

Assignment2

Durga Chowdary Koduru Lokesh

2023-10-01

```
library('caret')
```

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

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

```
## Warning: package 'lattice' was built under R version 4.2.2
```

```
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')
```

```
# Ignoring exsisting variables and creating a new dataset
```

```
UniversalBankData = read.csv("C:/Durga/UniversalBank (1).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
```

```
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
```

```
#partitioning into 60% for training dataset and 40% for testing dataset
```

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

```
#Prediction
```

```
To_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(To_Predict)

```

```

## Age Experience Income Family CCAvg Education Mortgage Securities.Account
## 1 40 10 84 2 2 1 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)

```

```

## 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(Prediction)

```

```

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

```

#Question 2

```

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 = Bankcontrol)
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 0.9526667 0.6968427
## 2 0.9505000 0.6825859
## 3 0.9571667 0.7087117
## 4 0.9543333 0.6858030
## 5 0.9550000 0.6861324
## 6 0.9528333 0.6687228
## 7 0.9515000 0.6558531
## 8 0.9518333 0.6571372
## 9 0.9491667 0.6331174
## 10 0.9475000 0.6149870
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

#Question3

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

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 1797   79
##           1   11  113
##
##           Accuracy : 0.955
##           95% CI : (0.945, 0.9637)
##           No Information Rate : 0.904
##           P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.692
##
## Mcnemar's Test P-Value : 1.636e-12
##
##           Sensitivity : 0.9939
##           Specificity : 0.5885
##           Pos Pred Value : 0.9579
##           Neg Pred Value : 0.9113
##           Prevalence : 0.9040
##           Detection Rate : 0.8985
##           Detection Prevalence : 0.9380
##           Balanced Accuracy : 0.7912
##
```

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

#Question4

```
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
```

#Question5

```
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  32
##          1   4  64
##
##          Accuracy : 0.964
##          95% CI : (0.9505, 0.9747)
##    No Information Rate : 0.904
##    P-Value [Acc > NIR] : 2.787e-13
##
##          Kappa : 0.7615
##
```

```
## McNemar's Test P-Value : 6.795e-06
##
##      Sensitivity : 0.9956
##      Specificity : 0.6667
##      Pos Pred Value : 0.9657
##      Neg Pred Value : 0.9412
##      Prevalence : 0.9040
##      Detection Rate : 0.9000
##      Detection Prevalence : 0.9320
##      Balanced Accuracy : 0.8311
##
##      'Positive' Class : 0
##
```

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

```
## Confusion Matrix and Statistics
##
##      Reference
## Prediction    0    1
##      0 2255    63
##      1    5   177
##
##      Accuracy : 0.9728
##      95% CI : (0.9656, 0.9788)
##      No Information Rate : 0.904
##      P-Value [Acc > NIR] : < 2.2e-16
##
##      Kappa : 0.8243
##
## McNemar's Test P-Value : 4.77e-12
##
##      Sensitivity : 0.9978
##      Specificity : 0.7375
##      Pos Pred Value : 0.9728
##      Neg Pred Value : 0.9725
##      Prevalence : 0.9040
##      Detection Rate : 0.9020
##      Detection Prevalence : 0.9272
##      Balanced Accuracy : 0.8676
##
##      'Positive' Class : 0
##
```

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

```
## Confusion Matrix and Statistics
##
##      Reference
## Prediction    0    1
##      0 1347    38
##      1    9   106
##
```

```

##           Accuracy : 0.9687
##           95% CI   : (0.9585, 0.9769)
##    No Information Rate : 0.904
##    P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa   : 0.8016
##
##    McNemar's Test P-Value : 4.423e-05
##
##           Sensitivity : 0.9934
##           Specificity : 0.7361
##           Pos Pred Value : 0.9726
##           Neg Pred Value : 0.9217
##           Prevalence : 0.9040
##           Detection Rate : 0.8980
##           Detection Prevalence : 0.9233
##           Balanced Accuracy : 0.8647
##
##           'Positive' Class : 0
##

```

#From the above data it can be determined that Training accuracy is slightly higher than the test and v