Universal Bank Data Analysis

Chathurani Ekanayake

2023-02-19

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

library(readxl)  
data <- read\_excel("F:/1st sem/ML/Assignment 02/originaldata.xlsx")  
View(data)  
  
dataframe <- data[, -c(1,5)]  
View(dataframe)  
dummy\_Education <- as.data.frame(dummy.code(dataframe$Education))  
names(dummy\_Education) <- c("Education\_01", "Education\_02","Education\_03")  
dataframe\_without\_education <- dataframe[-c(6)]  
UniBank\_data <- cbind(dataframe\_without\_education, dummy\_Education)  
View(UniBank\_data)  
names(UniBank\_data)[8] ="Securities.Account"  
names(UniBank\_data)[9] ="CD.Account"  
names(UniBank\_data)[7] ="Personal.Loan"  
View(UniBank\_data)  
  
  
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, ]  
  
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)  
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

accuracy.df <- data.frame(k = seq(1, 60, 1), accuracy = rep(0, 60))  
for(i in 1:60) {  
 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  
## 31 31 0.9385  
## 32 32 0.9365  
## 33 33 0.9390  
## 34 34 0.9360  
## 35 35 0.9370  
## 36 36 0.9335  
## 37 37 0.9355  
## 38 38 0.9340  
## 39 39 0.9355  
## 40 40 0.9345  
## 41 41 0.9360  
## 42 42 0.9335  
## 43 43 0.9345  
## 44 44 0.9325  
## 45 45 0.9350  
## 46 46 0.9325  
## 47 47 0.9325  
## 48 48 0.9310  
## 49 49 0.9325  
## 50 50 0.9300  
## 51 51 0.9320  
## 52 52 0.9285  
## 53 53 0.9310  
## 54 54 0.9280  
## 55 55 0.9285  
## 56 56 0.9260  
## 57 57 0.9270  
## 58 58 0.9255  
## 59 59 0.9260  
## 60 60 0.9245

View(accuracy.df)  
  
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   
##

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)  
prediction\_01 <- knn(train = train.df[,-7],test = customer.df, cl = train.df[,7], k=3, prob=TRUE)  
prediction\_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

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   
##