Assignment_2 FML

Rajeswari Theegala

2023-10-01

```r
#Import the necessary packages
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')
```

```r
UniversalBankData <- read.csv("C:/Users/shiva/Downloads/UniversalBank.csv")

UniversalBankData$ID <- NULL
UniversalBankData$ZIP.Code <- NULL
summary(UniversalBankData)
```

```
##       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
```

```
#A new data collection's "ID" and "ZIP Code" fields being ignored


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


Normalized_model <- preProcess(UniversalBankData[, -8],method = c("center", "scale"))
Bank_normalized <- predict(Normalized_model,UniversalBankData)
summary(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
```

```
#60% of the data is used for training, while 40% is used for testing.

Train_index <- createDataPartition(UniversalBankData$Personal.Loan, p = 0.6, list = FALSE)
train.df = Bank_normalized[Train_index,]
validation.df = Bank_normalized[-Train_index,]
```

```
#Predict
To_Predict = data.frame(Age = 40, Experience = 10, Income = 84, Family = 2,
                        CCAvg = 2, Education = 1, Mortgage = 0, Securities.Account =
```

```r
                          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
```

```r
To_Predict_Normalized <- predict(Normalized_model,To_Predict)
Prediction <- knn(train= train.df[, 1:10,11:11],
                  test = To_Predict_Normalized[,1:10,11:11],
                  cl=train.df$Personal.Loan,
                  k=1
                  )
print(Prediction)
```

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

```r
#Task2

#K=3 seems to be the ideal value of K, which achieves a balance between overfitting and ignoring predic

set.seed(123)
Bankcontrol <- 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 = Ba

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  0.9523333  0.6910618
##    2  0.9493333  0.6655549
##    3  0.9548333  0.6838594
##    4  0.9530000  0.6706485
##    5  0.9536667  0.6682236
##    6  0.9521667  0.6578821
##    7  0.9493333  0.6288027
##    8  0.9465000  0.6028787
##    9  0.9445000  0.5837140
```

```
##   10  0.9440000  0.5734513
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

```r
#Question 3
#For the Validation data, a confusion matrix

predictions <- predict(knn.model,validation.df)

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1792   68
##          1   16  124
##
##                Accuracy : 0.958
##                  95% CI : (0.9483, 0.9664)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7247
##
##  Mcnemar's Test P-Value : 2.628e-08
##
##             Sensitivity : 0.9912
##             Specificity : 0.6458
##          Pos Pred Value : 0.9634
##          Neg Pred Value : 0.8857
##              Prevalence : 0.9040
##          Detection Rate : 0.8960
##    Detection Prevalence : 0.9300
##       Balanced Accuracy : 0.8185
##
##        'Positive' Class : 0
##
```

```r
#Question 4

To_Predict_Normalization = 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)
To_Predict_Normalization = predict(Normalized_model, To_Predict)
predict(knn.model, To_Predict_Normalization)
```

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

```r
#Question 5
#Dividing the data into three categories: training (50%), validation (30%), and test (20%)
train_size = 0.5
Train_index = createDataPartition(UniversalBankData$Personal.Loan, p = 0.5, list = FALSE)
train.df = Bank_normalized[Train_index,]


test_size = 0.2
Test_index = createDataPartition(UniversalBankData$Personal.Loan, p = 0.2, list = FALSE)
Test.df = Bank_normalized[Test_index,]


valid_size = 0.3
Validation_index = createDataPartition(UniversalBankData$Personal.Loan, p = 0.3, list = FALSE)
validation.df = Bank_normalized[Validation_index,]



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 = train.df[,8], k =3)
Trainknn <- knn(train = train.df[,-8], test = train.df[,-8], cl = train.df[,8], k =3)

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 900  28
##          1   4  68
##
##                Accuracy : 0.968
##                  95% CI : (0.9551, 0.978)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 3.349e-15
##
##                   Kappa : 0.7924
##
##  Mcnemar's Test P-Value : 4.785e-05
##
##             Sensitivity : 0.9956
##             Specificity : 0.7083
##          Pos Pred Value : 0.9698
##          Neg Pred Value : 0.9444
##              Prevalence : 0.9040
##          Detection Rate : 0.9000
##    Detection Prevalence : 0.9280
##       Balanced Accuracy : 0.8520
##
##        'Positive' Class : 0
##
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2255   58
##          1    5  182
##
##                Accuracy : 0.9748
##                  95% CI : (0.9679, 0.9806)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8389
##
##  Mcnemar's Test P-Value : 5.701e-11
##
##             Sensitivity : 0.9978
##             Specificity : 0.7583
##          Pos Pred Value : 0.9749
##          Neg Pred Value : 0.9733
##              Prevalence : 0.9040
##          Detection Rate : 0.9020
##    Detection Prevalence : 0.9252
##       Balanced Accuracy : 0.8781
##
##        'Positive' Class : 0
##
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1349   30
##          1    7  114
##
##                Accuracy : 0.9753
##                  95% CI : (0.9662, 0.9826)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.847
##
##  Mcnemar's Test P-Value : 0.0002983
##
##             Sensitivity : 0.9948
##             Specificity : 0.7917
##          Pos Pred Value : 0.9782
##          Neg Pred Value : 0.9421
##              Prevalence : 0.9040
```

```
##          Detection Rate : 0.8993
##    Detection Prevalence : 0.9193
##       Balanced Accuracy : 0.8933
##
##        'Positive' Class : 0
##
```

#The fact that the training set's accuracy is a little higher than the test and validation sets' accuracy shows that the algorithm is working as it should.