.9718
##              Prevalence : 0.9040
##          Detection Rate : 0.9020
##    Detection Prevalence : 0.9290
##       Balanced Accuracy : 0.8583
##
##        'Positive' Class : 0
##
```

```r
# The accuracy for this knn model is 0.973 or 97.3%.
# The Sensitivity for this knn model is 0.9956 or 99.56%.
# The Specificity for this knn model is 0.7604 or 76.04%.
```