FML_knn_classification

Thanasit C.

2024-02-14

1. Summary

1). All Answers are in the Section 5. Below is a brief answer.
Question_1; customer accept the offer.
Question_2; Best k in this model is 3. As I done 15 training cycles, I found that the Best K can be either k = 3 (70%) or k = 1 (30%). And average model sensitivity is around 65%. The variation of k might be caused by the low number of observations or low number of training cycles.
Question_3; see detail in section 5.3.
Question_4; with k = 3, customer will accept the offer with the probability = 0.667.
Question_5; outcome is quite similar, sensitivity is tiny bit different but not significant. The main reason caused the different is the different norm curve.
2). 'class::knn', 'caret::train', and 'caret::knn3' functions are use in this analysis.
3). Please note that there is a loop function under section 5.2. As currently set training cycle to 15 times, it should take a couple minutes to execute. You can change the number of cycles at 'traningcycle'.
4). I don't fully understand question_5, so my work is based on my understanding. I do 2 times partition process to split train:validation:test, I compare confusion matrix of 'train-test' and 'validation-test' with K = best K found in section 5.2

2. Libraries

```r
library(dplyr)
library(caret)
library(class)
library(ggplot2)
library(ggpubr)
library(reshape2)
```

3. Import data

3.1. Set working directory
3.2. Import csv file as dataframe format
3.3. Check data structure

```r
## set working directory
setwd("/Users/sieng/Documents/Study/MS.Business Analytics/SPRING 2024/Fundamental of Machine Learning/Assignment/Assignment 2")

## import data
maindf <- read.csv("UniversalBank.csv") %>% as.data.frame()
```

```r
## check data structure
str(maindf)

## 'data.frame':    5000 obs. of  14 variables:
##  $ ID                : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP.Code          : int  91107 90089 94720 94112 91330 92121 91711
93943 90089 93023 ...
##  $ Family            : int  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : int  1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal.Loan     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ Securities.Account: int  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD.Account        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : int  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : int  0 0 0 0 1 0 0 1 0 0 ...
```

4. Data wrangling and prepairation

4.1. handle missing value

```r
## 1) Find N/A value
sumna <- sum(is.na(maindf))
print("Number of N/A values in data set")

## [1] "Number of N/A values in data set"

sumna

## [1] 0

colsumna <- colSums(is.na(maindf))
print("Number of N/A by column")

## [1] "Number of N/A by column"

colsumna

##                 ID                Age         Experience
Income
##                  0                  0                  0
0
##           ZIP.Code             Family              CCAvg
Education
##                  0                  0                  0
0
##           Mortgage      Personal.Loan Securities.Account
CD.Account
```

```
##                   0                  0                  0
0
##              Online         CreditCard
##                   0                  0

## 2) Handle N/A value by remove or fill
### 2.1) remove entired row
maindf <- na.omit(maindf)

### 2.2) impute value
#### 2.2.1) fill by column average
#### 2.2.2) fill by kNN average
```

4.2. Rearrange and remove irrelavent column
Remove ID and Zip.cod column. Since 'Personal.Loan' is our independent variable, move the column to the first one.

```
### 4.2 Rearrange and remove unrelated column ###############
maindf2 <- maindf[,-c(1,5)] %>% select(Personal.Loan, everything())
str(maindf2)

## 'data.frame':    5000 obs. of  12 variables:
##  $ Personal.Loan     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ Family            : int  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : int  1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ Securities.Account: int  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD.Account        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : int  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : int  0 0 0 0 1 0 0 1 0 0 ...
```

4.3. convert 'Personal.loan' form 'c(0,1)' to c("Decline", "Accept")
for a better understanding

```
maindf2$Personal.Loan <- maindf2$Personal.Loan %>%
                           gsub(pattern = 0, replacement = "Decline") %>%
                           gsub(pattern = 1, replacement = "Accept")
```

4.4. Reassign data attribute

```
### 4.4 correcting data attributes ###############
# number()/integer()

# factor()
maindf2$Personal.Loan <- factor(maindf2$Personal.Loan)
maindf2$Securities.Account <- factor(maindf2$Securities.Account)
maindf2$CD.Account <- factor(maindf2$CD.Account)
```

```r
maindf2$Online <- factor(maindf2$Online)
maindf2$CreditCard <- factor(maindf2$CreditCard)
maindf2$Education <- factor(maindf2$Education, ordered = FALSE)

str(maindf2)
```

```
## 'data.frame':    5000 obs. of  12 variables:
##  $ Personal.Loan     : Factor w/ 2 levels "Accept","Decline": 2 2 2 2 2 2
2 2 2 1 ...
##  $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ Family            : int  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 3 2 3
...
##  $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ Securities.Account: Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
##  $ CD.Account        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Online            : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
##  $ CreditCard        : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
```

4.5. Dummy variable
convert 'Education' which is 3 levels factor to be dummy variables using function 'dummyVars'.

```r
# Create dummy variable
dummy_edu <- dummyVars("~Education", data = maindf2)
education_dummy <- data.frame(predict(dummy_edu, newdata = maindf2))

maindf3 <- cbind(maindf2, education_dummy)
# remover original 'Education' column and Education.3 to prevent perfect
linearily
maindf3 <- maindf3[,-7]
maindf3$Education.1 <- factor(maindf3$Education.1)
maindf3$Education.2 <- factor(maindf3$Education.2)
maindf3$Education.3 <- factor(maindf3$Education.3)
str(maindf3)
```

```
## 'data.frame':    5000 obs. of  14 variables:
##  $ Personal.Loan     : Factor w/ 2 levels "Accept","Decline": 2 2 2 2 2 2
2 2 2 1 ...
##  $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ Family            : int  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ Securities.Account: Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
##  $ CD.Account        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ Online              : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
##  $ CreditCard          : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
##  $ Education.1         : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
##  $ Education.2         : Factor w/ 2 levels "0","1": 1 1 1 2 2 2 2 1 2 1 ...
##  $ Education.3         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 2 ...
```

5. Modelling and problem solving

5.1. Question_1
At default probability = 0.5. 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, and CreditCard = 1. using k = 1. What is the classification of this customer?
Answer; This customer will ‘Accept’ the loan offer.

```
# I'm going to use 'class::knn' function to solve this problem. I will use
all data to predict the result with K = 1
## Create test data set
q1_customer <- 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)

## reassign data attributes
q1_customer$Education.1 <- factor(q1_customer$Education.1)
q1_customer$Education.2 <- factor(q1_customer$Education.2)
q1_customer$Securities.Account <- factor(q1_customer$Securities.Account)
q1_customer$CD.Account <- factor(q1_customer$CD.Account)
q1_customer$Online <- factor(q1_customer$Online)
q1_customer$CreditCard <- factor(q1_customer$CreditCard)

## normalize
q1_norm_processs <- caret::preProcess(maindf3, method = c("center", "scale"))
q1_train_norm <- predict(q1_norm_processs, maindf3)
q1_customer_norm <- predict(q1_norm_processs, q1_customer)

## Assumptions
q1_k = 1

## Predicting by using all data
```

```r
q1_prediction <- knn(train = q1_train_norm[,-1], test = q1_customer_norm, cl
= q1_train_norm[,1], k = q1_k, prob = TRUE)
q1_prediction
```

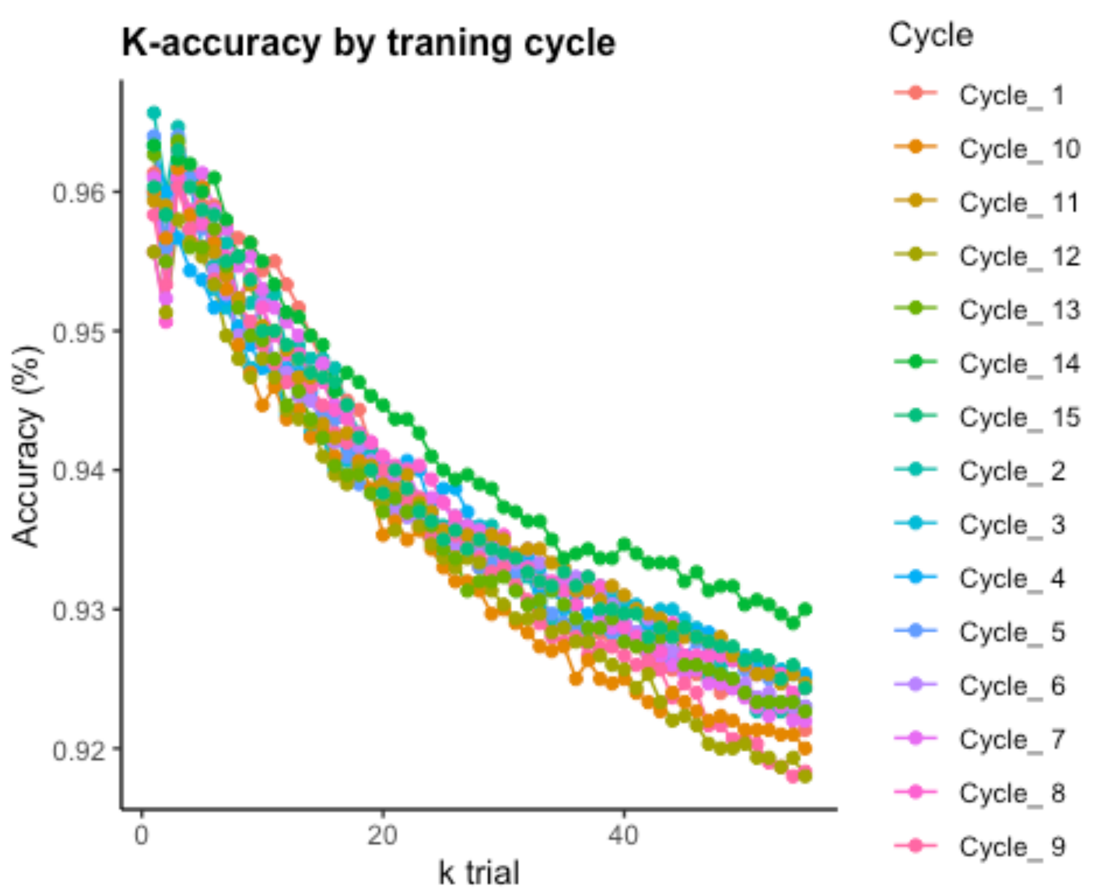
```
## [1] Accept
## attr(,"prob")
## [1] 1
## Levels: Accept Decline
```

5.2. Question_2
What is the choice of K?

```r
# In this question, I'm going to use 'caret::train' to find the optimal k-
value.
# Split training/testing ratio = 60:40
# Max k value is equal to square-root of number of observation use in the
model training
# Training cycle = 15 times > each time gets 1 optimal k-value. There are 15
optimal k-value in total then I'm choosing by most frequent number.
# Use 'caret::train' function to train model

## split train/test to 60:40
q2_trainsplit = 0.6

## training cycle
traningcycle = 15

## create output list
q2_koptimallist <- list()
q2_kaccuracylist <- list()
q2_performancelist <- list()


for (i in 1:traningcycle){
  print(i)
  ## create split partition
  q2_trainsplit_index <- caret::createDataPartition(y =
maindf3$Personal.Loan, p = q2_trainsplit, list = FALSE)
  q2_train <- maindf3[q2_trainsplit_index,]
  q2_test <- maindf3[-q2_trainsplit_index,]
  q2_test_label <- q2_test$Personal.Loan

  ## max k
  maxk <- round(sqrt(nrow(q2_train)),0)

  ## create number of k
  kgrid = expand.grid(k = c(1:maxk))

  ## normalize
  q2_norm_processs <- caret::preProcess(q2_train, method = c("center",
```

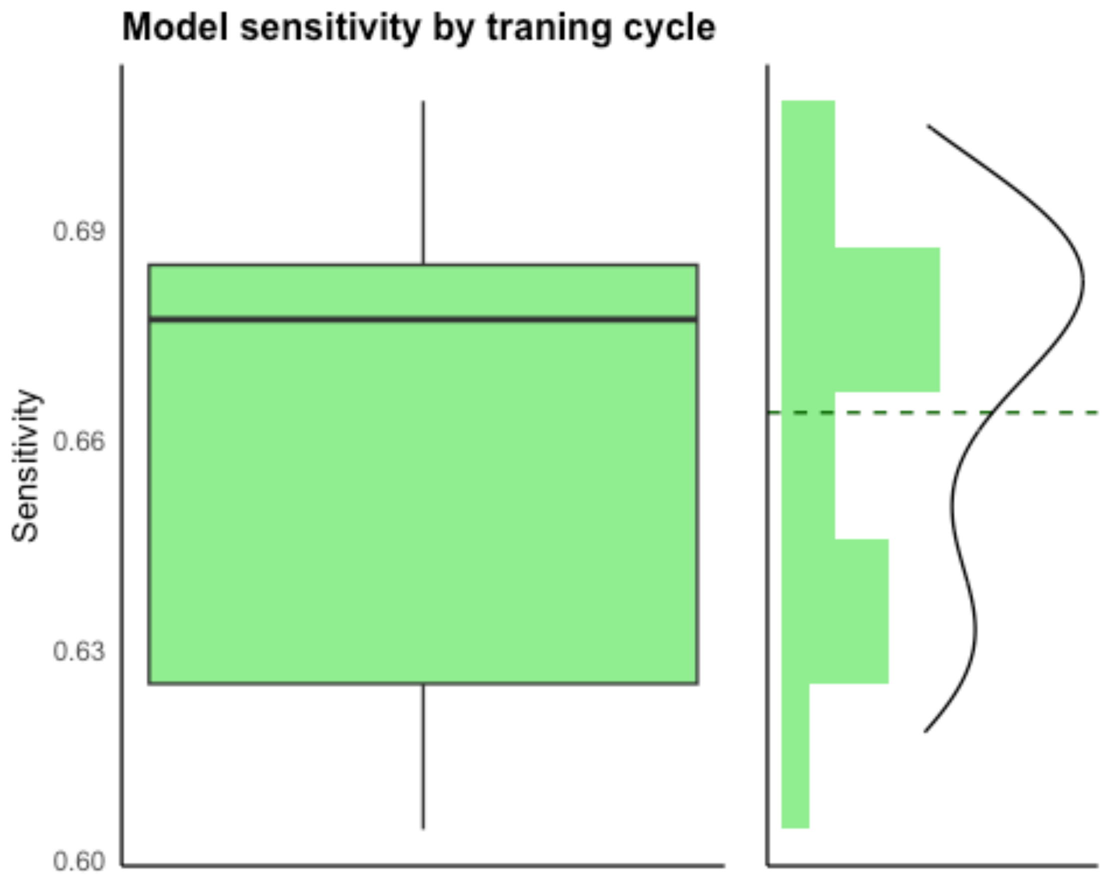
```r
"scale"))
  q2_train_norm <- predict(q2_norm_processs, q2_train)
  q2_test_norm <- predict(q2_norm_processs, q2_test)

  ## training control with 5-fold
  q2_ctrl <- caret::trainControl(method = "CV", number = 5)

  ## model fitting
  q2_fit <- caret::train(Personal.Loan ~.,
                         data = q2_train_norm,
                         method = "knn",
                         trControl = q2_ctrl,
                         tuneGrid = kgrid)

  ### result collecting - k-optimal
  q2_koptimallist[i] <- q2_fit[[6]]

  ### result collecting - k-accuracy
  kresults <- q2_fit[[4]] %>% as.data.frame()
  kresults <- kresults[,2]
  q2_kaccuracylist[[i]] <- as.vector(kresults)

  ## predict model
  q2_predict <- predict(q2_fit, newdata = q2_test_norm)
  q2_confusion_matrix <- confusionMatrix(q2_predict, q2_test_label)

  ### result collecting - Model performance
  q2_performancelist[i] <- q2_confusion_matrix[4]
}

koptimal <- unlist(q2_koptimallist)
koptimaldf <- as.data.frame(koptimal)

kaccuracy <- unlist(q2_kaccuracylist)
kaccuracydf <- as.data.frame(matrix(kaccuracy, ncol = maxk, byrow = TRUE))
row.names(kaccuracydf) <- paste("Cycle_", seq(1:traningcycle))
nrow_kaccuracydf <- nrow(kaccuracydf)+1
kaccuracydf[nrow_kaccuracydf,] <- seq(1:maxk)
row.names(kaccuracydf)[nrow_kaccuracydf] <- "K"
kaccuracydf <- t(kaccuracydf) %>% as.data.frame() %>% select(K, everything())
kaccuracydf <- tidyr::gather(kaccuracydf, "Cycle", "perAccuracy", -K)

modelperformance <- unlist(q2_performancelist)
modelperformancedf <- as.data.frame(matrix(modelperformance, ncol = 11, byrow
= TRUE))
names(modelperformancedf) <- c("Sensitivity",
                               "Specificity",
                               "Pos Pred Valuer",
                               "Neg Pred Value",
```

```
                                "Precision",
                                "Recall",
                                "F1",
                                "Prevalence",
                                "Detection Rate",
                                "Detection Prevalence",
                                "Balanced Accuracy")
modelperformancedf$Validation <- c(1:traningcycle)

## plot K-accuracy by testing cycle
kaccuracydf %>% ggplot(aes(x = K, y = perAccuracy)) +
                  geom_line(aes(color = Cycle)) +
                  geom_point(aes(color = Cycle)) +
                  labs(title = "K-accuracy by traning cycle") +
                  xlab(label = "k trial") +
                  ylab(label = "Accuracy (%)") +
                  theme_classic() +
                  theme(plot.title = element_text(face = "bold", size = 12))
```

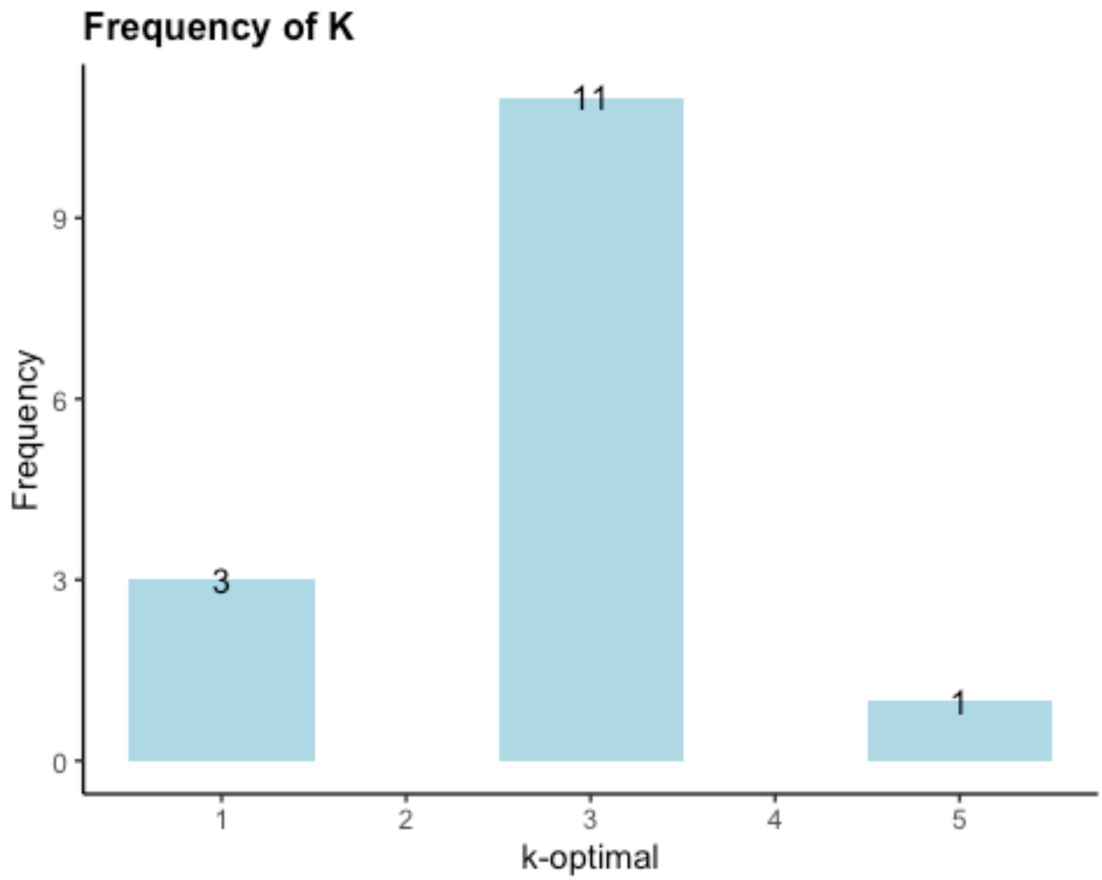



```
## plot model sensitivity by testing cycle
plotboxtmodelsensi <- modelperformancedf %>% ggplot(aes(y = Sensitivity)) +
                                                geom_boxplot(fill =
"lightgreen") +
```

```
                                                labs(title = "Model
sensitivity by traning cycle") +
                                                theme_classic() +
                                                theme(axis.title.x =
element_blank(),
                                                      axis.text.x =
element_blank(),
                                                      axis.ticks =
element_blank(),
                                                      legend.position = "top",
                                                      plot.title =
element_text(face = "bold", size = 12))

plothismodelsense <- modelperformancedf %>% ggplot(aes(x = Sensitivity)) +
                                               geom_histogram(fill =
"lightgreen", binwidth = 0.025) +
                                               geom_vline(mapping =
aes(xintercept = mean(Sensitivity)),
                                                          color = "darkgreen",
                                                          linetype = "dashed")
+
                                               stat_density(geom = "line") +
                                               theme_classic() +
                                               theme(axis.title.x =
element_blank(),
                                                     axis.text.x =
element_blank(),
                                                     axis.title.y =
element_blank(),
                                                     axis.text.y =
element_blank(),
                                                     axis.ticks =
element_blank()) +
                                               coord_flip()

ggarrange(plotboxtmodelsensi, plothismodelsense,
          ncol = 2, nrow = 1,
          widths = c(2,1), heights = c(1,1),
          common.legend = TRUE,
          align = "h")
```




```
## K optimal Histogram
koptimaldf %>% ggplot(aes(x = koptimal)) +
                 geom_histogram(fill = "lightblue", binwidth = 1) +
                 stat_count(aes(label = ..count..), geom = "text") +
                 labs(title = "Frequency of K") +
                 xlab(label = "k-optimal") +
                 ylab(label = "Frequency") +
                 theme_classic() +
                 theme(plot.title = element_text(face = "bold", size = 12))
```




```
## The most frequent K
### Create function
getk <- function(k) {
   uniqv <- unique(k)
   uniqv[which.max(tabulate(match(k, uniqv)))]
}
### find k
vec_koptiomal <- as.vector(koptimaldf$koptimal)
bestk <- getk(vec_koptiomal)
print(paste("Best K is", bestk))

## [1] "Best K is 3"
```

5.3 Question_3
Show the confusion matrix for the validation data that results from using the best k.

```
# Use data and variables from question 2
# Use 'caret::knn3' to train model

## k value = bestk
q3_k = bestk
```

```r
## normalize
q3_norm_processs <- caret::preProcess(q2_train, method = c("center",
"scale"))
q3_train_norm <- predict(q3_norm_processs, q2_train)

## train model
q3_fit <- caret::knn3(Personal.Loan ~., data = q3_train_norm, k = q3_k)
q3_prediction <- predict(q3_fit, q2_test_norm, type = "class")

## confusion matrix
q3_confusionmatrix <- confusionMatrix(q3_prediction, q2_test_label)
q3_confusionmatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Accept Decline
##    Accept     116       7
##    Decline     76    1801
##
##                Accuracy : 0.9585
##                  95% CI : (0.9488, 0.9668)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7152
##
##  Mcnemar's Test P-Value : 8.395e-14
##
##             Sensitivity : 0.6042
##             Specificity : 0.9961
##          Pos Pred Value : 0.9431
##          Neg Pred Value : 0.9595
##              Prevalence : 0.0960
##          Detection Rate : 0.0580
##    Detection Prevalence : 0.0615
##       Balanced Accuracy : 0.8001
##
##        'Positive' Class : Accept
##
```

5.4 Question_4

Consider the following customer: 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. Classify the customer using the best k.

Answer:

The probability is 0.667 which higher that 0.5 so the customer will accept the loan offered

```
# Use customer data from question1
# Use 'class::knn' to train model
# Besr K from question2

## Best K
q4_k = bestk

## Predicting by using all data
q4_prediction <- knn(train = q1_train_norm[,-1], test = q1_customer_norm, cl
= q1_train_norm[,1], k = q4_k, prob = TRUE)
q4_prediction

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

5.5 Question_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. Comment on the differences and their reason.

Since I am not fully understand the question, I will explain things that I've done for this particular question.

I do 2 times partition process to split train:validation:test, I compare confusion matrix of 'train-test' and 'validation-test' with K = best K found in section 5.2

```
# Use 'caret::train' function to train model.
## train/test proportion
q5_testsplit = 0.8

## validation proportion
q5_validationsplit = 0.375

## split test partition
q5_testsplit_index <- caret::createDataPartition(y = maindf3$Personal.Loan, p
= q5_testsplit, list = FALSE)
q5_therest <- maindf3[q5_testsplit_index,]
q5_test <- maindf3[-q5_testsplit_index,]
q5_test_label <- q5_test$Personal.Loan

## split validation partition
q5_validationsplit_index <- caret::createDataPartition(y =
q5_therest$Personal.Loan, p = q5_validationsplit, list = FALSE)
q5_validation <- q5_therest[q5_validationsplit_index,]
q5_train <- q5_therest[-q5_validationsplit_index,]

## k
q5_k = bestk
```

```r
## normalize
q5_train_norm_processs <- caret::preProcess(q5_train, method = c("center",
"scale"))
q5_train_norm <- predict(q5_train_norm_processs, q5_train)
q5_test_train_norm <- predict(q5_train_norm_processs, q5_test)

q5_validation_norm_processs <- caret::preProcess(q5_validation, method =
c("center", "scale"))
q5_validation_norm <- predict(q5_validation_norm_processs, q5_validation)
q5_test_validation_norm <- predict(q5_validation_norm_processs, q5_test)

## training control
q5_ctrl <- caret::trainControl(method = "CV", number = 5)

## fitting train
q5_train_fit <- caret::knn3(Personal.Loan ~., data = q5_train_norm, k = q5_k)

## fitting validation
q5_validation_fit <- caret::knn3(Personal.Loan ~., data = q5_validation_norm,
k = q5_k)

## predict - train model
q5_predict_train <- predict(q5_train_fit, newdata = q5_test_train_norm, type
= "class")

## predict - validation model
q5_predict_validation <- predict(q5_train_fit, newdata =
q5_test_validation_norm, type = "class")

## confusion matrix train vs test
q5_confm_traintest <- confusionMatrix(q5_predict_train, q5_test_label)
q5_confm_traintest

## Confusion Matrix and Statistics
##
##             Reference
## Prediction Accept Decline
##    Accept      58       0
##    Decline     38     904
##
##                Accuracy : 0.962
##                  95% CI : (0.9482, 0.973)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 2.140e-12
##
##                   Kappa : 0.734
##
##  Mcnemar's Test P-Value : 1.947e-09
```

```
##
##             Sensitivity : 0.6042
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9597
##              Prevalence : 0.0960
##          Detection Rate : 0.0580
##    Detection Prevalence : 0.0580
##       Balanced Accuracy : 0.8021
##
##        'Positive' Class : Accept
##
```

```r
## confusion matrix validation vs test
q5_confm_validationtest <- confusionMatrix(q5_predict_validation,
q5_test_label)
q5_confm_validationtest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Accept Decline
##    Accept      57       0
##    Decline     39     904
##
##                Accuracy : 0.961
##                  95% CI : (0.9471, 0.9721)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 5.695e-12
##
##                   Kappa : 0.7255
##
##  Mcnemar's Test P-Value : 1.166e-09
##
##             Sensitivity : 0.5938
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9586
##              Prevalence : 0.0960
##          Detection Rate : 0.0570
##    Detection Prevalence : 0.0570
##       Balanced Accuracy : 0.7969
##
##        'Positive' Class : Accept
##
```