Assignment 2 (BA 64060-002)

Jared Bartee

2023-09-28

Load necessary packages

```
library(caret)
```

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

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

```
library(class)
```

Load the data the data summary

```
data <- read.csv("UniversalBank.csv")
summary(data)
```

```
##        ID            Age          Experience       Income
ZIP.Code
##  Min.   :   1   Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   :
9307
##  1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st
Qu.:91911
##  Median :2500   Median :45.00   Median :20.0   Median : 64.00   Median
:93437
##  Mean   :2500   Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean
:93153
##  3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd
Qu.:94608
##  Max.   :5000   Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.
:96651
##      Family          CCAvg          Education        Mortgage
##  Min.   :1.000   Min.   : 0.000   Min.   :1.000   Min.   :  0.0
##  1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0
##  Median :2.000   Median : 1.500   Median :2.000   Median :  0.0
##  Mean   :2.396   Mean   : 1.938   Mean   :1.881   Mean   : 56.5
##  3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0
##  Max.   :4.000   Max.   :10.000   Max.   :3.000   Max.   :635.0
##  Personal.Loan   Securities.Account   CD.Account         Online
##  Min.   :0.000   Min.   :0.0000     Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.000   Median :0.0000     Median :0.0000   Median :1.0000
##  Mean   :0.096   Mean   :0.1044     Mean   :0.0604   Mean   :0.5968
```

```
##  3rd Qu.:0.000   3rd Qu.:0.0000     3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :1.000   Max.   :1.0000     Max.   :1.0000   Max.   :1.0000
##    CreditCard
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.294
##  3rd Qu.:1.000
##  Max.   :1.000
```

Load the data structure

```
str(data)
```

```
## 'data.frame':    5000 obs. of  14 variables:
##  $ ID                : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP.Code          : int  91107 90089 94720 94112 91330 92121 91711
93943 90089 93023 ...
##  $ Family            : int  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : int  1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal.Loan     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ Securities.Account: int  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD.Account        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : int  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : int  0 0 0 0 1 0 0 1 0 0 ...
```

#1 Consider the following customer: Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Education_1 = 0, Education_2 = 1, Education_3 = 0, Mortgage = 0, Securities Account = 0, CD Account = 0, Online = 1, and Credit Card = 1. Perform a k-NN classification with all predictors except ID and ZIP code using k = 1. Remember to transform categorical predictors with more than two categories into dummy variables first. Specify the success class as 1 (loan acceptance), and use the default cutoff value of 0.5. How would this customer be classified?

#1(a) Convert Education to a factor

```
data$Education = as.factor(data$Education)
```

#Remove ID and ZIP code from the dataset. Also, transforming categorical predictor Education w/more than two categories into dummy variables

```
data_dummy = model.matrix(~ . - ZIP.Code - ID - 1, data = data)
head(data_dummy)
```

```
##   Age Experience Income Family CCAvg Education1 Education2 Education3
Mortgage
## 1  25          1     49      4   1.6          1          0          0
0
## 2  45         19     34      3   1.5          1          0          0
0
## 3  39         15     11      1   1.0          1          0          0
0
## 4  35          9    100      1   2.7          0          1          0
0
## 5  35          8     45      4   1.0          0          1          0
0
## 6  37         13     29      4   0.4          0          1          0
155
##   Personal.Loan Securities.Account CD.Account Online CreditCard
## 1             0                  1          0      0          0
## 2             0                  1          0      0          0
## 3             0                  0          0      0          0
## 4             0                  0          0      0          0
## 5             0                  0          0      0          1
## 6             0                  0          0      1          0
```

#Convert Personal.Loan to a factor present in the dataset

```
data_dummy <- as.data.frame(data_dummy)
data_dummy$Personal.Loan = as.factor(data_dummy$Personal.Loan)
```

#Set set.seed

```
set.seed(3.14)
```

#Divide the data into validation and training sets

```
train.index <- sample(row.names(data_dummy), 0.6*dim(data_dummy)[1])
test.index <- setdiff(row.names(data_dummy), train.index)
train_data <- data_dummy[train.index, ]
valid_data <- data_dummy[test.index, ]
```

#Classify the given customer

```
Given_CusData = data.frame(Age=40 , Experience=10, Income = 84, Family = 2,
CCAvg = 2, Education1 = 0, Education2 = 1, Education3 = 0, Mortgage = 0,
Securities.Account = 0, CD.Account = 0, Online = 1,  CreditCard = 1,
stringsAsFactors = FALSE)
Given_CusData
```

```
##   Age Experience Income Family CCAvg Education1 Education2 Education3
Mortgage
## 1  40         10     84      2     2          0          1          0
0
##   Securities.Account CD.Account Online CreditCard
## 1                  0          0      1          1
```

```
norm.values <- preProcess(train_data[, -c(10)], method=c("center", "scale"))
train_data[, -c(10)] <- predict(norm.values, train_data[, -c(10)])
valid_data[, -c(10)] <- predict(norm.values, valid_data[, -c(10)])
new.df <- predict(norm.values, Given_CusData)

knn.1 <- knn(train = train_data[,-c(10)],test = new.df, cl = train_data[,10],
k=5, prob=TRUE)
knn.attributes <- attributes(knn.1)

knn.attributes[1]

## $levels
## [1] "0" "1"

knn.attributes[3]

## $prob
## [1] 1
```

#2What is a choice of k that balances between overfitting and ignoring the predictor information? #Answer: The best choice of k which also balances the model from overfitting is k = 3.

```
my_accurateChoice <- data.frame(k = seq(1, 14, 1), accuracy = rep(0, 14))

for(i in 1:14) {
  test1 <- knn(train = train_data[,-10],test = valid_data[,-10], cl =
train_data[,10], k=i, prob=TRUE)
  my_accurateChoice[i, 2] <- confusionMatrix(test1,
valid_data[,10])$overall[1]
}
my_accurateChoice

##     k accuracy
## 1   1   0.9585
## 2   2   0.9565
## 3   3   0.9635
## 4   4   0.9600
## 5   5   0.9635
## 6   6   0.9615
## 7   7   0.9605
## 8   8   0.9590
## 9   9   0.9560
## 10 10   0.9565
## 11 11   0.9555
## 12 12   0.9545
## 13 13   0.9540
## 14 14   0.9530
```

#3 Show the confusion matrix for the validation data that results from using the best k.

```
test2 <- knn(train = train_data[,-10],test = valid_data[,-10], cl =
train_data[,10], k=3, prob=TRUE)
confusionMatrix(test2, valid_data[,10])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1794   62
##          1   11  133
##
##                Accuracy : 0.9635
##                  95% CI : (0.9543, 0.9713)
##     No Information Rate : 0.9025
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7652
##
##  Mcnemar's Test P-Value : 4.855e-09
##
##             Sensitivity : 0.9939
##             Specificity : 0.6821
##          Pos Pred Value : 0.9666
##          Neg Pred Value : 0.9236
##              Prevalence : 0.9025
##          Detection Rate : 0.8970
##    Detection Prevalence : 0.9280
##       Balanced Accuracy : 0.8380
##
##        'Positive' Class : 0
##
```

#4 Consider the following customer: Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Education_1 = 0, Education_2 = 1, Education_3 = 0, Mortgage = 0, Securities Account = 0, CD Account = 0, Online = 1 and Credit Card = 1. Classify the customer using the best k.

```
Given_CusData2= data.frame(Age = 40, Experience = 10, Income = 84, Family =
2, CCAvg = 2, Education_1 = 0, Education_2 = 1, Education_3 = 0, Mortgage =
0, Securities.Account = 0, CD.Account = 0, Online = 1, CreditCard = 1)
my_knn <- knn(train = train_data[,-10],test = Given_CusData2, cl =
train_data[,10], k=3, prob=TRUE)
my_knn
```

```
## [1] 1
## attr(,"prob")
## [1] 1
## Levels: 0 1
```

#5 Repartition the data, this time into training, validation, and test sets (50% : 30% : 20%). Apply the k-NN method with the k chosen above. Compare the confusion matrix of the test set with that of the training and validation sets. Comment on the differences and their reason.

```
set.seed(3.14)
train.index <- sample(rownames(data_dummy), 0.5*dim(data_dummy)[1])
valid.index <- sample(setdiff(rownames(data_dummy),train.index),
0.3*dim(data_dummy)[1])
test.index = setdiff(rownames(data_dummy), union(train.index, valid.index))

train_data<- data_dummy[train.index, ]
valid_data <- data_dummy[valid.index, ]
test_data <- data_dummy[test.index, ]

norm.values <- preProcess(train_data[, -c(10)], method=c("center", "scale"))
train_data[, -c(10)] <- predict(norm.values, train_data[, -c(10)])
valid_data[, -c(10)] <- predict(norm.values, valid_data[, -c(10)])
test_data[,-c(10)] <- predict(norm.values, test_data[,-c(10)])

test_data1 <- knn(train = train_data[,-c(10)],test = test_data[,-c(10)], cl =
train_data[,10], k=3, prob=TRUE)
valid_data1 <- knn(train = train_data[,-c(10)],test = valid_data[,-c(10)], cl
= train_data[,10], k=3, prob=TRUE)
train_data1 <- knn(train = train_data[,-c(10)],test = train_data[,-c(10)], cl
= train_data[,10], k=3, prob=TRUE)

confusionMatrix(test_data1, test_data[,10])

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 892  34
##          1   7  67
##
##                Accuracy : 0.959
##                  95% CI : (0.9448, 0.9704)
##     No Information Rate : 0.899
##     P-Value [Acc > NIR] : 1.416e-12
##
##                   Kappa : 0.7438
##
##  Mcnemar's Test P-Value : 4.896e-05
##
##             Sensitivity : 0.9922
##             Specificity : 0.6634
##          Pos Pred Value : 0.9633
##          Neg Pred Value : 0.9054
##              Prevalence : 0.8990
```

```
##          Detection Rate : 0.8920
##    Detection Prevalence : 0.9260
##       Balanced Accuracy : 0.8278
##
##        'Positive' Class : 0
##
```