Assignment 2

Meghana gitay 09/29/2023

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

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

```
## Warning in register(): Can't find generic `scale_type` in package ggplot2 to ## register S3 method.
```

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

```
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 data set universal bank csv file
UniversalBank <- read.csv("Downloads/machine learning/assignment _2/UniversalBank.csv")
#assigning colnames colnames(UniversalBank)
```

```
## [1] "ID"                "Age"               "Experience"
## [4] "Income"            "ZIP.Code"          "Family"
## [7] "CCAvg"             "Education"         "Mortgage"
## [10] "Personal.Loan"    "Securities.Account" "CD.Account"
## [13] "Online"           "CreditCard"
```

```
#getting rid of column names id and zip code
UniversalBank$ID = NULL UniversalBank$ZIP.Code = NULL
summary(UniversalBank)
```

```
##       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
```

```
#Dummy Variable
UniversalBank$Personal.Loan = as.factor(UniversalBank$Personal.Loan)


Model_range_normalized <- preProcess(UniversalBank,method = "range")
UniversalBank_norm <- predict(Model_range_normalized,UniversalBank) summary(UniversalBank_norm)
```

```
##       Age          Experience         Income          Family
## Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
## 1st Qu.:0.2727    1st Qu.:0.2826   1st Qu.:0.1435   1st Qu.:0.0000
## Median :0.5000    Median :0.5000   Median :0.2593   Median :0.3333
## Mean   :0.5077   Mean   :0.5023   Mean   :0.3045   Mean   :0.4655
## 3rd Qu.:0.7273    3rd Qu.:0.7174   3rd Qu.:0.4167   3rd Qu.:0.6667
## Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##      CCAvg          Education        Mortgage       Personal.Loan
## Min.   :0.0000   Min.   :0.0000   Min.   :0.00000  0:4520
## 1st Qu.:0.0700    1st Qu.:0.0000   1st Qu.:0.00000  1: 480
## Median :0.1500    Median :0.5000   Median :0.00000

## Mean   :0.1938   Mean   :0.4405   Mean   :0.08897

## 3rd Qu.:0.2500    3rd Qu.:1.0000   3rd Qu.:0.15906

## Max.   :1.0000   Max.   :1.0000   Max.   :1.00000

## Securities.Account  CD.Account       Online         CreditCard
## Min.   :0.0000   Min.   :0.0000Min.   :0.0000Min.   :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   :0.0604Mean   :0.5968Mean   :0.294
## 3rd Qu.:0.0000    3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000
## Max.   :1.0000   Max.   :1.0000Max.   :1.0000Max.   :1.000
```

```
View(UniversalBank_norm)

#Data Partition into testing and training sets
Train_index <- createDataPartition(UniversalBank$Personal.Loan, p = 0.6, list = FALSE) train.df =
UniversalBank_norm[Train_index,] validation.df = UniversalBank_norm[-Train_index,]

#Question 1 (Perform k-nn classification with all the predictors expect id and zip code using k=1 )
To_Predict = data.frame(Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, 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_norm <- predict(Model_range_normalized,To_Predict)

Prediction <- knn(train = train.df[,1:7], test = To_Predict[,1:7], cl = print(Prediction)
```

```
train.df$Personal.Loan, k = 1)
```

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

```
#Question 2 (reducing the effects of underfitting and overfitting) set.seed(123)
UniversalBankcontrol <- 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, knn.model
```

```
trControl =
```

```
## 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 0.9561667 0.7231217 ##2
0.9498333 0.6816410 ##  3 0.9533333
0.6814059 ##  4 0.9493333
0.6458532 ##  5 0.9513333
0.6503733 ##  6 0.9483333
0.6241498 ##  7 0.9458333
0.5993678 ##  8 0.9441667
0.5843972 ##  9 0.9418333
0.5560415 ##  10 0.9383333
0.5168398
##
```

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 1.

```
#Question 3 (confusion matrix for the validation data that results from using the best k) predictions <-
predict(knn.model, validation.df) confusionMatrix(predictions, validation.df$Personal.Loan)
```

Confusion Matrix and Statistics

##

```
                    Reference
## Prediction         0    1
##          0 1790   67 ##     1
18 125
##
##                               Accuracy : 0.9575
##                                 95% CI : (0.9477, 0.9659)
##               No Information Rate : 0.904
##                 P-Value [Acc > NIR] : < 2.2e-16
##
##                                Kappa : 0.7236
##
## Mcnemar's Test P-Value : 1.926e-07
##
##          Sensitivity : 0.9900 ##       Specificity :
0.6510 ##           Pos Pred Value : 0.9639 ##
Neg Pred Value : 0.8741 ##             Prevalence :
0.9040 ##           Detection Rate : 0.8950
##          Detection Prevalence : 0.9285 ##
Balanced Accuracy : 0.8205
##
##                    'Positive' Class : 0
##
```

```
#Question 4 (classify the following customers)
To_Predict_norm = data.frame(Age = 40, Experience = 10, Income = 84, family = 2, CCAvg = 2                                    , Education =
To_Predict_norm = predict(Model_range_normalized, To_Predict) predict(knn.model,
To_Predict_norm)
```

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

```
#Question 5 (confusion matrix of the test set with that of the training and validation sets) train_size = 0.5
Train_index = createDataPartition(UniversalBank$Personal.Loan, p = 0.5, list = FALSE) train.df =
UniversalBank_norm[Train_index,]

test_size = 0.2
Test_index = createDataPartition(UniversalBank$Personal.Loan, p = 0.2, list = FALSE)
Test.df = UniversalBank_norm[Train_index,]                                                    k =1)

valid_size = 0.3
validation_index = createDataPartition(UniversalBank$Personal.Loan, p = 0.3, list = FALSE) validation.df =
UniversalBank_norm[validation_index,]

Trainknn = knn(train=train.df[,-8], test = train.df[,-8], cl = train.df[,8], k =1)
Testknn <- knn(train = train.df[,-8], test = Test.df[,-8], cl = train.df[,8], k =1) Validationknn <- knn(train = train.df[,-8],
test = validation.df[,-8], cl = train.df[,8], confusionMatrix(Trainknn, train.df[,8])
```

```
## 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
##
```

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

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

## Confusion Matrix and Statistics
##
##                       Reference
## Prediction              0        1
## 0 1348 23 ## 1 8 121
##
##                                  Accuracy : 0.9793
##                                    95% CI : (0.9708, 0.9859)
##               No Information Rate : 0.904
##                P-Value [Acc > NIR] : < 2e-16
##
##                                  Kappa : 0.8751
##
## Mcnemar's Test P-Value : 0.01192
##
##          Sensitivity : 0.9941 ##          Specificity :
0.8403 ##             Pos Pred Value : 0.9832 ##
Neg Pred Value : 0.9380 ##             Prevalence :
0.9040 ##             Detection Rate : 0.8987
##          Detection Prevalence : 0.9140 ##
Balanced Accuracy : 0.9172
##
##                       'Positive' Class : 0
##
#conclusion comment: From the above matrices, we can see that the accuracies of Testing
#and Training sets are exactly equal which means the algorithm is doing #what it is supposed to do that is avoiding
overfitting or underfitting.
```