ML Assignment 2 KNN

Pavan Chaitanya Bommadevara

2022-09-28

```
#Question 1:
#Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Education_1 =
0, Education_2 = 1, #Education_3 = 0, Mortgage = 0, Securities Account = 0, C
D Account = 0, Online = 1, and Credit Card #= 1.
#Perform a k-NN classification with all predictors except ID and ZIP code usi
ng k = 1.
#Remember to transform categorical predictors with more than two categories i
nto dummy #variables first.
#Specify the success class as 1 (loan acceptance), and use the default cut-of
f value of 0.5. How would #this customer be classified?
#Installing all the packages required and importing the data by using the rea
d.csv function
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')

Uni_Bank <- read.csv("C:/Users/Pavan Chaitanya/Downloads/UniversalBank.csv",
sep = ',' )

#Converting the  ID Column and ZIP Column as NULL as specified in the given Q
uestion.
Uni_Bank$ID <- NULL
Uni_Bank$ZIP.Code <- NULL
summary(Uni_Bank)
```

```
##       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
```

```
#Tranforming the categorial variable Personal Loan as a factor which can clas
sify the response as yes or no.
Uni_Bank$Personal.Loan =  as.factor(Uni_Bank$Personal.Loan)

#Firstly, Using the Normalization Method to normalize the data.
Normalized_Model <- preProcess(Uni_Bank[, -8],method = c("center", "scale"))
Uni_Bank_normalized <- predict(Normalized_Model,Uni_Bank)
summary(Uni_Bank_normalized)
```

```
##       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
```

```
#Partitiong the data into different sets like testing set and training set
Train_index <- createDataPartition(Uni_Bank$Personal.Loan, p = 0.6, list = FA
LSE)
train.df = Uni_Bank_normalized[Train_index,]
validation.df = Uni_Bank_normalized[-Train_index,]

#Performing the Prediction
To_Predict = data.frame(Age = 40, Experience = 10, Income = 84, Family = 2,
                        CCAvg = 2, Education = 1, Mortgage = 0, Securities.Ac
count =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
```

```
To_Predict_Normalized <- predict(Normalized_Model,To_Predict)

Prediction <- knn(train= train.df[,1:7,9:12],
                  test = To_Predict_Normalized[,1:7,9:12],
                  cl= train.df$Personal.Loan,
                  k=1)

print(Prediction)
```

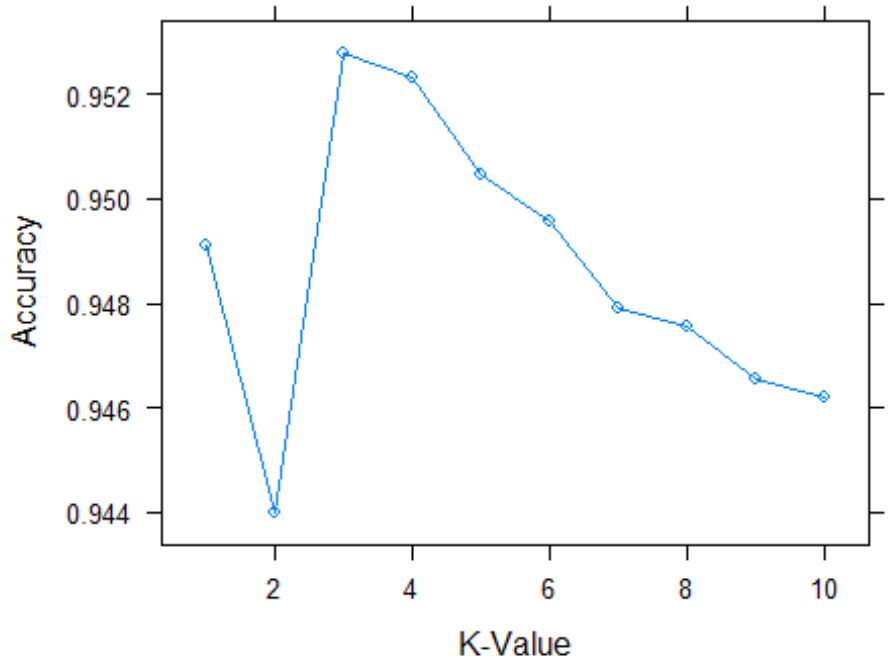
```
## [1] 0
## Levels: 0 1
```

```
#Question 2
#What is a choice of k that balances between overfitting and ignoring the pre
dictor information?

set.seed(123)

Uni_Bankcontrol <- trainControl(method= "repeatedcv", number = 3, repeats = 3
)
searchGrid = expand.grid(k=1:10)

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

knn.model
```

```
## k-Nearest Neighbors
##
## 3000 samples
##   11 predictor
```

```
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 3 times)
## Summary of sample sizes: 2000, 2000, 2000, 2000, 2000, 2000, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9491111  0.6722512
##    2  0.9440000  0.6343772
##    3  0.9527778  0.6726916
##    4  0.9523333  0.6642641
##    5  0.9504444  0.6442037
##    6  0.9495556  0.6362941
##    7  0.9478889  0.6170360
##    8  0.9475556  0.6140369
##    9  0.9465556  0.6032389
##   10  0.9462222  0.5988031
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

```
#Plotting the graph for the best fit.
plot(knn.model, type = "b", xlab = "K-Value", ylab = "Accuracy")
```




```
#Question 3
# Show the confusion matrix for the validation data that results from using t
```

```
he best k
predictions <- predict(knn.model,validation.df)

confusionMatrix(predictions,validation.df$Personal.Loan)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1799   67
##          1    9  125
##
##                Accuracy : 0.962
##                  95% CI : (0.9527, 0.9699)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7469
##
##  Mcnemar's Test P-Value : 6.22e-11
##
##             Sensitivity : 0.9950
##             Specificity : 0.6510
##          Pos Pred Value : 0.9641
##          Neg Pred Value : 0.9328
##              Prevalence : 0.9040
##          Detection Rate : 0.8995
##    Detection Prevalence : 0.9330
##       Balanced Accuracy : 0.8230
##
##        'Positive' Class : 0
##

#Question 4
# Consider the following customer: Age = 40, Experience = 10, Income = 84, Fa
mily = 2, CCAvg = 2, #Education_1 = 0, Education_2 = 1, Education_3 = 0, Mort
gage = 0, Securities Account = 0, CD #Account = 0, Online = 1 and Credit Card
= 1. Classify the customer using the best k
#Classifying the customer using the best K.
To_Predict_Normalized = data.frame(Age = 40, Experience = 10, Income = 84, Fa
mily = 2,CCAvg = 2, Education = 1, Mortgage = 0,Securities.Account =0,
CD.Account = 0, Online = 1,CreditCard = 1)
To_Predict_Normalized = predict(Normalized_Model, To_Predict)
predict(knn.model, To_Predict_Normalized)

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

#Question 5
# Repartition the data, this time into training, validation, and test sets (5
0% : 30% : 20%).
```

```r
#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 th
e differences and their reason.

#Partitioning the data into training set, testing set and validation sets.

#Training Set
train_size = 0.5
Train_index = createDataPartition(Uni_Bank$Personal.Loan, p = 0.5, list = FAL
SE)
train.df = Uni_Bank_normalized[Train_index,]

#Testing Set
test_size = 0.2
Test_index = createDataPartition(Uni_Bank$Personal.Loan, p = 0.2, list = FALS
E)
Test.df = Uni_Bank_normalized[Test_index,]

#Validation Set
valid_size = 0.3
Validation_index = createDataPartition(Uni_Bank$Personal.Loan, p = 0.3, list
= FALSE)
validation.df = Uni_Bank_normalized[Validation_index,]

# Applying the k-NN method with the chosen K.

Testknn <- knn(train = train.df[,-8], test = Test.df[,-8], cl = train.df[,8],
k =3)
Validationknn <- knn(train = train.df[,-8], test = validation.df[,-8], cl = t
rain.df[,8], k =3)
Trainknn <- knn(train = train.df[,-8], test = train.df[,-8], cl = train.df[,8
], k =3)

# Comparing the confusion matrix of the test set, training set and validation
sets

confusionMatrix(Testknn, 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(Trainknn, train.df[,8])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2253   52
##          1    7  188
##
##                Accuracy : 0.9764
##                  95% CI : (0.9697, 0.982)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8516
##
##  Mcnemar's Test P-Value : 1.014e-08
##
##             Sensitivity : 0.9969
##             Specificity : 0.7833
##          Pos Pred Value : 0.9774
##          Neg Pred Value : 0.9641
##              Prevalence : 0.9040
##          Detection Rate : 0.9012
##    Detection Prevalence : 0.9220
##       Balanced Accuracy : 0.8901
##
##        'Positive' Class : 0
##
```

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

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

```
##           Reference
## Prediction    0    1
##          0 1348   44
##          1    8  100
##
##                Accuracy : 0.9653
##                  95% CI : (0.9548, 0.974)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7751
##
##  Mcnemar's Test P-Value : 1.212e-06
##
##             Sensitivity : 0.9941
##             Specificity : 0.6944
##          Pos Pred Value : 0.9684
##          Neg Pred Value : 0.9259
##              Prevalence : 0.9040
##          Detection Rate : 0.8987
##    Detection Prevalence : 0.9280
##       Balanced Accuracy : 0.8443
##
##        'Positive' Class : 0
##

# From the above matrices we can say that the accuracy of the training set is
somewhat greater than the accuracy of the test and validation sets.
```