Assignment 2

Lukas van der Watt

10/1/2021

```
#Get rid of ID and Zip Code
#Set Education to #as.factor
```

```
UBdata <- read.csv("UniversalBank.csv") #Reading the data file into R.
head(UBdata) #Showcasing the upper part of the data for a quick look.
```

```
##   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
## 1             0                  1          0      0          0
## 2             0                  1          0      0          0
## 3             0                  0          0      0          0
## 4             0                  0          0      0          0
## 5             0                  0          0      0          1
## 6             0                  0          0      1          0
```

```
#Loading the required packages required for all the commands used.
library(dummies)
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```
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)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(ISLR)
```

```
UBdata$Education = as.factor(UBdata$Education) #converting the education column to as.factor to be able

#Creating Dummy model using Categorical Variables in the original data effectively making a copy of the
Dummy_model=dummy.data.frame(select(UBdata,-c(ZIP.Code,ID)))#creating the model and Removing the ID and
```

```
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = FALSE):
## non-list contrasts argument ignored
```

```
head(Dummy_model)
```

```
##   Age Experience Income Family CCAvg Education1 Education2 Education3 Mortgage
## 1  25          1     49      4   1.6          1          0          0        0
## 2  45         19     34      3   1.5          1          0          0        0
## 3  39         15     11      1   1.0          1          0          0        0
## 4  35          9    100      1   2.7          0          1          0        0
## 5  35          8     45      4   1.0          0          1          0        0
## 6  37         13     29      4   0.4          0          1          0      155
##   Personal.Loan Securities.Account CD.Account Online CreditCard
## 1             0                  1          0      0          0
## 2             0                  1          0      0          0
## 3             0                  0          0      0          0
## 4             0                  0          0      0          0
## 5             0                  0          0      0          1
## 6             0                  0          0      1          0
```

```
Dummy_model$Personal.Loan=as.factor(Dummy_model$Personal.Loan) #Using the Personal Loan column in the d
Dummy_model$CCAvg=as.integer(Dummy_model$CCAvg)#Using the CCAvg column in the dummy model and convertin
#dummy_model <- dummyVars(UBdata[,(2:14)],data = UBdata)
```

```
# Partitioning the data into the training (60%) and validation(40%) sets
library(caret) #loading a package
set.seed(111) #setting a seed for Randomizing a sample set of the dummy data
train_ind  <- sample(row.names(Dummy_model),0.6*dim(Dummy_model)) #creating the train index & using 60%
valid_ind  <- sample(row.names(Dummy_model),train_ind) #Creating the validation index sample of the dum
train.data <- Dummy_model[train_ind, ]#Making the train data to be used in calculation of Knn algorithm
valid.data <- Dummy_model[valid_ind, ]#Making the validation data to be used in calculation of Knn algo
#Creating the Test Set Data of a customer provided.
#Establishing the integer values of the varible colums to be used in the dummy model.
Age <- (40)
Experience <- as.integer(10)
Income <- as.integer(84)
```

```
Family <- as.integer(2)
CCAvg <- as.integer(2)
Education1 <- as.integer(0)
Education2 <- as.integer(1)
Education3 <- as.integer(0)
Mortgage <-as.integer(0)
Securities.Account <- as.integer(0)
CD.Account <- as.integer(0)
Online <- as.integer(1)
CreditCard <- as.integer(1)

#Creating a new dataframe with the test data.
new.df <- data.frame(Age,Experience, Income, Family, CCAvg,Education1,Education2,Education3,Mortgage  ,

#Note: Only Normalize once on the training data set and then start predicting.
#Normalizing! Using the centralized method which is relevant to the problem of predicting weather a cus

Norm<- preProcess(train.data,method = c("center","scale"))

#Predicting for the Train, validation and test sets of the normalized dummy model.
train.norm <- predict(Norm,train.data)
valid.norm <- predict(Norm,valid.data)
test.norm <- predict(Norm,new.df)

#Modeling the K nearest neighbor
K1 <-knn(train = train.norm[,-c(10)], test = test.norm , cl = train.norm[,10],k=3,prob=TRUE)
#Had to change the length of the data to be able to run the command.
#Note: In general  5% error is common in the analasis and by shortening the normalized train data we ha
knn.attribute <- attributes(K1)
knn.attribute[3]
```

```
## $prob
## [1] 1
```

Problem 3

```
#Showcasing the confusion matrix after testing for the best k value of 1. An Accuracy of 98.37 was achi
K2 <-knn(train = train.norm[,-10], test = valid.norm[,-10], cl = train.norm[,10],k=1,prob=TRUE)
confusionMatrix(K2,valid.norm[,10])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 893  11
##          1   5  70
##
##                Accuracy : 0.9837
##                  95% CI : (0.9736, 0.9906)
##     No Information Rate : 0.9173
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8886
```

```
##
##  Mcnemar's Test P-Value : 0.2113
##
##             Sensitivity : 0.9944
##             Specificity : 0.8642
##          Pos Pred Value : 0.9878
##          Neg Pred Value : 0.9333
##              Prevalence : 0.9173
##          Detection Rate : 0.9122
##    Detection Prevalence : 0.9234
##       Balanced Accuracy : 0.9293
##
##        'Positive' Class : 0
##
```

The Best K in this problem would be a k value of 1 Problem 4

```
CUS4 = data.frame(Age = 40, Experience = 10, Income = 84,Family = 2, CCAvg = 2, Education1 = 0, Educatio

K3 <-knn(train = train.norm[,-10], test = CUS4 , cl = train.norm[,10],k=3,prob=TRUE)
K3
```

```
## [1] 1
## attr(,"prob")
## [1] 0.6666667
## Levels: 0 1
```

We can see here that the highest accuracy/probability is a k factor of 3. The customer will be likely to accept the loan offer. Note: This does seem a little low. Problem 5

```
#This is basically repeating the code and process of the upper parts only with a 50%, 30% and 20% data f
library(caret)
set.seed(111)
train_ind  <- sample(row.names(Dummy_model),0.5*dim(Dummy_model))
valid_ind  <- sample(setdiff(row.names(Dummy_model),train_ind),0.3*dim(Dummy_model)[1])
test_ind <- setdiff(row.names(Dummy_model),union(train_ind,valid_ind))
train.data <- Dummy_model[train_ind, ]
valid.data <- Dummy_model[valid_ind, ]
test.data <- Dummy_model[test_ind, ]

Norm<- preProcess(train.data,method = c("center","scale"))

train.norm <- predict(Norm,train.data)
valid.norm <- predict(Norm,valid.data)
test.norm <- predict(Norm,test.data)

Test.Knn <-knn(train = train.norm[,-c(10)], test = test.norm[,-c(10)] , cl = train.norm[,10],k=1,prob=TR
Valid.Knn <-knn(train = train.norm[,-c(10)], test = valid.norm[,-c(10)] , cl = train.norm[,10],k=3,prob=
Train.Knn <-knn(train = train.norm[,-c(10)], test = train.norm[,-c(10)] , cl = train.norm[,10],k=3,prob=

confusionMatrix(Test.Knn,test.norm[,10])
```

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

```
##
##           Reference
## Prediction   0   1
##          0 899  31
##          1   8  62
##
##                Accuracy : 0.961
##                  95% CI : (0.9471, 0.9721)
##     No Information Rate : 0.907
##     P-Value [Acc > NIR] : 4.071e-11
##
##                   Kappa : 0.74
##
##  Mcnemar's Test P-Value : 0.000427
##
##             Sensitivity : 0.9912
##             Specificity : 0.6667
##          Pos Pred Value : 0.9667
##          Neg Pred Value : 0.8857
##              Prevalence : 0.9070
##          Detection Rate : 0.8990
##    Detection Prevalence : 0.9300
##       Balanced Accuracy : 0.8289
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(Valid.Knn,valid.norm[,10])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1357   48
##          1    4   91
##
##                Accuracy : 0.9653
##                  95% CI : (0.9548, 0.974)
##     No Information Rate : 0.9073
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7597
##
##  Mcnemar's Test P-Value : 2.476e-09
##
##             Sensitivity : 0.9971
##             Specificity : 0.6547
##          Pos Pred Value : 0.9658
##          Neg Pred Value : 0.9579
##              Prevalence : 0.9073
##          Detection Rate : 0.9047
##    Detection Prevalence : 0.9367
##       Balanced Accuracy : 0.8259
##
```

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

```
confusionMatrix(Train.Knn,train.norm[,10])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2249   64
##          1    3  184
##
##                Accuracy : 0.9732
##                  95% CI : (0.9661, 0.9792)
##     No Information Rate : 0.9008
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8316
##
##  Mcnemar's Test P-Value : 2.299e-13
##
##             Sensitivity : 0.9987
##             Specificity : 0.7419
##          Pos Pred Value : 0.9723
##          Neg Pred Value : 0.9840
##              Prevalence : 0.9008
##          Detection Rate : 0.8996
##    Detection Prevalence : 0.9252
##       Balanced Accuracy : 0.8703
##
##        'Positive' Class : 0
##
```

The K values for the 3 confusion matrices is set to the optimal accuracy. The train set is 96.1%, Validation is at 96.53% accuracy and the test set is at 97.32%. This is the highest of the three which is good however a small percentage error does seem to be present. We would like the accuracy to be higher around 99%.