Assignment 2 Knn Model - Universal Bank Data

Jyothsna P - 811251679

2023-02-19

Add all necessary libraries needed to run the code

```
#install.packages("readr")
library(readr)
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(fastDummies)
library(ISLR)
library(class)
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(gmodels)
```

Data Load and Manipulation

Read universal Bank Data to R environment

```
universalBankData <- read.csv("C:/Users/peddi/OneDrive/Desktop/Spring 2023/FML/Module 4/Assignment 2/Uni
```

To verify the total rows in the data set

```
nrow(universalBankData)
```

```
## [1] 5000
```

Verify if there are any null values in the datasets

```
any(is.na(universalBankData))
```

```
## [1] FALSE
```

To check descriptive statistics of all the features in the universal Bank data

```
summary(universalBankData)
```

```
##        ID            Age          Experience       Income          ZIP.Code    
##  Min.   :   1   Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   : 9307  
##  1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st Qu.:91911  
##  Median :2500   Median :45.00   Median :20.0   Median : 64.00   Median :93437  
##  Mean   :2500   Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean   :93153  
##  3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd Qu.:94608  
##  Max.   :5000   Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.   :96651  
##      Family          CCAvg          Education        Mortgage    
##  Min.   :1.000   Min.   : 0.000   Min.   :1.000   Min.   :  0.0  
##  1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0  
##  Median :2.000   Median : 1.500   Median :2.000   Median :  0.0  
##  Mean   :2.396   Mean   : 1.938   Mean   :1.881   Mean   : 56.5  
##  3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0  
##  Max.   :4.000   Max.   :10.000   Max.   :3.000   Max.   :635.0  
##  Personal.Loan   Securities.Account   CD.Account         Online      
##  Min.   :0.000   Min.   :0.0000     Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.000   1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :0.000   Median :0.0000     Median :0.0000   Median :1.0000  
##  Mean   :0.096   Mean   :0.1044     Mean   :0.0604   Mean   :0.5968  
##  3rd Qu.:0.000   3rd Qu.:0.0000     3rd Qu.:0.0000   3rd Qu.:1.0000  
##  Max.   :1.000   Max.   :1.0000     Max.   :1.0000   Max.   :1.0000  
##    CreditCard   
##  Min.   :0.000  
##  1st Qu.:0.000  
##  Median :0.000  
##  Mean   :0.294  
##  3rd Qu.:1.000  
##  Max.   :1.000
```

Remove ID and Zip.Code columns from the dataframe

```r
#dplyr package is helpful for data manipulation
#select function in dplyr helps in selecting fewer columns in the dataframe or excluding columns in the

#remove ID and Zipcode columns from the Universal Bank Data
universalBankData <- select(universalBankData,-c(ID,ZIP.Code))

#Existing features in the dataset clarifies that ID and Zip code are removed from the dataset
colnames(universalBankData)
```

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

Identify data type of all the features in the dataset

```r
sapply(universalBankData,class)
```

```
##                Age         Experience             Income             Family
##          "integer"          "integer"          "integer"          "integer"
##              CCAvg          Education           Mortgage      Personal.Loan
##          "numeric"          "integer"          "integer"          "integer"
## Securities.Account         CD.Account             Online         CreditCard
##          "integer"          "integer"          "integer"          "integer"
```

Convert the Personal Loan variable to factor

```r
#universalBankData$Personal.Loan <- factor(universalBankData$Personal.Loan)
#summary(universalBankData$Personal.Loan)
```

Create dummy variables for education using fastdummies package

```r
#install.packages("fastDummies")
library(fastDummies)

#dummy_cols function in fastDummies package helps in creating dummy variables automatically using the b
universalBankData <- dummy_cols(universalBankData,select_columns = "Education")
colnames(universalBankData)
```

```
##  [1] "Age"                "Experience"         "Income"
##  [4] "Family"             "CCAvg"              "Education"
##  [7] "Mortgage"           "Personal.Loan"      "Securities.Account"
## [10] "CD.Account"         "Online"             "CreditCard"
## [13] "Education_1"        "Education_2"        "Education_3"
```

Remove Education variable after Dummy variables are created for Education

```
universalBankData <- select(universalBankData,-"Education")
#Column names confirm that the education column is removed from the dataset
colnames(universalBankData)
```

```
##  [1] "Age"                "Experience"         "Income"
##  [4] "Family"             "CCAvg"              "Mortgage"
##  [7] "Personal.Loan"      "Securities.Account" "CD.Account"
## [10] "Online"             "CreditCard"         "Education_1"
## [13] "Education_2"        "Education_3"
```

```
summary(universalBankData)
```

```
##       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           Mortgage      Personal.Loan   Securities.Account
##  Min.   : 0.000   Min.   :  0.0   Min.   :0.000   Min.   :0.0000
##  1st Qu.: 0.700   1st Qu.:  0.0   1st Qu.:0.000   1st Qu.:0.0000
##  Median : 1.500   Median :  0.0   Median :0.000   Median :0.0000
##  Mean   : 1.938   Mean   : 56.5   Mean   :0.096   Mean   :0.1044
##  3rd Qu.: 2.500   3rd Qu.:101.0   3rd Qu.:0.000   3rd Qu.:0.0000
##  Max.   :10.000   Max.   :635.0   Max.   :1.000   Max.   :1.0000
##    CD.Account         Online         CreditCard     Education_1
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000
##  Median :0.0000   Median :1.0000   Median :0.000   Median :0.0000
##  Mean   :0.0604   Mean   :0.5968   Mean   :0.294   Mean   :0.4192
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000   Max.   :1.0000
##   Education_2      Education_3
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000
##  Mean   :0.2806   Mean   :0.3002
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
```

Question 1

Use 60% of data for training and 40% of data for validation

```
Index_train <- createDataPartition(universalBankData$Personal.Loan,
                                   p=0.6,list=FALSE)
```

```r
train <- universalBankData[Index_train,]
val <- universalBankData[-Index_train,]
```

Test Input provided in the question

```r
input <- 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_1 = 0, Education_2 = 1, Education_3 = 0)
```

Numeric variables in the dataset as a vector

```r
numericVariables <- c("Age","Experience","Income","Family","CCAvg","Mortgage")
```

Normalize the datasets for model building

```r
normValues <- preProcess(train[,numericVariables],
                         method=c("center","scale"))

train_norm <- predict(normValues,train)
#validation normalized variable
val_norm <-  predict(normValues,val)

input_norm <- predict(normValues,input)
```

Summary of normalized train, input, and validation datasets

```r
summary(train_norm)
```

```
##       Age              Experience           Income            Family
##  Min.   :-1.94707   Min.   :-2.01501   Min.   :-1.4124   Min.   :-1.2033
##  1st Qu.:-0.90169   1st Qu.:-0.88353   1st Qu.:-0.7615   1st Qu.:-1.2033
##  Median : 0.05657   Median :-0.01317   Median :-0.2191   Median :-0.3468
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.84060   3rd Qu.: 0.85719   3rd Qu.: 0.4752   3rd Qu.: 1.3660
##  Max.   : 1.88597   Max.   : 1.90163   Max.   : 3.1440   Max.   : 1.3660
##      CCAvg             Mortgage         Personal.Loan     Securities.Account
##  Min.   :-1.1201   Min.   :-0.5558   Min.   :0.00000   Min.   :0.000
##  1st Qu.:-0.7107   1st Qu.:-0.5558   1st Qu.:0.00000   1st Qu.:0.000
##  Median :-0.2428   Median :-0.5558   Median :0.00000   Median :0.000
##  Mean   : 0.0000   Mean   : 0.0000   Mean   :0.09867   Mean   :0.106
##  3rd Qu.: 0.3420   3rd Qu.: 0.4451   3rd Qu.:0.00000   3rd Qu.:0.000
##  Max.   : 4.7283   Max.   : 5.7369   Max.   :1.00000   Max.   :1.000
##    CD.Account           Online          CreditCard      Education_1
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
```

```
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000
##  Median :0.00000   Median :1.0000   Median :0.000   Median :0.0000
##  Mean   :0.06033   Mean   :0.5997   Mean   :0.298   Mean   :0.4173
##  3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.000   Max.   :1.0000
##   Education_2      Education_3
##  Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.000
##  Median :0.0000   Median :0.000
##  Mean   :0.2787   Mean   :0.304
##  3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :1.0000   Max.   :1.000
```

```
summary(val_norm)
```

```
##       Age              Experience          Income            Family
##  Min.   :-1.947071   Min.   :-2.01501   Min.   :-1.41240   Min.   :-1.20326
##  1st Qu.:-0.901695   1st Qu.:-0.88353   1st Qu.:-0.73979   1st Qu.:-1.20326
##  Median :-0.030548   Median :-0.01317   Median :-0.17567   Median :-0.34685
##  Mean   :-0.002671   Mean   :-0.01017   Mean   : 0.03677   Mean   :-0.01841
##  3rd Qu.: 0.840598   3rd Qu.: 0.85719   3rd Qu.: 0.60543   3rd Qu.: 0.50957
##  Max.   : 1.885974   Max.   : 1.98866   Max.   : 3.27418   Max.   : 1.36598
##      CCAvg           Mortgage         Personal.Loan   Securities.Account
##  Min.   :-1.1201   Min.   :-0.55583   Min.   :0.000   Min.   :0.000
##  1st Qu.:-0.7107   1st Qu.:-0.55583   1st Qu.:0.000   1st Qu.:0.000
##  Median :-0.2428   Median :-0.55583   Median :0.000   Median :0.000
##  Mean   : 0.0332   Mean   : 0.01016   Mean   :0.092   Mean   :0.102
##  3rd Qu.: 0.4005   3rd Qu.: 0.44506   3rd Qu.:0.000   3rd Qu.:0.000
##  Max.   : 4.7283   Max.   : 5.29096   Max.   :1.000   Max.   :1.000
##    CD.Account         Online         CreditCard     Education_1
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.000
##  Median :0.0000   Median :1.0000   Median :0.000   Median :0.000
##  Mean   :0.0605   Mean   :0.5925   Mean   :0.288   Mean   :0.422
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000   Max.   :1.000
##   Education_2      Education_3
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000
##  Mean   :0.2835   Mean   :0.2945
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
```

```
summary(input_norm)
```

```
##       Age            Experience         Income           Family
##  Min.   :-0.4661   Min.   :-0.8835   Min.   :0.2366   Min.   :-0.3468
##  1st Qu.:-0.4661   1st Qu.:-0.8835   1st Qu.:0.2366   1st Qu.:-0.3468
##  Median :-0.4661   Median :-0.8835   Median :0.2366   Median :-0.3468
##  Mean   :-0.4661   Mean   :-0.8835   Mean   :0.2366   Mean   :-0.3468
##  3rd Qu.:-0.4661   3rd Qu.:-0.8835   3rd Qu.:0.2366   3rd Qu.:-0.3468
##  Max.   :-0.4661   Max.   :-0.8835   Max.   :0.2366   Max.   :-0.3468
```

```
##      CCAvg            Mortgage       Securities.Account   CD.Account
##  Min.   :0.04958   Min.   :-0.5558   Min.   :0          Min.   :0
##  1st Qu.:0.04958   1st Qu.:-0.5558   1st Qu.:0          1st Qu.:0
##  Median :0.04958   Median :-0.5558   Median :0          Median :0
##  Mean   :0.04958   Mean   :-0.5558   Mean   :0          Mean   :0
##  3rd Qu.:0.04958   3rd Qu.:-0.5558   3rd Qu.:0          3rd Qu.:0
##  Max.   :0.04958   Max.   :-0.5558   Max.   :0          Max.   :0
##      Online     CreditCard   Education_1  Education_2  Education_3
##  Min.   :1   Min.   :1    Min.   :0    Min.   :1    Min.   :0
##  1st Qu.:1   1st Qu.:1    1st Qu.:0    1st Qu.:1    1st Qu.:0
##  Median :1   Median :1    Median :0    Median :1    Median :0
##  Mean   :1   Mean   :1    Mean   :0    Mean   :1    Mean   :0
##  3rd Qu.:1   3rd Qu.:1    3rd Qu.:0    3rd Qu.:1    3rd Qu.:0
##  Max.   :1   Max.   :1    Max.   :0    Max.   :1    Max.   :0
```

Use Knn function to predit outcome of input given in the question

```
train_predictors <- select(train_norm,-Personal.Loan)
train_label <- select(train_norm,Personal.Loan)

val_predictors <- select(val_norm,-Personal.Loan)
val_label <- select(val_norm,Personal.Loan)


input_norm_pred <- knn(train=train_predictors, test=input_norm,cl=train_label$Personal.Loan,k=1)
```

Class 0 - Loan Not Accepted, Class 1 -Loan Accepted

```
#As mentioned in the question.
cutoff <- 0.5
successOutcome <- ifelse(as.numeric(input_norm_pred) <0.5,
                         "Loan Not Accepted","Loan Accepted")
print(successOutcome)
```

```
## [1] "Loan Accepted"
```

Question 2

Tuning Model to find the best K value for knn model

```
#knn
set.seed(428)
search_grid <- expand.grid(k=c(1:20))

train_norm$Personal.Loan <- factor(train_norm$Personal.Loan)

model <- train(Personal.Loan~., data=train_norm,method="knn",tuneGrid=search_grid,metric="Accuracy")
```

```r
model
```

```
## k-Nearest Neighbors
##
## 3000 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3000, 3000, 3000, 3000, 3000, 3000, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9581875  0.7386056
##    2  0.9541162  0.7083446
##    3  0.9532061  0.6985916
##    4  0.9531494  0.6934360
##    5  0.9537246  0.6916418
##    6  0.9524832  0.6790538
##    7  0.9524206  0.6742759
##    8  0.9518803  0.6678036
##    9  0.9515211  0.6627067
##   10  0.9501152  0.6478388
##   11  0.9489907  0.6364732
##   12  0.9484822  0.6316471
##   13  0.9479742  0.6255914
##   14  0.9468504  0.6147834
##   15  0.9459356  0.6065240
##   16  0.9452103  0.5985509
##   17  0.9439004  0.5865921
##   18  0.9440106  0.5859618
##   19  0.9426255  0.5737960
##   20  0.9426650  0.5734876
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```r
#Below code prints the best K value from the knn model tuning
cat("Optimal K value for the dataset using the train method is ",as.character(model$bestTune[,"k"]))
```

```
## Optimal K value for the dataset using the train method is  1
```

Alternative way to find the best k value using the train and validation dataset

```r
val_label$Personal.Loan <- factor(val_label$Personal.Loan)
accuracydf <- data.frame(kValue=seq(1,14,1),Accuracy=0)

for(i in 1:nrow(accuracydf)){
```

```r
val_label_predict <-  knn(train=train_predictors,test=val_predictors,
                          cl=train_label$Personal.Loan,k=i)

accuracydf[i,2] <- confusionMatrix(val_label_predict,
                                   val_label$Personal.Loan)$overall[1]
}
accuracydf
```

```
##    kValue Accuracy
## 1       1   0.9595
## 2       2   0.9605
## 3       3   0.9645
## 4       4   0.9620
## 5       5   0.9615
## 6       6   0.9600
## 7       7   0.9570
## 8       8   0.9555
## 9       9   0.9555
## 10     10   0.9550
## 11     11   0.9535
## 12     12   0.9540
## 13     13   0.9520
## 14     14   0.9495
```

```r
bestk_alternativeOption <- accuracydf[which.max(accuracydf$Accuracy),][1]

cat("Alternative Approach - Optimal K value for the dataset is ",
    as.character(bestk_alternativeOption))
```

```
## Alternative Approach - Optimal K value for the dataset is  3
```

Question 3

```r
Optimal_k_value =bestk_alternativeOption

#Use train_norm and val_norm created in previous steps for inputs of knn function

predicted_Label <- knn(train=train_predictors,test=val_predictors,
                       cl=train_label$Personal.Loan,k=Optimal_k_value)

#Get the top 10 rows from the predicted label
head(predicted_Label,n = 10)
```

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

Build confusion matrix for the Predicted vs actual outcome

```r
confusionMatrix <- CrossTable(val_label$Personal.Loan,
                              predicted_Label,prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  2000
##
##
##                         | predicted_Label
## val_label$Personal.Loan |         0 |         1 | Row Total |
## ------------------------|-----------|-----------|-----------|
##                       0 |      1812 |         4 |      1816 |
##                         |     0.998 |     0.002 |     0.908 |
##                         |     0.964 |     0.033 |           |
##                         |     0.906 |     0.002 |           |
## ------------------------|-----------|-----------|-----------|
##                       1 |        67 |       117 |       184 |
##                         |     0.364 |     0.636 |     0.092 |
##                         |     0.036 |     0.967 |           |
##                         |     0.034 |     0.058 |           |
## ------------------------|-----------|-----------|-----------|
##            Column Total |      1879 |       121 |      2000 |
##                         |     0.940 |     0.060 |           |
## ------------------------|-----------|-----------|-----------|
##
##
```

```r
confusionMatrix
```

```
## $t
##    y
## x      0    1
##   0 1812    4
##   1   67  117
##
## $prop.row
##    y
## x             0           1
##   0 0.997797357 0.002202643
##   1 0.364130435 0.635869565
##
## $prop.col
##    y
```

```
## x            0          1
##   0 0.96434274 0.03305785
##   1 0.03565726 0.96694215
##
## $prop.tbl
##    y
## x        0      1
##   0 0.9060 0.0020
##   1 0.0335 0.0585
```

Question 4

```r
input_norm_pred_WithOptimal_k <- knn(train=train_predictors,test=input_norm,
                                     cl=train_label$Personal.Loan,
                                     k=Optimal_k_value)

input_norm_pred_WithOptimal_k
```

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

Quesiton 5

For partition of data into three sets using the partition function available in the splitTools package

```r
#install.packages("splitTools")
#install.packages("ranger")
library(splitTools)
library(ranger)


partitionIndex <- partition(universalBankData$Age,
                            type=c("stratified"),
                            p = c(train=0.5,val=0.3,test=0.2))

# Summary of partition Index
summary(partitionIndex)
```

```
##       Length Class  Mode
## train 2496   -none- numeric
## val   1502   -none- numeric
## test  1002   -none- numeric
```

```r
#structure of partition Index
str(partitionIndex)
```

```
## List of 3
##  $ train: int [1:2496] 2 6 7 8 9 11 13 15 16 17 ...
##  $ val  : int [1:1502] 1 3 32 33 34 39 40 42 43 45 ...
##  $ test : int [1:1002] 4 5 10 12 14 21 24 26 29 53 ...
```

Create three data frames for train, val and test using the partition index created in the previous step

```r
train_new <- universalBankData[partitionIndex$train,]
val_new <- universalBankData[partitionIndex$val,]
test_new <- universalBankData[partitionIndex$test,]
```

Normalize the three datasets using the preProcess method

```r
normValues_new <- preProcess(train_new[,numericVariables],
                             method=c("center","scale")) # method="range")

train_new_norm <- predict(normValues_new,train_new)
val_new_norm <-  predict(normValues_new,val_new)
test_new_norm <-  predict(normValues_new,test_new)
```

Create separate datasets for predictors and labels for normalized train, validate and test dataset

```r
train_new_predictors <- select(train_new_norm,-Personal.Loan)
train_new_label <- select(train_new_norm,Personal.Loan)

val_new_predictors <- select(val_new_norm,-Personal.Loan)
val_new_label <- select(val_new_norm,Personal.Loan)

test_new_predictors <- select(test_new_norm,-Personal.Loan)
test_new_label <- select(test_new_norm,Personal.Loan)
```

```r
set.seed(428)
search_grid <- expand.grid(k=c(1:20))

train_new_norm$Personal.Loan <- factor(train_new_norm$Personal.Loan)

model_new <- train(Personal.Loan~.,train_new_norm,
                   method='knn',tuneGrid=search_grid,metric="Accuracy")

model_new
```

```
## k-Nearest Neighbors
##
```

```
## 2496 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2496, 2496, 2496, 2496, 2496, 2496, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9586049  0.7282167
##    2  0.9546113  0.7006498
##    3  0.9525784  0.6813123
##    4  0.9529633  0.6776276
##    5  0.9532574  0.6733820
##    6  0.9531245  0.6692184
##    7  0.9529202  0.6627532
##    8  0.9517406  0.6515338
##    9  0.9513847  0.6462971
##   10  0.9495967  0.6284242
##   11  0.9477760  0.6101437
##   12  0.9476050  0.6078071
##   13  0.9469029  0.6000013
##   14  0.9460671  0.5922751
##   15  0.9454996  0.5847386
##   16  0.9437561  0.5672650
##   17  0.9425956  0.5552687
##   18  0.9421520  0.5506337
##   19  0.9413148  0.5400740
##   20  0.9409726  0.5362114
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```r
#Below code prints the best K value from the knn model tuning
cat("Optimal K value for the dataset using the train method is ",
    as.character(model_new$bestTune[,"k"]))
```

```
## Optimal K value for the dataset using the train method is  1
```

Alternative way to find the best k value using the train and validation dataset

```r
val_new_label$Personal.Loan <- factor(val_new_label$Personal.Loan)
train_new_label$Personal.Loan <- factor(train_new_label$Personal.Loan)
test_new_label$Personal.Loan <- factor(test_new_label$Personal.Loan)


accuracydf <- data.frame(kValue=seq(1,14,1),Accuracy_Train=0,
                         Accuracy_Val=0,Accuracy_Test=0)

for(i in 1:nrow(accuracydf)){
```

```r
#val_label_predict <-  knn(train=train_new_predictors,test=val_new_predictors,
#                          cl=train_new_label$Personal.Loan,k=i)
train_new_label_Predicted <- knn(train_new_predictors,train_new_predictors,
                                 train_new_label$Personal.Loan,
                                 k=i)


accuracydf[i,2] <- confusionMatrix(train_new_label_Predicted, train_new_label$Personal.Loan,positive="1

val_new_label_Predicted <- knn(train_new_predictors,val_new_predictors,
                               train_new_label$Personal.Loan,
                                 k=i)


accuracydf[i,3] <- confusionMatrix(val_new_label_Predicted,
                                   val_new_label$Personal.Loan,positive="1")$overall[1]


test_new_label_Predicted <- knn(train_new_predictors,test_new_predictors,
                                train_new_label$Personal.Loan,
                                 k=i)


accuracydf[i,4] <- confusionMatrix(test_new_label_Predicted,
                                   test_new_label$Personal.Loan,positive="1")$overall[1]


}
accuracydf
```

```
##    kValue Accuracy_Train Accuracy_Val Accuracy_Test
## 1       1      1.0000000    0.9593875     0.9610778
## 2       2      0.9831731    0.9573901     0.9600798
## 3       3      0.9767628    0.9587217     0.9650699
## 4       4      0.9731571    0.9553928     0.9590818
## 5       5      0.9711538    0.9553928     0.9630739
## 6       6      0.9687500    0.9573901     0.9580838
## 7       7      0.9635417    0.9547270     0.9610778
## 8       8      0.9599359    0.9520639     0.9600798
## 9       9      0.9611378    0.9487350     0.9560878
## 10     10      0.9583333    0.9447403     0.9560878
## 11     11      0.9579327    0.9414115     0.9540918
## 12     12      0.9567308    0.9414115     0.9500998
## 13     13      0.9563301    0.9394141     0.9520958
## 14     14      0.9515224    0.9407457     0.9510978
```

k=1 has accuracy 1 which could mean there is chance of overfitting. Validation and Test has lesser accuracy

k=3 has best accuracy considering all three datasets train, validation and test

```
bestk_alternativeOption_1 <- 3
#accuracydf[which.max(accuracydf$Accuracy),][1]

cat("Alternative Approach - Optimal K value for the dataset is ",
    as.character(bestk_alternativeOption_1))
```

```
## Alternative Approach - Optimal K value for the dataset is  3
```

knn output for train dataset

```
train_new_label_Predicted <- knn(train_new_predictors,train_new_predictors,
                                 train_new_label$Personal.Loan,
                                 k=bestk_alternativeOption_1)

head(train_new_label_Predicted)
```

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

knn output for validation data set

```
val_new_label_Predicted <- knn(train_new_predictors,val_new_predictors,
                               train_new_label$Personal.Loan,
                                 k=bestk_alternativeOption_1)

head(val_new_label_Predicted)
```

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

knn output for test data set

```
test_new_label_Predicted <- knn(train_new_predictors,test_new_predictors,
                                train_new_label$Personal.Loan,
                                 k=bestk_alternativeOption_1)

head(test_new_label_Predicted)
```

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

```r
confusionMatrix(train_new_label_Predicted,train_new_label$Personal.Loan,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2257   58
##          1    1  180
##
##                Accuracy : 0.9764
##                  95% CI : (0.9696, 0.982)
##     No Information Rate : 0.9046
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8465
##
##  Mcnemar's Test P-Value : 3.086e-13
##
##             Sensitivity : 0.75630
##             Specificity : 0.99956
##          Pos Pred Value : 0.99448
##          Neg Pred Value : 0.97495
##              Prevalence : 0.09535
##          Detection Rate : 0.07212
##    Detection Prevalence : 0.07252
##       Balanced Accuracy : 0.87793
##
##        'Positive' Class : 1
##
```

```r
confusionMatrix(val_new_label_Predicted,val_new_label$Personal.Loan,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1345   59
##          1    3   95
##
##                Accuracy : 0.9587
##                  95% CI : (0.9474, 0.9682)
##     No Information Rate : 0.8975
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7326
##
##  Mcnemar's Test P-Value : 2.848e-12
##
##             Sensitivity : 0.61688
##             Specificity : 0.99777
##          Pos Pred Value : 0.96939
##          Neg Pred Value : 0.95798
```

```
##              Prevalence : 0.10253
##          Detection Rate : 0.06325
##    Detection Prevalence : 0.06525
##       Balanced Accuracy : 0.80733
##
##        'Positive' Class : 1
##
```

```r
confusionMatrix(test_new_label_Predicted,test_new_label$Personal.Loan,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 910  31
##          1   4  57
##
##                Accuracy : 0.9651
##                  95% CI : (0.9518, 0.9756)
##     No Information Rate : 0.9122
##     P-Value [Acc > NIR] : 2.386e-11
##
##                   Kappa : 0.7469
##
##  Mcnemar's Test P-Value : 1.109e-05
##
##             Sensitivity : 0.64773
##             Specificity : 0.99562
##          Pos Pred Value : 0.93443
##          Neg Pred Value : 0.96706
##              Prevalence : 0.08782
##          Detection Rate : 0.05689
##    Detection Prevalence : 0.06088
##       Balanced Accuracy : 0.82168
##
##        'Positive' Class : 1
##
```

```r
trainAccuracy <- confusionMatrix(train_new_label_Predicted,
                                 train_new_label$Personal.Loan,positive="1")$overall[1]

validationAccuracy <- confusionMatrix(val_new_label_Predicted,
                                      val_new_label$Personal.Loan,positive="1")$overall[1]

testAccuracy <- confusionMatrix(test_new_label_Predicted,
                                test_new_label$Personal.Loan,positive="1")$overall[1]

cat("The accuracy of train, validation, and test datasets observed using their\n
    confusion matrices are ",as.character(round(100*trainAccuracy,2)),"%, ",
    as.character(round(100*validationAccuracy,2)),"%, and ",
    as.character(round(100*testAccuracy,2)),"%.\n
    The test and validation data accuracy are important in
    determining the k value. \nFor the value k=",
```

```
    as.character(bestk_alternativeOption_1),", train, test,
    and validation data predicted outcomes accuracy improved.")
```

```
## The accuracy of train, validation, and test datasets observed using their
##
##     confusion matrices are  97.64 %,  95.87 %, and  96.51 %.
##
##     The test and validation data accuracy are important in
##     determining the k value.
## For the value k= 3 , train, test,
##     and validation data predicted outcomes accuracy improved.
```