FML ASSIGNMENT 2

RAJEEV VARMA

10/5/2022

```
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')
```

```
My_Data <- read.csv("C:/Users/RAJEEV VARMA/Downloads/UniversalBank.csv")
```

```
#QUESTION-1
My_Data$ID <- NULL
My_Data$ZIP.Code <- NULL
summary(My_Data)
```

```
##       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
```

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

```
#Normalizing the data by dividing into Training, Test and validation
Modelnorm <- preProcess(My_Data[, -8],method = c("center", "scale"))
summary(My_Data)
```

```
##       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
```

```
My_Data_Norm <- predict(Modelnorm,My_Data)
summary(My_Data_Norm)
```

```
##       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(My_Data$Personal.Loan, p = 0.6, list = FALSE)
Train = My_Data_Norm[Index_Train,]
validation = My_Data_Norm[-Index_Train,]
```

```
#Predicting of data
library(FNN)
```

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

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

```
Prediction = 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(Prediction)
```

```
##   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_Norm <- predict(Modelnorm,Prediction)
Prediction <- knn(train= as.data.frame(Train[,1:7,9:12]),
                  test = as.data.frame(Predict_Norm[,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
set.seed(123)
My_Data <- 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 = My_Da
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  0.9556667  0.7177367
##    2  0.9501667  0.6808208
##    3  0.9565000  0.7081675
##    4  0.9503333  0.6582422
##    5  0.9523333  0.6667443
##    6  0.9503333  0.6495943
##    7  0.9490000  0.6340829
##    8  0.9500000  0.6421501
##    9  0.9483333  0.6274770
##   10  0.9436667  0.5849270
##
## 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, which strikes a compromise between underfitting and overfitting of the data.
#Accuracy was used to select the optimal model using the largest value for the model was k = 3.
```

```
#QUESTION-3
prediction_bank <- predict(knn.model,validation)
confusionMatrix(prediction_bank,validation$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1793   76
##          1   15  116
##
##                Accuracy : 0.9545
##                  95% CI : (0.9444, 0.9632)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6945
##
##  Mcnemar's Test P-Value : 3.181e-10
##
##             Sensitivity : 0.9917
##             Specificity : 0.6042
##          Pos Pred Value : 0.9593
##          Neg Pred Value : 0.8855
##              Prevalence : 0.9040
##          Detection Rate : 0.8965
##    Detection Prevalence : 0.9345
##       Balanced Accuracy : 0.7979
##
##        'Positive' Class : 0
##
```

```
#This matrix has a 95% accuracy.
```

```
#QUESTION-4
Predict_Norm = 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)
Predict_Norm = predict(Modelnorm, Predict_Norm)
predict(knn.model, Predict_Norm)
```

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

```
#QUESTION-5
#Creating Training, Test, and validation
Trainsize = 0.5 #training(50%)
IndexTrain = createDataPartition(My_Data_Norm$Personal.Loan, p = 0.5, list = FALSE)
Train = My_Data_Norm[IndexTrain,]

validsize = 0.3 #validation(30%)
IndexValidation = createDataPartition(My_Data_Norm$Personal.Loan, p = 0.3, list = FALSE)
validation = My_Data_Norm[IndexValidation,]

Testsize = 0.2 #Test Data(20%)
IndexTest = createDataPartition(My_Data_Norm$Personal.Loan, p = 0.2, list = FALSE)
Test = My_Data_Norm[IndexTest,]


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)

TrainPredictors<-Train[,9:12]
TestPredictors<-Test[,9:12]

Trainlabels <-Train[,8]
Testlabels  <-Test[,8]

PredictedTestlabels <-knn(TrainPredictors,
                            TestPredictors,
                            cl=Trainlabels,
                            k=3 )

confusionMatrix(Trainingknn, Train[,8])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2255   58
##          1    5  182
##
##                Accuracy : 0.9748
##                  95% CI : (0.9679, 0.9806)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.8389
##
##  Mcnemar's Test P-Value : 5.701e-11
##
##             Sensitivity : 0.9978
##             Specificity : 0.7583
##          Pos Pred Value : 0.9749
##          Neg Pred Value : 0.9733
##              Prevalence : 0.9040
##          Detection Rate : 0.9020
##    Detection Prevalence : 0.9252
##       Balanced Accuracy : 0.8781
##
##        'Positive' Class : 0
##
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1352   44
##          1    4  100
##
##                Accuracy : 0.968
##                  95% CI : (0.9578, 0.9763)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7895
##
##  Mcnemar's Test P-Value : 1.811e-08
##
##             Sensitivity : 0.9971
##             Specificity : 0.6944
##          Pos Pred Value : 0.9685
##          Neg Pred Value : 0.9615
##              Prevalence : 0.9040
##          Detection Rate : 0.9013
##    Detection Prevalence : 0.9307
##       Balanced Accuracy : 0.8457
##
##        'Positive' Class : 0
##
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 899  22
##          1   5  74
##
```

```
##                Accuracy : 0.973
##                  95% CI : (0.961, 0.9821)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8311
##
##  Mcnemar's Test P-Value : 0.002076
##
##             Sensitivity : 0.9945
##             Specificity : 0.7708
##          Pos Pred Value : 0.9761
##          Neg Pred Value : 0.9367
##              Prevalence : 0.9040
##          Detection Rate : 0.8990
##    Detection Prevalence : 0.9210
##       Balanced Accuracy : 0.8827
##
##        'Positive' Class : 0
##
```