Universal Bank Data Analysis

Chathurani Ekanayake

2023-02-19

01.

```
library(readr)
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(psych)

##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha

install.packages("psych")

## Warning: package 'psych' is in use and will not be installed

library(psych)
library(class)
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(FNN)

##
## Attaching package: 'FNN'
```

```
## The following objects are masked from 'package:class':
##
##     knn, knn.cv
```

```r
# Get the Data set
library(readxl)
data <- read_excel("F:/1st sem/ML/Assignment 02/originaldata.xlsx")
View(data)

#Remove ID & Zip code
D_frame <- data[, -c(1,5)]
View(D_frame)

#Convert Education into Dummy Variables
dummy <- as.data.frame(dummy.code(D_frame$Education))
names(dummy) <- c("Education_01", "Education_02","Education_03")

#Remove Education column from the Data set
new_dataset<- D_frame[-c(6)]
UniBank_data <- cbind(new_dataset, dummy)
View(UniBank_data)
names(UniBank_data)[8] ="Securities.Account"
names(UniBank_data)[9] ="CD.Account"
names(UniBank_data)[7] ="Personal.Loan"
View(UniBank_data)

#Divide the Data set into 60%
set.seed(1)
train.index <- sample(row.names(UniBank_data), 0.6*dim(UniBank_data)[1])
test.index <- setdiff(row.names(UniBank_data), train.index)
train.df <- UniBank_data[train.index, ]
valid.df <- UniBank_data[test.index, ]

#customer details
new.df = data.frame(Age=40 , Experience=10, Income = 84, Family = 2, CCAvg =
2, Mortgage = 0, Securities.Account = 0, CD.Account = 0, Online = 1,
CreditCard = 1, Education_01 = 0, Education_02 = 1, Education_03 = 0)

#Normalizing Data
norm.values <- preProcess(train.df[, -c(7)], method=c("center", "scale"))
train.df[, -c(7)] <- predict(norm.values, train.df[, -c(7)])
valid.df[, -c(7)] <- predict(norm.values, valid.df[, -c(7)])
new.df <- predict(norm.values, new.df)

prediction <- knn(train = train.df[,-c(7)],test = new.df, cl = train.df[,7],
k=1, prob=TRUE)
knn.attri <- attributes(prediction)
knn.attri[3]
```

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

actual_data= valid.df$Personal.Loan
head(actual_data)

## [1] 0 0 0 0 1 0

#prediction_prob = att(prediction,"prob")
#table(prediction,actual_data)

mean(prediction==actual_data)

## [1] 0.8975
```

02. Find the accuracy Table

```
accuracy.df <- data.frame(k = seq(1, 30, 1), accuracy = rep(0, 30))
for(i in 1:30) {
  prediction <- knn(train = train.df[,-7], test = valid.df[-7],
                    cl = train.df[,7], k = i, prob=TRUE)
  accuracy.df[i,2] <- mean(prediction==actual_data)
}

accuracy.df

##     k accuracy
## 1   1   0.9630
## 2   2   0.9570
## 3   3   0.9640
## 4   4   0.9550
## 5   5   0.9605
## 6   6   0.9535
## 7   7   0.9580
## 8   8   0.9515
## 9   9   0.9535
## 10 10   0.9485
## 11 11   0.9495
## 12 12   0.9485
## 13 13   0.9500
## 14 14   0.9485
## 15 15   0.9485
## 16 16   0.9480
## 17 17   0.9500
## 18 18   0.9445
## 19 19   0.9460
## 20 20   0.9415
## 21 21   0.9450
## 22 22   0.9430
## 23 23   0.9430
```

```
## 24 24   0.9405
## 25 25   0.9400
## 26 26   0.9380
## 27 27   0.9400
## 28 28   0.9390
## 29 29   0.9400
## 30 30   0.9375
```

```
View(accuracy.df)
```

Largest accuracy value : k=3

03. Find Confusion Matrix using k=3

```
set.seed(123)
prediction <- knn(train = train.df[,-7], test = valid.df[,-7],
                  cl = train.df[,7], k = 3, prob=TRUE)
confusionMatrix(prediction, as.factor(valid.df[,7]))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 1786   63
##          1    9  142
## 
##                Accuracy : 0.964
##                  95% CI : (0.9549, 0.9717)
##     No Information Rate : 0.8975
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.7785
## 
##  Mcnemar's Test P-Value : 4.208e-10
## 
##             Sensitivity : 0.9950
##             Specificity : 0.6927
##          Pos Pred Value : 0.9659
##          Neg Pred Value : 0.9404
##              Prevalence : 0.8975
##          Detection Rate : 0.8930
##    Detection Prevalence : 0.9245
##       Balanced Accuracy : 0.8438
## 
##        'Positive' Class : 0
## 
```

04. Classify the customer

```
New_customer.df= data.frame(Age = 40, Experience = 10, Income = 84, Family =
2, CCAvg = 2, Education_01 = 0, Education_02 = 1, Education_03 = 0, Mortgage
= 0, Securities.Account = 0, CD.Account = 0, Online = 1, CreditCard = 1)
predict_01 <- knn(train = train.df[,-7],test = New_customer.df, cl =
train.df[,7], k=3, prob=TRUE)
predict_01
```

```
## [1] 1
## attr(,"prob")
## [1] 1
## attr(,"nn.index")
##      [,1] [,2] [,3]
## [1,] 2721  939 2146
## attr(,"nn.dist")
##          [,1]     [,2]     [,3]
## [1,] 90.49831 90.53126 90.53372
## Levels: 1
```

05. Partitioning Data into three parts

```
set.seed(1)
train.index <- sample(rownames(UniBank_data), 0.5*dim(UniBank_data)[1])

set.seed(1)
valid.index <- sample(setdiff(rownames(UniBank_data),train.index),
0.3*dim(UniBank_data)[1])
test.index = setdiff(rownames(UniBank_data), union(train.index, valid.index))

train.df <- UniBank_data[train.index, ]
valid.df <- UniBank_data[valid.index, ]
test.df <- UniBank_data[test.index, ]

norm.values <- preProcess(train.df[, -c(7)], method=c("center", "scale"))
train.df[, -c(7)] <- predict(norm.values, train.df[, -c(7)])
valid.df[, -c(7)] <- predict(norm.values, valid.df[, -c(7)])
test.df[,-c(7)] <- predict(norm.values, test.df[,-c(7)])

test_prediction_01 <- knn(train = train.df[,-c(7)],test = test.df[,-c(7)], cl
= train.df[,7], k=3, prob=TRUE)
valid_prediction_01 <- knn(train = train.df[,-c(7)],test = valid.df[,-c(7)],
cl = train.df[,7], k=3, prob=TRUE)
train_prediction_01 <- knn(train = train.df[,-c(7)],test = train.df[,-c(7)],
cl = train.df[,7], k=3, prob=TRUE)

confusionMatrix(test_prediction_01, as.factor(test.df[,7]))
```

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

```
##           Reference
## Prediction   0   1
##          0 889  35
##          1   3  73
##
##                Accuracy : 0.962
##                  95% CI : (0.9482, 0.973)
##     No Information Rate : 0.892
##     P-Value [Acc > NIR] : 4.592e-16
##
##                   Kappa : 0.7732
##
##  Mcnemar's Test P-Value : 4.934e-07
##
##             Sensitivity : 0.9966
##             Specificity : 0.6759
##          Pos Pred Value : 0.9621
##          Neg Pred Value : 0.9605
##              Prevalence : 0.8920
##          Detection Rate : 0.8890
##    Detection Prevalence : 0.9240
##       Balanced Accuracy : 0.8363
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(valid_prediction_01, as.factor(valid.df[,7]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1353   42
##          1    7   98
##
##                Accuracy : 0.9673
##                  95% CI : (0.957, 0.9757)
##     No Information Rate : 0.9067
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7826
##
##  Mcnemar's Test P-Value : 1.191e-06
##
##             Sensitivity : 0.9949
##             Specificity : 0.7000
##          Pos Pred Value : 0.9699
##          Neg Pred Value : 0.9333
##              Prevalence : 0.9067
##          Detection Rate : 0.9020
```

```
##    Detection Prevalence : 0.9300
##       Balanced Accuracy : 0.8474
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(train_prediction_01, as.factor(train.df[,7]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2263   54
##          1    5  178
##
##                Accuracy : 0.9764
##                  95% CI : (0.9697, 0.982)
##     No Information Rate : 0.9072
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8452
##
##  Mcnemar's Test P-Value : 4.129e-10
##
##             Sensitivity : 0.9978
##             Specificity : 0.7672
##          Pos Pred Value : 0.9767
##          Neg Pred Value : 0.9727
##              Prevalence : 0.9072
##          Detection Rate : 0.9052
##    Detection Prevalence : 0.9268
##       Balanced Accuracy : 0.8825
##
##        'Positive' Class : 0
##
```