knn classification

Shashidhar Reddy Boreddy

2022-10-03

```r
#importing the requiored packages in r
library('caret')
```

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

```r
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
```

```r
library('class')

#Importing the dataset from local folders

sb.data <- read.csv("~/Documents/assignments/FUNDAMENTALS ML/UB.csv", header = TRUE,
                         sep =",", stringsAsFactors = FALSE)
#Question_1
#conducting a k-NN classification
#predictors removed, i.e., removing ID and ZIP Code from each and every column from the data set
sb.data$ID <- NULL
sb.data$ZIP.Code <- NULL
summary(sb.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
```

```
#converting categorical variable "personal loan" into a factor that responses as "yes" or "no."

sb.data$Personal.Loan =  as.factor(sb.data$Personal.Loan)


#normalize the data by dividing
#training and validation, use preProcess() from the caret package.
M_norm <- preProcess(sb.data[, -8],method = c("center", "scale"))
sb.data_norm <- predict(M_norm,sb.data)
summary(sb.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
```

```
#partition of the data into test and training sets as per the requirements
sb_train_index <- createDataPartition(sb.data$Personal.Loan, p = 0.6, list = FALSE)
my_train.df = sb.data_norm[sb_train_index,]
validate.sb.df = sb.data_norm[-sb_train_index,]

print(head(my_train.df))
```

```
##           Age  Experience     Income     Family      CCAvg  Education
## 2 -0.02952064 -0.09632058 -0.8640230  0.5259383 -0.2505855 -1.0489730
## 3 -0.55293627 -0.44511864 -1.3636566 -1.2167334 -0.5366825 -1.0489730
## 4 -0.90188002 -0.96831574  0.5697084 -1.2167334  0.4360473  0.1416887
## 5 -0.90188002 -1.05551525 -0.6250678  1.3972742 -0.5366825  0.1416887
## 6 -0.72740814 -0.61951767 -0.9726390  1.3972742 -0.8799989  0.1416887
```

```
## 7  0.66836686  0.60127554 -0.0385413 -0.3453975 -0.2505855  0.1416887
##     Mortgage Personal.Loan Securities.Account CD.Account     Online CreditCard
## 2 -0.5554684             0          2.9286223 -0.2535149 -1.2164961 -0.6452498
## 3 -0.5554684             0         -0.3413892 -0.2535149 -1.2164961 -0.6452498
## 4 -0.5554684             0         -0.3413892 -0.2535149 -1.2164961 -0.6452498
## 5 -0.5554684             0         -0.3413892 -0.2535149 -1.2164961  1.5494774
## 6  0.9684153             0         -0.3413892 -0.2535149  0.8218687 -0.6452498
## 7 -0.5554684             0         -0.3413892 -0.2535149  0.8218687 -0.6452498
```

```
#predict dataset from the above data given.
library(caret)
library(FNN)
```

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

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

```
sb.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(sb.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
```

```
sb.predict_Norm <- predict(M_norm,sb.predict)

predictions <- knn(train= as.data.frame(my_train.df[,1:7,9:12]),
                  test = as.data.frame(sb.predict_Norm[,1:7,9:12]),
                  cl= my_train.df$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)'
```

```
print(predictions)
```

```
## [1] 0
## attr(,"nn.index")
##      [,1]
## [1,]  411
## attr(,"nn.dist")
##           [,1]
## [1,] 0.2986486
```

```
## Levels: 0
```

```
#Question_2
#determining the K value that balances overfitting and underfitting from the data set

set.seed(123)
SB.Bank <- trainControl(method= "repeatedcv", number = 3, repeats = 2)
searchGrid = expand.grid(k=1:10)

knn.model = train(Personal.Loan~., data = my_train.df, method = 'knn', tuneGrid = searchGrid,trControl =

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  0.9525000  0.6960335
##    2  0.9481667  0.6681321
##    3  0.9550000  0.6873771
##    4  0.9515000  0.6593667
##    5  0.9510000  0.6514024
##    6  0.9501667  0.6455524
##    7  0.9486667  0.6263958
##    8  0.9470000  0.6135666
##    9  0.9466667  0.6092029
##   10  0.9448333  0.5905702
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

```
#perfect value of k is 3
#strikes a compromise between underfitting and overfitting of the data above.

#Question 3
#confusion Matrix is below
predictors_bank <- predict(knn.model,validate.sb.df)

confusionMatrix(predictors_bank,validate.sb.df$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1803   63
##          1    5  129
##
##                Accuracy : 0.966
```

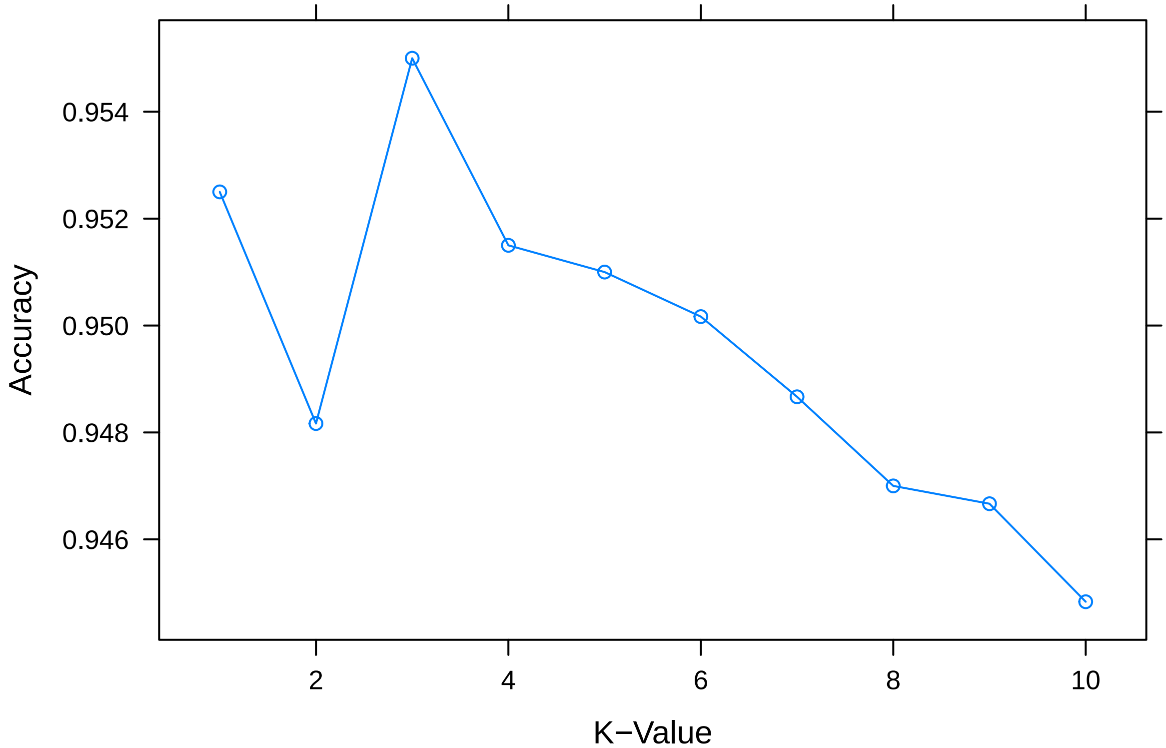
```
##                  95% CI : (0.9571, 0.9735)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7735
##
##  Mcnemar's Test P-Value : 4.77e-12
##
##             Sensitivity : 0.9972
##             Specificity : 0.6719
##          Pos Pred Value : 0.9662
##          Neg Pred Value : 0.9627
##              Prevalence : 0.9040
##          Detection Rate : 0.9015
##    Detection Prevalence : 0.9330
##       Balanced Accuracy : 0.8346
##
##        'Positive' Class : 0
##
```

```r
#The confustionmatrix has a 95.1% accuracy.
```

```r
#Question 4
#Levels
#using the best K to classify the consumer.
sb.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)
sb.predict_Norm = predict(M_norm, sb.predict)
predict(knn.model, sb.predict_Norm)
```

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

```r
#A plot that shows the best value of K (3), the one with the highest accuracy, is also present.
plot(knn.model, type = "b", xlab = "K-Value", ylab = "Accuracy")
```




```r
#Question 5
#creating training, test, and validation sets from the data collection.
t_size = 0.5 #training(50%)
sb_train_index = createDataPartition(sb.data$Personal.Loan, p = 0.5, list = FALSE)
my_train.df = sb.data_norm[sb_train_index,]


t.data_size = 0.2 #Test Data(20%)
Test.data_index = createDataPartition(sb.data$Personal.Loan, p = 0.2, list = FALSE)
t.data.df = sb.data_norm[Test.data_index,]


validation_size = 0.3 #validation(30%)
Validation.sb_index = createDataPartition(sb.data$Personal.Loan, p = 0.3, list = FALSE)
validate.sb.df = sb.data_norm[Validation.sb_index,]



Test.data.knn <- knn(train = my_train.df[,-8], test = t.data.df[,-8], cl = my_train.df[,8], k =3)
Validation.knn <- knn(train = my_train.df[,-8], test = validate.sb.df[,-8], cl = my_train.df[,8], k =3)
Training.knn <- knn(train = my_train.df[,-8], test = my_train.df[,-8], cl = my_train.df[,8], k =3)

confusionMatrix(Test.data.knn, t.data.df[,8])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 901  28
##          1   3  68
```

```
##
##                Accuracy : 0.969
##                  95% CI : (0.9563, 0.9788)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 1.027e-15
##
##                   Kappa : 0.7979
##
##  Mcnemar's Test P-Value : 1.629e-05
##
##             Sensitivity : 0.9967
##             Specificity : 0.7083
##          Pos Pred Value : 0.9699
##          Neg Pred Value : 0.9577
##              Prevalence : 0.9040
##          Detection Rate : 0.9010
##    Detection Prevalence : 0.9290
##       Balanced Accuracy : 0.8525
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(Validation.knn, validate.sb.df[,8])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1351   32
##          1    5  112
##
##                Accuracy : 0.9753
##                  95% CI : (0.9662, 0.9826)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8449
##
##  Mcnemar's Test P-Value : 1.917e-05
##
##             Sensitivity : 0.9963
##             Specificity : 0.7778
##          Pos Pred Value : 0.9769
##          Neg Pred Value : 0.9573
##              Prevalence : 0.9040
##          Detection Rate : 0.9007
##    Detection Prevalence : 0.9220
##       Balanced Accuracy : 0.8870
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(Training.knn, my_train.df[,8])
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    0    1
##          0 2255   59
##          1    5  181
##
##                Accuracy : 0.9744
##                  95% CI : (0.9674, 0.9802)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.836
##
##  Mcnemar's Test P-Value : 3.472e-11
##
##             Sensitivity : 0.9978
##             Specificity : 0.7542
##          Pos Pred Value : 0.9745
##          Neg Pred Value : 0.9731
##              Prevalence : 0.9040
##          Detection Rate : 0.9020
##    Detection Prevalence : 0.9256
##       Balanced Accuracy : 0.8760
##
##        'Positive' Class : 0
##
```

```
#Final Verdict: The training data have improved accuracy and sensitivity. According to the aforemention
```