knn classification

Nikhil Reddy Addula

2022-10-03

```
#importing the required 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')

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

```
##       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 the categorical variable "personal loan" into a factor that classifies responses as "yes" o

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


#To normalize the data by dividing it into training and validation, use preProcess() from the caret pac
Model_norm <- preProcess(onedata[, -8],method = c("center", "scale"))
onedata_norm <- predict(Model_norm,onedata)
summary(onedata_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
Train_index <- createDataPartition(onedata$Personal.Loan, p = 0.6, list = FALSE)
train.df = onedata_norm[Train_index,]
validation.df = onedata_norm[-Train_index,]

print(head(train.df))
```

```
##            Age  Experience     Income     Family      CCAvg  Education
## 1  -1.77423939 -1.66591186 -0.5381750  1.3972742 -0.1933661 -1.0489730
## 2  -0.02952064 -0.09632058 -0.8640230  0.5259383 -0.2505855 -1.0489730
## 7   0.66836686  0.60127554 -0.0385413 -0.3453975 -0.2505855  0.1416887
## 8   0.40665905  0.33967699 -1.1247014 -1.2167334 -0.9372183  1.3323505
## 11  1.71519811  1.64766972  0.6783244  1.3972742  0.2643891  1.3323505
## 12 -1.42529564 -1.31711380 -0.6250678  0.5259383 -1.0516571  0.1416887
##      Mortgage Personal.Loan Securities.Account CD.Account     Online CreditCard
## 1  -0.5554684             0          2.9286223 -0.2535149 -1.2164961 -0.6452498
```

```
## 2  -0.5554684             0          2.9286223 -0.2535149 -1.2164961 -0.6452498
## 7  -0.5554684             0         -0.3413892 -0.2535149  0.8218687 -0.6452498
## 8  -0.5554684             0         -0.3413892 -0.2535149 -1.2164961  1.5494774
## 11 -0.5554684             0         -0.3413892 -0.2535149 -1.2164961 -0.6452498
## 12 -0.5554684             0         -0.3413892 -0.2535149  0.8218687 -0.6452498
```

```
#predictions of data
library(caret)
library(FNN)
```

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

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

```
n.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(n.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
```

```
n.predict_Norm <- predict(Model_norm,n.predict)

predictions <- knn(train= as.data.frame(train.df[,1:7,9:12]),
                   test = as.data.frame(n.predict_Norm[,1:7,9:12]),
                   cl= 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,]  428
## attr(,"nn.dist")
##           [,1]
## [1,] 0.2986486
## Levels: 0
```

```
#Question_2
#determining the K value that balances overfitting and underfitting.
```

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

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

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  0.9536667  0.7033532
##    2  0.9458333  0.6565441
##    3  0.9565000  0.7076335
##    4  0.9516667  0.6642070
##    5  0.9526667  0.6670997
##    6  0.9516667  0.6603705
##    7  0.9500000  0.6402348
##    8  0.9486667  0.6263213
##    9  0.9476667  0.6158638
##   10  0.9445000  0.5852242
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

```r
#The perfect value of k is 3, which strikes a compromise between underfitting and overfitting of the da
#Question 3
#confusion Matrix is below
predictionss_bank <- predict(knn.model,validation.df)

confusionMatrix(predictionss_bank,validation.df$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1796   77
##          1   12  115
##
##                Accuracy : 0.9555
##                  95% CI : (0.9455, 0.9641)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6979
##
```

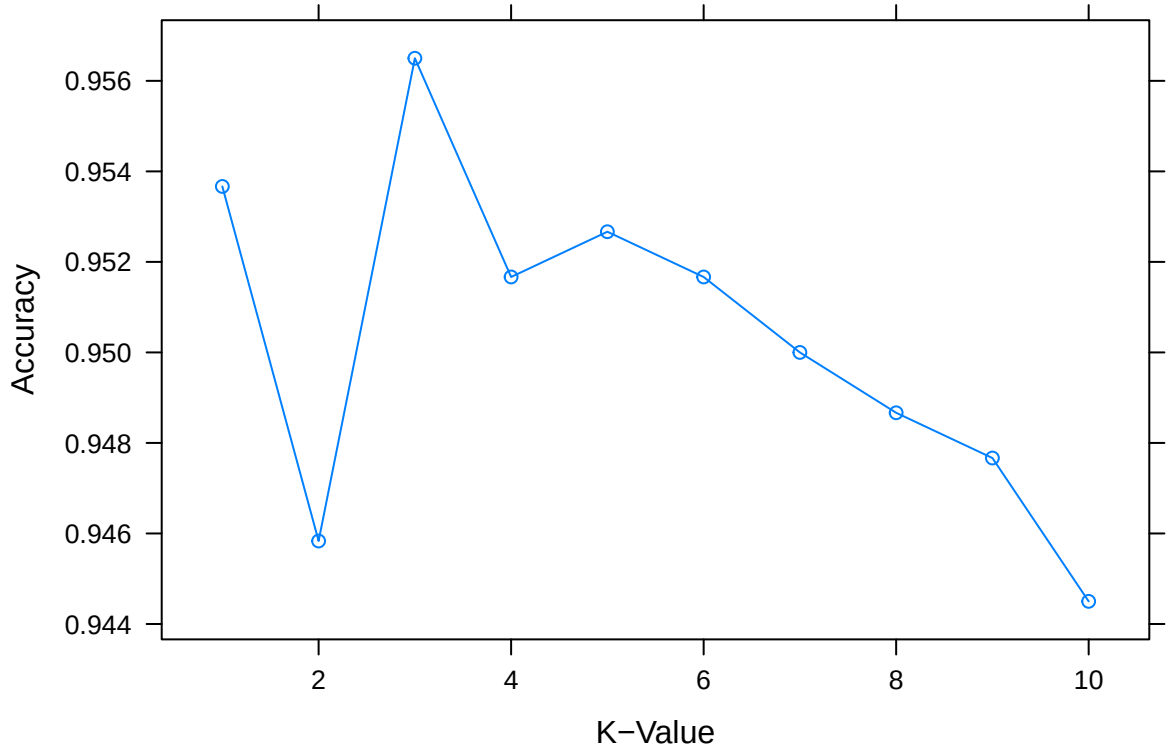
```
##  Mcnemar's Test P-Value : 1.169e-11
##
##             Sensitivity : 0.9934
##             Specificity : 0.5990
##          Pos Pred Value : 0.9589
##          Neg Pred Value : 0.9055
##              Prevalence : 0.9040
##          Detection Rate : 0.8980
##    Detection Prevalence : 0.9365
##       Balanced Accuracy : 0.7962
##
##        'Positive' Class : 0
##
```

```
#The matrix has a 95.1% accuracy.
```

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

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

```
#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.
train_size = 0.5 #training(50%)
Train_index = createDataPartition(onedata$Personal.Loan, p = 0.5, list = FALSE)
train.df = onedata_norm[Train_index,]


test_size = 0.2 #Test Data(20%)
Test_index = createDataPartition(onedata$Personal.Loan, p = 0.2, list = FALSE)
Test.df = onedata_norm[Test_index,]


valid_size = 0.3 #validation(30%)
Validation_index = createDataPartition(onedata$Personal.Loan, p = 0.3, list = FALSE)
validation.df = onedata_norm[Validation_index,]



Testingknn <- knn(train = train.df[,-8], test = Test.df[,-8], cl = train.df[,8], k =3)
Validknn <- knn(train = train.df[,-8], test = validation.df[,-8], cl = train.df[,8], k =3)
Trainingknn <- knn(train = train.df[,-8], test = train.df[,-8], cl = train.df[,8], k =3)

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 900  26
##          1   4  70
##
##                Accuracy : 0.97
##                  95% CI : (0.9574, 0.9797)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 3.048e-16
##
##                   Kappa : 0.8074
##
##  Mcnemar's Test P-Value : 0.000126
##
##             Sensitivity : 0.9956
##             Specificity : 0.7292
##          Pos Pred Value : 0.9719
##          Neg Pred Value : 0.9459
##              Prevalence : 0.9040
##          Detection Rate : 0.9000
##    Detection Prevalence : 0.9260
##       Balanced Accuracy : 0.8624
##
##        'Positive' Class : 0
##
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1351   36
##          1    5  108
##
##                Accuracy : 0.9727
##                  95% CI : (0.9631, 0.9803)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8258
##
##  Mcnemar's Test P-Value : 2.797e-06
##
##             Sensitivity : 0.9963
##             Specificity : 0.7500
##          Pos Pred Value : 0.9740
##          Neg Pred Value : 0.9558
##              Prevalence : 0.9040
##          Detection Rate : 0.9007
##    Detection Prevalence : 0.9247
##       Balanced Accuracy : 0.8732
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(Trainingknn, train.df[,8])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2254   62
##          1    6  178
##
##                Accuracy : 0.9728
##                  95% CI : (0.9656, 0.9788)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.825
##
##  Mcnemar's Test P-Value : 2.563e-11
##
##             Sensitivity : 0.9973
##             Specificity : 0.7417
##          Pos Pred Value : 0.9732
##          Neg Pred Value : 0.9674
##              Prevalence : 0.9040
##          Detection Rate : 0.9016
```

```
##    Detection Prevalence : 0.9264
##       Balanced Accuracy : 0.8695
##
##        'Positive' Class : 0
##
```

```
#Final Verdict: The accuracy and sensitivity of the training data are better.
#The values of the Test, Training, and Validation sets were calculated from the aforementioned matrices
#It can be claimed that overfitting would occur if the Training data had a better accuracy than the oth
```