Machine Learning Assignment - Sai Supriya Vengala

2022-10-03

install.packages("psych", repos = "http://cran.us.r-project.org")

##   
## The downloaded binary packages are in  
## /var/folders/s0/bmcnbw5s19v51v5f0f1lcj\_m0000gn/T//Rtmpu65xJI/downloaded\_packages

install.packages("caret", repos = "http://cran.us.r-project.org")

##   
## The downloaded binary packages are in  
## /var/folders/s0/bmcnbw5s19v51v5f0f1lcj\_m0000gn/T//Rtmpu65xJI/downloaded\_packages

install.packages("FNN", repos = "http://cran.us.r-project.org")

##   
## The downloaded binary packages are in  
## /var/folders/s0/bmcnbw5s19v51v5f0f1lcj\_m0000gn/T//Rtmpu65xJI/downloaded\_packages

install.packages("class", repos = "http://cran.us.r-project.org")

##   
## The downloaded binary packages are in  
## /var/folders/s0/bmcnbw5s19v51v5f0f1lcj\_m0000gn/T//Rtmpu65xJI/downloaded\_packages

install.packages("dplyr", repos = "http://cran.us.r-project.org")

##   
## The downloaded binary packages are in  
## /var/folders/s0/bmcnbw5s19v51v5f0f1lcj\_m0000gn/T//Rtmpu65xJI/downloaded\_packages

install.packages("rlang",repos = "http://cran.us.r-project.org")

##   
## The downloaded binary packages are in  
## /var/folders/s0/bmcnbw5s19v51v5f0f1lcj\_m0000gn/T//Rtmpu65xJI/downloaded\_packages

###Project Background:

*Liability customers - Majority - Depositors* Asset customers - Small - Borrowers *Campaign of last year - conversion rate of 9.6% [Among the 5000 customers, only 480 (= 9.6%) accepted the personal loan that was offered to them in the earlier campaign.]* Goal : use k-NN to predict whether a new customer will accept a loan offer. \* Data (rows): 5000 customers \*Success class as 1 (loan acceptance)

####Packages used:

library(psych) #for creating dummies

## Warning: package 'psych' was built under R version 4.1.2

library(caret) #for data partition, normalize data

## Warning: package 'caret' was built under R version 4.1.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.1.2

##   
## Attaching package: 'ggplot2'

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

## Loading required package: lattice

library(FNN) #for Perfoming knn classification

## Warning: package 'FNN' was built under R version 4.1.2

library(class)

## Warning: package 'class' was built under R version 4.1.2

##   
## Attaching package: 'class'

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

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.1.2

##   
## 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(rlang)

## Warning: package 'rlang' was built under R version 4.1.2

###importing data

`UniversalBank.(1)` <- read.csv("~/Downloads/UniversalBank (1).csv", stringsAsFactors=TRUE)

#Eliminating variables [id & zip code] from the dataset  
df=subset(`UniversalBank.(1)`, select=-c(ID, ZIP.Code ))

#creating dummies  
dummy\_Education <- as.data.frame(dummy.code(df$Education))  
names(dummy\_Education) <- c("Education\_1", "Education\_2","Education\_3") #renaming dummy variable  
df\_without\_education <- subset(df, select=-c(Education)) #eliminating education variable  
  
`UniversalBank.(1)` <- cbind(df\_without\_education, dummy\_Education) #main dataset

###Data partition

#Partitioning the data into Traning(60%) and Validation(40%)  
set.seed(1234)  
Train\_Index = createDataPartition(`UniversalBank.(1)`$Age, p= 0.6 , list=FALSE)  
Train\_Data =`UniversalBank.(1)`[Train\_Index,] #3001 observations  
  
Validation\_Data = `UniversalBank.(1)`[-Train\_Index,] #1999 observations

###Generating test data

Test\_Data <- data.frame(Age=40 , Experience=10, Income = 84, Family = 2, CCAvg = 2, Education\_1 = 0, Education\_2 = 1, Education\_3 = 0, Mortgage = 0, Securities.Account = 0, CD.Account = 0, Online = 1, CreditCard = 1, stringsAsFactors = FALSE)

###Data Normalization

train.norm.df <- Train\_Data  
valid.norm.df <- Validation\_Data  
test.norm.df <- Test\_Data  
maindata.norm.df <- `UniversalBank.(1)`  
  
head(maindata.norm.df)

## Age Experience Income Family CCAvg Mortgage Personal.Loan Securities.Account  
## 1 25 1 49 4 1.6 0 0 1  
## 2 45 19 34 3 1.5 0 0 1  
## 3 39 15 11 1 1.0 0 0 0  
## 4 35 9 100 1 2.7 0 0 0  
## 5 35 8 45 4 1.0 0 0 0  
## 6 37 13 29 4 0.4 155 0 0  
## CD.Account Online CreditCard Education\_1 Education\_2 Education\_3  
## 1 0 0 0 1 0 0  
## 2 0 0 0 1 0 0  
## 3 0 0 0 1 0 0  
## 4 0 0 0 0 0 1  
## 5 0 0 1 0 0 1  
## 6 0 1 0 0 0 1

# use preProcess() from the caret package to normalize .  
norm.values <- preProcess(Train\_Data[,-7], method=c("center", "scale"))  
  
train.norm.df[,-7] <- predict(norm.values, Train\_Data[,-7]) #Training Data  
valid.norm.df [,-7]<- predict(norm.values, Validation\_Data[,-7])#Validation Data  
test.norm.df <- predict(norm.values, Test\_Data)#Test Data  
maindata.norm.df[,-7] <- predict(norm.values,`UniversalBank.(1)`[,-7]) #Training + Validation data  
  
head(maindata.norm.df)

## Age Experience Income Family CCAvg Mortgage  
## 1 -1.77136698 -1.6613124 -0.5177762 1.3933091 -0.1845814 -0.5438042  
## 2 -0.03145296 -0.0978843 -0.8425723 0.5187388 -0.2419870 -0.5438042  
## 3 -0.55342717 -0.4453128 -1.3405930 -1.2304018 -0.5290146 -0.5438042  
## 4 -0.90140997 -0.9664555 0.5865306 -1.2304018 0.4468794 -0.5438042  
## 5 -0.90140997 -1.0533126 -0.6043885 1.3933091 -0.5290146 -0.5438042  
## 6 -0.72741857 -0.6190270 -0.9508377 1.3933091 -0.8734478 1.0035659  
## Personal.Loan Securities.Account CD.Account Online CreditCard Education\_1  
## 1 0 2.9564494 -0.2533042 -1.2038741 -0.6538696 1.1696714  
## 2 0 2.9564494 -0.2533042 -1.2038741 -0.6538696 1.1696714  
## 3 0 -0.3381309 -0.2533042 -1.2038741 -0.6538696 1.1696714  
## 4 0 -0.3381309 -0.2533042 -1.2038741 -0.6538696 -0.8546561  
## 5 0 -0.3381309 -0.2533042 -1.2038741 1.5288474 -0.8546561  
## 6 0 -0.3381309 -0.2533042 0.8303749 -0.6538696 -0.8546561  
## Education\_2 Education\_3  
## 1 -0.6414311 -0.6331615  
## 2 -0.6414311 -0.6331615  
## 3 -0.6414311 -0.6331615  
## 4 -0.6414311 1.5788497  
## 5 -0.6414311 1.5788497  
## 6 -0.6414311 1.5788497

###Perfoming k-NN classification , using k = 1

set.seed(2000)  
prediction <- knn(train = train.norm.df[,-7], test = valid.norm.df[,-7],   
 cl = train.norm.df[,7], k = 1, prob=TRUE)   
actual= valid.norm.df$Personal.Loan  
prediction\_prob = attr(prediction,"prob")  
table(prediction,actual)

## actual  
## prediction 0 1  
## 0 1770 68  
## 1 25 136

mean(prediction==actual)

## [1] 0.9534767

NROW(train.norm.df)

## [1] 3001

sqrt(3001)

## [1] 54.78138

accuracy.df <- data.frame(k = seq(1, 60, 1), accuracy = rep(0, 60))  
  
# compute knn for different k on validation.  
for(i in 1:60) {  
prediction <- knn(train = train.norm.df[,-7], test = valid.norm.df[-7],   
 cl = train.norm.df[,7], k = i, prob=TRUE)   
  
accuracy.df[i,2] <- mean(prediction==actual)  
}  
accuracy.df

## k accuracy  
## 1 1 0.9534767  
## 2 2 0.9484742  
## 3 3 0.9544772  
## 4 4 0.9529765  
## 5 5 0.9524762  
## 6 6 0.9479740  
## 7 7 0.9489745  
## 8 8 0.9459730  
## 9 9 0.9454727  
## 10 10 0.9469735  
## 11 11 0.9439720  
## 12 12 0.9434717  
## 13 13 0.9424712  
## 14 14 0.9409705  
## 15 15 0.9409705  
## 16 16 0.9404702  
## 17 17 0.9399700  
## 18 18 0.9399700  
## 19 19 0.9404702  
## 20 20 0.9379690  
## 21 21 0.9379690  
## 22 22 0.9354677  
## 23 23 0.9364682  
## 24 24 0.9344672  
## 25 25 0.9349675  
## 26 26 0.9329665  
## 27 27 0.9354677  
## 28 28 0.9344672  
## 29 29 0.9339670  
## 30 30 0.9314657  
## 31 31 0.9314657  
## 32 32 0.9314657  
## 33 33 0.9319660  
## 34 34 0.9304652  
## 35 35 0.9304652  
## 36 36 0.9299650  
## 37 37 0.9284642  
## 38 38 0.9284642  
## 39 39 0.9284642  
## 40 40 0.9274637  
## 41 41 0.9269635  
## 42 42 0.9254627  
## 43 43 0.9254627  
## 44 44 0.9249625  
## 45 45 0.9249625  
## 46 46 0.9239620  
## 47 47 0.9244622  
## 48 48 0.9249625  
## 49 49 0.9244622  
## 50 50 0.9224612  
## 51 51 0.9244622  
## 52 52 0.9234617  
## 53 53 0.9234617  
## 54 54 0.9239620  
## 55 55 0.9224612  
## 56 56 0.9219610  
## 57 57 0.9229615  
## 58 58 0.9224612  
## 59 59 0.9214607  
## 60 60 0.9214607

The value of k we choose is 1 as it is given in the question [i.e the choice of k that balances between overfitting and ignoring the predictor information]

####Validation data results using best k value [i.e: k = 1]

set.seed(1234)  
prediction <- knn(train = train.norm.df[,-7], test = valid.norm.df[,-7],   
 cl = train.norm.df[,7], k = 1, prob=TRUE)   
actual= valid.norm.df$Personal.Loan  
prediction\_prob = attr(prediction,"prob")  
  
  
  
#Answer 3: confusion matrix for the best k value =1  
table(prediction,actual)

## actual  
## prediction 0 1  
## 0 1770 68  
## 1 25 136

#accuracy of the best k=1  
mean(prediction==actual)

## [1] 0.9534767

#### Classifying the customer using the best k [perfominng k-NN classification on test data]

prediction\_test <- knn(train = maindata.norm.df[,-7], test = Test\_Data,   
 cl = maindata.norm.df[,7], k = 1, prob=TRUE)   
head(prediction\_test)

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

k-NN model predicted that the new customer will accept a loan offer [loan accepted]

### 5)Repartition the data, this time into training, validation, and test sets (50% : 30% : 20%). Apply the k-NN method with the k chosen above. Compare the confusion matrix of the test set with that of the training and validation sets.

#Partitioning the data into Traning(50%) ,Validation(30%), Test(20%)  
set.seed(1234)  
  
Test\_Index\_1 = createDataPartition(`UniversalBank.(1)`$Age, p= 0.2 , list=FALSE) #20% test data   
Test\_Data\_1 = `UniversalBank.(1)` [Test\_Index\_1,]  
  
Rem\_DATA = `UniversalBank.(1)`[-Test\_Index\_1,] #80% remaining data [training + validation]  
  
Train\_Index\_1 = createDataPartition(Rem\_DATA$Age, p= 0.5 , list=FALSE)  
Train\_Data\_1 = Rem\_DATA[Train\_Index\_1,] #Training data  
  
Validation\_Data\_1 = Rem\_DATA[-Train\_Index\_1,] #Validation data

#Data Normalization  
  
  
# Copy the original data  
train.norm.df\_1 <- Train\_Data\_1  
valid.norm.df\_1 <- Validation\_Data\_1  
test.norm.df\_1 <- Test\_Data\_1  
rem\_data.norm.df\_1 <- Rem\_DATA  
  
# use preProcess() from the caret package to normalize Sales and Age.  
norm.values\_1 <- preProcess(Train\_Data\_1[-7], method=c("center", "scale"))  
  
train.norm.df\_1[-7] <- predict(norm.values\_1, Train\_Data\_1[-7]) #Training Data  
valid.norm.df\_1[-7] <- predict(norm.values\_1, Validation\_Data\_1[-7])#Validation Data  
test.norm.df\_1[-7] <- predict(norm.values\_1, test.norm.df\_1[-7]) #Test Data  
test.norm.df\_1[-7] <- predict(norm.values\_1, Test\_Data\_1[-7])  
rem\_data.norm.df\_1[-7] <- predict(norm.values\_1,Rem\_DATA[-7]) #Training + Validation data  
  
head(test.norm.df\_1)

## Age Experience Income Family CCAvg Mortgage  
## 9 -0.90840439 -0.883582836 0.1435652 0.5333142 -0.780693325 0.4495336  
## 28 0.05751618 -0.008054857 1.8189997 -1.2081200 0.234699617 -0.5532869  
## 32 -0.46934959 -0.358266049 -0.9878972 -1.2081200 0.009056741 -0.5532869  
## 40 -0.64497151 -0.620924443 0.1218063 1.4040313 -0.724282606 2.1948269  
## 42 -0.99621536 -0.971135634 -0.3133715 0.5333142 0.178288898 -0.5532869  
## 63 -0.29372767 -0.183160453 -1.1402094 -1.2081200 -0.555050449 -0.5532869  
## Personal.Loan Securities.Account CD.Account Online CreditCard  
## 9 0 -0.3360202 -0.2646808 0.8429167 -0.6350646  
## 28 0 -0.3360202 -0.2646808 0.8429167 1.5738557  
## 32 0 -0.3360202 -0.2646808 0.8429167 -0.6350646  
## 40 0 -0.3360202 -0.2646808 0.8429167 -0.6350646  
## 42 0 -0.3360202 -0.2646808 -1.1857637 -0.6350646  
## 63 0 -0.3360202 -0.2646808 -1.1857637 -0.6350646  
## Education\_1 Education\_2 Education\_3  
## 9 -0.827392 -0.6607293 1.566207  
## 28 1.208013 -0.6607293 -0.638166  
## 32 -0.827392 -0.6607293 1.566207  
## 40 -0.827392 1.5127224 -0.638166  
## 42 1.208013 -0.6607293 -0.638166  
## 63 1.208013 -0.6607293 -0.638166

#Perfoming k-NN classification on Training Data, k = 1  
set.seed(1234)  
prediction\_Q5 <- knn(train = train.norm.df\_1[,-7], test = valid.norm.df\_1[,-7],   
 cl = train.norm.df\_1[,7], k = 1, prob=TRUE)   
actual= valid.norm.df\_1$Personal.Loan  
prediction\_prob = attr(prediction\_Q5,"prob")  
  
table(prediction\_Q5,actual) #confusion matrix for the best k value =1

## actual  
## prediction\_Q5 0 1  
## 0 1795 69  
## 1 16 119

mean(prediction\_Q5==actual) #accuracy of the best k=1

## [1] 0.9574787

set.seed(1234)  
prediction\_Q5 <- knn(train = rem\_data.norm.df\_1[,-7], test = test.norm.df\_1[,-7],   
 cl = rem\_data.norm.df\_1[,7], k = 1, prob=TRUE)   
actual= test.norm.df\_1$Personal.Loan  
prediction\_prob = attr(prediction\_Q5,"prob")  
  
table(prediction\_Q5,actual) #confusion matrix for the best k value =1

## actual  
## prediction\_Q5 0 1  
## 0 907 25  
## 1 12 57

mean(prediction\_Q5==actual) #accuracy of the best k=1

## [1] 0.963037

The model performed better in the test set, as it got enough data to learn from i.e 80% of the data, Whereas when we were working on training data it only learned from 50% of the data.