# Assignment-2

Bhavika

2024-02-26

```r
library(readr)
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(class)
```

```r
Data<-read_csv("UniversalBank.csv")
```

```
## Rows: 5000 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (14): ID, Age, Experience, Income, ZIP_Code, Family, CCAvg, Education, M...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(Data)
```

```
## # A tibble: 6 x 14
##      ID   Age Experience Income ZIP_Code Family CCAvg Education Mortgage
##   <dbl> <dbl>      <dbl>  <dbl>    <dbl>  <dbl> <dbl>     <dbl>    <dbl>
## 1     1    25          1     49    91107      4   1.6         1        0
## 2     2    45         19     34    90089      3   1.5         1        0
## 3     3    39         15     11    94720      1   1           1        0
## 4     4    35          9    100    94112      1   2.7         2        0
## 5     5    35          8     45    91330      4   1           2        0
## 6     6    37         13     29    92121      4   0.4         2      155
## # i 5 more variables: Personal_Loan <dbl>, Securities_Account <dbl>,
## #   CD_Account <dbl>, Online <dbl>, CreditCard <dbl>
```

```r
#converting personal loan to factor
Data$Personal_Loan <-as.factor(Data$Personal_Loan)
class(Data$Personal_Loan)
```

```
## [1] "factor"
```

```r
#creating dummy variables
education_1 <- ifelse(Data$Education==1,"1","0")
education_2 <- ifelse(Data$Education==2,"1","0")
education_3 <- ifelse(Data$Education==3,"1","0")

#combining dummy variables and original variables
Data <- cbind(Data,education_1,education_2,education_3)# cbind is used for combined the data
head(Data)
```

```
##   ID Age Experience Income ZIP_Code Family CCAvg Education Mortgage
## 1  1  25          1     49    91107      4   1.6         1        0
## 2  2  45         19     34    90089      3   1.5         1        0
## 3  3  39         15     11    94720      1   1.0         1        0
## 4  4  35          9    100    94112      1   2.7         2        0
## 5  5  35          8     45    91330      4   1.0         2        0
## 6  6  37         13     29    92121      4   0.4         2      155
##   Personal_Loan Securities_Account CD_Account Online CreditCard education_1
## 1             0                  1          0      0          0           1
## 2             0                  1          0      0          0           1
## 3             0                  0          0      0          0           1
## 4             0                  0          0      0          0           0
## 5             0                  0          0      0          1           0
## 6             0                  0          0      1          0           0
##   education_2 education_3
## 1           0           0
## 2           0           0
## 3           0           0
## 4           1           0
## 5           1           0
## 6           1           0
```

```r
#Removing education,Id and Zip Code variables
Data <-Data[-c(1,5,8)]
```

```r
#Normalization
norm_model <- preProcess(Data[,-(6:9)],method = c('scale','center'))
Data_normalized <- predict(norm_model,Data)
head(Data_normalized)
```

```
##           Age  Experience     Income     Family      CCAvg Mortgage
## 1 -1.77423939 -1.66591186 -0.5381750  1.3972742 -0.1933661        0
## 2 -0.02952064 -0.09632058 -0.8640230  0.5259383 -0.2505855        0
## 3 -0.55293627 -0.44511864 -1.3636566 -1.2167334 -0.5366825        0
## 4 -0.90188002 -0.96831574  0.5697084 -1.2167334  0.4360473        0
## 5 -0.90188002 -1.05551525 -0.6250678  1.3972742 -0.5366825        0
## 6 -0.72740814 -0.61951767 -0.9726390  1.3972742 -0.8799989      155
##   Personal_Loan Securities_Account CD_Account     Online CreditCard education_1
## 1             0                  1          0 -1.2164961 -0.6452498           1
## 2             0                  1          0 -1.2164961 -0.6452498           1
## 3             0                  0          0 -1.2164961 -0.6452498           1
## 4             0                  0          0 -1.2164961 -0.6452498           0
## 5             0                  0          0 -1.2164961  1.5494774           0
## 6             0                  0          0  0.8218687 -0.6452498           0
##   education_2 education_3
## 1           0           0
## 2           0           0
## 3           0           0
## 4           1           0
## 5           1           0
## 6           1           0
```

```
#Creating Test Data
TestData<-data.frame(Age=40,Experience=10,Income=84,Family=2,CCAvg=2,Mortgage=0,SecuritiesAccount=0,CDA

#Normalizing Test Data
TestData_normalized <- predict(norm_model,TestData)
View(TestData_normalized)
```

```
#Partition the data into training (60%) and validation (40%) sets.
spilt<-createDataPartition(Data_normalized$Personal_Loan,p=0.6, list=FALSE)
Training<-Data_normalized[spilt,]
validation<-Data_normalized[-spilt,] #(-) is used balance amount of total amount
```

```
#Building a KNN Model

Train_Predictor<-Training[,-7]
Train_label<- Training[,7]


Validation_Predictor<-validation[,-7]
Validation_label<- validation[,7]


KNN_Model<-knn(Train_Predictor,TestData_normalized,cl=Train_label,k=1)

KNN_Model
```

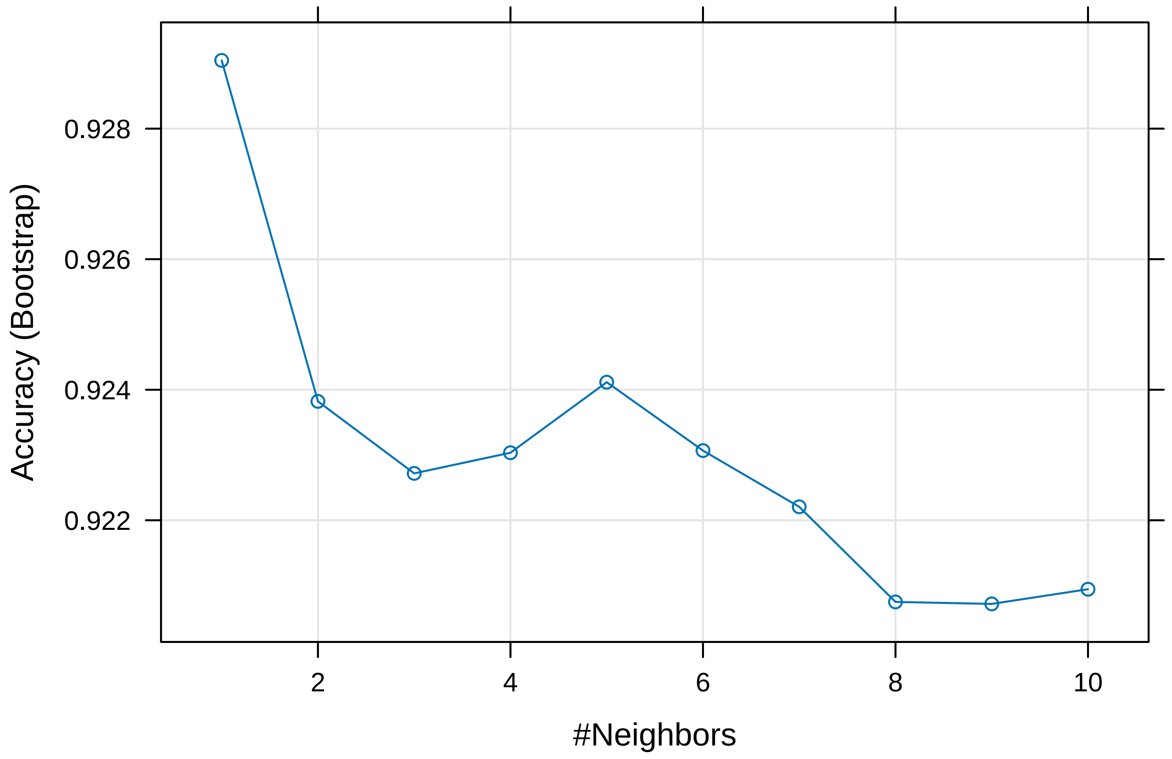
```
## [1] 0
## Levels: 0 1
```

```
#As the value is 0, customer in the test data will not accept the Personal Loan
```

```
#2. Finding the best choice of K

set.seed(2873)

searchGrid<- expand.grid(k=seq(1:10))

model_new<-train(Personal_Loan~Age+Experience+Income+Family+CCAvg+Mortgage+Securities_Account+CD_Accoun
                 data=Training,
                 method="knn",
                 tuneGrid=searchGrid)

model_new
```

```
## k-Nearest Neighbors
##
## 3000 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3000, 3000, 3000, 3000, 3000, 3000, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9290454  0.5269636
##    2  0.9238215  0.4872306
##    3  0.9227180  0.4654614
##    4  0.9230348  0.4557685
##    5  0.9241157  0.4448629
##    6  0.9230675  0.4195140
##    7  0.9222060  0.3976387
##    8  0.9207488  0.3713709
##    9  0.9207187  0.3622717
##   10  0.9209435  0.3560548
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```
best_k<-model_new$bestTune[[1]]

plot(model_new)
```



```r
#3.Confusion Matrix

Prediction_Model<-predict(model_new,validation[,-7])

confusionMatrix(Prediction_Model,Validation_label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1767   91
##          1   41  101
##
##                Accuracy : 0.934
##                  95% CI : (0.9222, 0.9445)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 1.029e-06
##
##                   Kappa : 0.5697
##
##  Mcnemar's Test P-Value : 2.000e-05
##
##             Sensitivity : 0.9773
##             Specificity : 0.5260
##          Pos Pred Value : 0.9510
##          Neg Pred Value : 0.7113
##               Prevalence : 0.9040
##           Detection Rate : 0.8835
##     Detection Prevalence : 0.9290
##        Balanced Accuracy : 0.7517
##
##         'Positive' Class : 0
##
```

```
#4

best_k_model<-knn(Train_Predictor,TestData_normalized,cl=Train_label,k=best_k)

best_k_model
```

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

```
#Conclusion: Customer will not accept the loan.
```

```
#5.
#Repartition the data into training, validation, and test sets (50% :    30% : 20%)
NewSplit <- createDataPartition(Data_normalized$Personal_Loan,p=0.5,list=FALSE)
NewTraining<-Data_normalized[NewSplit,]
Remaining <- Data_normalized[-NewSplit,]

SplitRemaining <- createDataPartition(Remaining$Personal_Loan,p=0.6,list=FALSE)
Newvalidation<-Remaining[SplitRemaining,]
NewTest <- Remaining[-SplitRemaining,]
```

```
NewTrain_Predictor<-NewTraining[,-7]
NewValidation_Predictor<-Newvalidation[,-7]
NewTest_Predictor<-NewTest[,-7]


NewTrain_label<- NewTraining[,7]
NewValidation_label<- Newvalidation[,7]
NewTest_label<- NewTest[,7]

NewKNN_Training<-knn(NewTrain_Predictor,NewTrain_Predictor,cl=NewTrain_label,k=best_k)

NewKNN_Validation<-knn(NewTrain_Predictor,NewValidation_Predictor,cl=NewTrain_label,k=best_k)

NewKnn_Test<-knn(NewTrain_Predictor,NewTest_Predictor,cl=NewTrain_label,k=best_k)


#Confusion Matrix of Training:

confusionMatrix(NewKNN_Training,NewTrain_label)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0    1
##          0 2260    0
##          1    0  240
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.904
##          Detection Rate : 0.904
##    Detection Prevalence : 0.904
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 0
##
```

```
#Confusion Matrix of Validation:

confusionMatrix(NewKNN_Validation,NewValidation_label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1333   75
##          1   23   69
##
##                Accuracy : 0.9347
##                  95% CI : (0.921, 0.9466)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 1.369e-05
##
##                   Kappa : 0.5512
##
##  Mcnemar's Test P-Value : 2.580e-07
##
##             Sensitivity : 0.9830
##             Specificity : 0.4792
##          Pos Pred Value : 0.9467
##          Neg Pred Value : 0.7500
##              Prevalence : 0.9040
##          Detection Rate : 0.8887
##    Detection Prevalence : 0.9387
##       Balanced Accuracy : 0.7311
##
```

```
##        'Positive' Class : 0
##
```

```r
#Confusion Matrix of Test:

confusionMatrix(NewKnn_Test,NewTest_label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 891  45
##          1  13  51
##
##                Accuracy : 0.942
##                  95% CI : (0.9257, 0.9557)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 8.744e-06
##
##                   Kappa : 0.6073
##
##  Mcnemar's Test P-Value : 4.691e-05
##
##             Sensitivity : 0.9856
##             Specificity : 0.5312
##          Pos Pred Value : 0.9519
##          Neg Pred Value : 0.7969
##              Prevalence : 0.9040
##          Detection Rate : 0.8910
##    Detection Prevalence : 0.9360
##       Balanced Accuracy : 0.7584
##
##        'Positive' Class : 0
##
```

#Interpretation and Conclusion:

#While comparing the confusion Matrix of the test data with that of the training and validation data, we can exclude training data as the accuracy is 100% which is because the model has seen the data already, so it classified every customer of accepting/rejecting the personal loan offer accurately.

#By comparing Accuracy of Validation and Test data from the confusion Matrix, it can be noticed that the Validation data has better accuracy of 94.1 compared to test data accuracy of 93.4

#When comparing other metrics like sensitivity and specificity, model of the validation data performed better with 98.1 and 56.2 respectively.

#Even though both Validation and Test data is unseen to the model, it has performed better on the Validation due to the more number of observations which helps model to identify customers accurately.