Week2_HW

2023-09-06

#Question 3.1

Using the same data set (credit_card_data.txt or credit_card_data-headers.txt) as in Question 2.2, use the ksvm or kknn function to find a good classifier: (a) using cross-validation (do this for the k-nearest-neighbors model; SVM is optional);

```
#import library
library(kknn)
library(tidyr)
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(lattice)
library(caret)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:kknn':
##
##     contr.dummy
```

```
library(readr)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.2.1     v stringr 1.5.0
## v purrr   1.0.2     v forcats 1.0.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(kernlab)
```

```
##
```

```
## Attaching package: 'kernlab'

## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(NbClust)
library(ISLR)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
#create a random sample and reproducible.
set.seed(1234)

#load data from txt doc.
cc_data <- read.table("credit_card_data.txt", stringsAsFactors = FALSE, header = FALSE)
head(cc_data)
```

```
##   V1    V2    V3   V4 V5 V6 V7 V8  V9 V10 V11
## 1  1 30.83 0.000 1.25  1  0  1  1 202   0   1
## 2  0 58.67 4.460 3.04  1  0  6  1  43 560   1
## 3  0 24.50 0.500 1.50  1  1  0  1 280 824   1
## 4  1 27.83 1.540 3.75  1  0  5  0 100   3   1
## 5  1 20.17 5.625 1.71  1  1  0  1 120   0   1
## 6  1 32.08 4.000 2.50  1  1  0  0 360   0   1
```

```
#Generate a random sample of 80% of the rows
split_sample <- createDataPartition(cc_data$V11, p = 0.8, list = FALSE)

#Assign the train data set to 80% of the orginal data
train_data = cc_data[split_sample,]
#Assign the testData set to the remaining 20% of the original data
test_data = cc_data[-split_sample,]

#Use LOOCV to determind best K- value
Loocv_model <- train.kknn((V11)~., data = train_data, kmax = 100, distance =1, scale = TRUE)
summary(Loocv_model)
```

```
##
## Call:
## train.kknn(formula = (V11) ~ ., data = train_data, kmax = 100,     distance = 1, scale = TRUE)
##
## Type of response variable: continuous
## minimal mean absolute error: 0.1965649
## Minimal mean squared error: 0.1101358
## Best kernel: optimal
## Best k: 32
```

We got the best k value is 32 and best kernel is optimal for from the training data set. Next, we plug in

k=32 and using kernel = "optimal" into the train data set to get the train accuracy.

```
#run the accuracy check for train_data.
predicted_train <- rep(0,(nrow(train_data))) # predictions: start with a vector of all zeros
train_accuracy<- 0  #initialize variable

for (i in 1:nrow(train_data)){
  # use scaled data
  model=kknn(V11~.,train_data[-i,],train_data[i,],k=32,kernel="optimal", scale = TRUE)
   # round off to 0 or 1
  predicted_train[i]<- as.integer(fitted(model)+0.5)
}

# calculate fraction of correct predictions
train_accuracy<- sum(predicted_train == train_data[,11]) / nrow(train_data)

#the result of train data with best k=32.
train_accuracy
```

```
## [1] 0.8492366
```

We got the train accuracy is 0.8492366. Then we plug in k=32 and using kernel = "optimal" into the test data set to get the test accuracy.

```
predicted_test <- rep(0,(nrow(test_data))) # predictions: start with a vector of all zeros
test_accuracy<- 0 #initialize variable

for (i in 1:nrow(test_data)){
  # use scaled data
  model=kknn(V11~.,test_data[-i,],test_data[i,],k=32,kernel="optimal", scale = TRUE)
  # round off to 0 or 1
  predicted_test[i]<- as.integer(fitted(model)+0.5)
}

# calculate fraction of correct predictions
test_accuracy<- sum(predicted_test == test_data[,11]) / nrow(test_data)

#the result of test data with best k=8.
test_accuracy
```

```
## [1] 0.8538462
```

Compare the train accuracy 0.8492366 to test accuracy 0.8538462. As we split 80% for the train data and 20% of the test data. We are expecting the test accuracy should be lower than the train accuracy. In this case, my test accuracy is higher than the train accuaracy. I would say my model of cross validation is too optimistic.

(b) splitting the data into training, validation, and test data sets (pick either KNN or SVM; the other is optional).

Splited the data for 80% train data, 10% validation data and 10% test data.

```
#create a random sample and reproducible.
set.seed(1234)

#load data from txt doc.
cc_data <- read.table("credit_card_data.txt", stringsAsFactors = FALSE, header = FALSE)
head(cc_data)
```

```
##   V1    V2    V3   V4 V5 V6 V7 V8  V9 V10 V11
## 1  1 30.83 0.000 1.25  1  0  1  1 202   0   1
## 2  0 58.67 4.460 3.04  1  0  6  1  43 560   1
## 3  0 24.50 0.500 1.50  1  1  0  1 280 824   1
## 4  1 27.83 1.540 3.75  1  0  5  0 100   3   1
## 5  1 20.17 5.625 1.71  1  1  0  1 120   0   1
## 6  1 32.08 4.000 2.50  1  1  0  0 360   0   1
```

```
#Generate a random sample of 80% of the rows
splitsample <- createDataPartition(cc_data$V1, p = 0.8, list = FALSE)

#Assign the train data set to 80% of the orginal data
trainData = cc_data[splitsample,]
#Assign the remaining data set for 20% of the original data
remainData = cc_data[-splitsample,]

#Generate a half of random sample of the remaining rows in remaining Data
splitsample2 <-sample(1:nrow(remainData),as.integer(0.5*nrow(remainData)))

#Assign the validation data set for 10% of the remaining data
valiData <- remainData[splitsample2,]

#Assgin the test data set for 10% of the remaining data
testData <- remainData[-splitsample2,]
#get the best k value for accuracy training.
loocv_model <- train.kknn((V11)~., data = train_data, kmax = 100,
           distance = 2, scale = TRUE)

summary(loocv_model)
```

```
##
## Call:
## train.kknn(formula = (V11) ~ ., data = train_data, kmax = 100,     distance = 2, scale = TRUE)
##
## Type of response variable: continuous
## minimal mean absolute error: 0.1927481
## Minimal mean squared error: 0.1119506
## Best kernel: optimal
## Best k: 44
```

In the train data set, we get the best k value is 44 and the best kernel is optimal. Now, we run through the model again on validation data set to get the a better k value and kernel.

```
#run knn with validation data set
kmax<-100
accuracy <- rep(0,kmax)

for (i in 1:kmax) {

  # fit k-nearest-neighbor model using training set, validate on validation set

  knn_model <- kknn(V11~.,trainData,valiData ,k=i,kernel = 'optimal',scale=TRUE)

  #  compare models using validation set

  pred <- as.integer(fitted(knn_model)+0.5) # round off to 0 or 1
```

```
  accuracy[i] = sum(pred == valiData[,11]) / nrow(valiData)
}

#find the hightest accuracy
kknn_result <- data.frame(accuracy)
View(kknn_result)

#find the best k value
best_kvalue <- which.max(accuracy)
View(best_kvalue)

print(paste0("The highest kknn accuracy rate is ", max(accuracy),
             " and the best k value is ", best_kvalue))
```

```
## [1] "The highest kknn accuracy rate is 0.876923076923077 and the best k value is 7"
```

Then we run through the model and get the accuracy model for the validation data. Here we find out the best k value is 7 instead of 44. And the validation accuracy is about 87.69% correct. We plug in k=7 and use the optimal kernel to run the test data.

```
#run knn with test data set using the best k=7 and optimal kernel we had from the validation set.

knnmodel <- kknn(V11~.,trainData,testData,k=7,kernel = 'optimal',scale=TRUE)

pred <- as.integer(fitted(knnmodel)+0.5) # round off to 0 or 1

#get the accuracy for the test data by using the best k value from validation data set
test_accuracy = sum(pred == testData[,11]) / nrow(testData)
test_accuracy
```

```
## [1] 0.8615385
```

```
print(paste0("We found the model accuracy rate for test dataset is ", test_accuracy))
```

```
## [1] "We found the model accuracy rate for test dataset is 0.861538461538462"
```

Compare the test accuracy 0.8615385 to validation accuracy 0.876923076923077. The same model run through validation data set is more accurate and not too optimistic.

#Question 4.1

##Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

I used to work as an e-commerce marketing analyst for online grocery store. We gather customers data into 4 different clusters. Age, locations, shopping preference, and order amount. We need to utilize different clusters to create specialize event for customers fall in different clusters. For instance, we may have large group of customers fall in location by Brooklyn. We will create an event or promotion sales for Brooklyn customer on a special day range. In this case, we are able to maintain the inventory, as well as the delivery traffic.

#Question 4.2

The iris data set iris.txt contains 150 data points, each with four predictor variables and one categorical response. The predictors are the width and length of the sepal and petal of flowers and the response is the type of flower. The data is available from the R library datasets and can be accessed with iris once the library is loaded. It is also available at the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Iris). The response values are only given to see how well a specific method performed and should not be used to build the model.

Use the R function kmeans to cluster the points as well as possible. Report the best combination of predictors, your suggested value of k, and how well your best clustering predicts flower type.

```
#load the data into a table
iris_data <- read.table("iris.txt", stringsAsFactors = FALSE, header = TRUE)

tail(iris_data)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 145          6.7         3.3          5.7         2.5 virginica
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
```

```
#Removed species to create cluster.
scaled_data <- scale(iris_data[1:4] )
k2 <- kmeans(scaled_data, centers = 2, nstart = 25)
k3 <- kmeans(scaled_data, centers = 3, nstart = 25)
k4 <- kmeans(scaled_data, centers = 4, nstart = 25)

# plots to compare
c1 <- fviz_cluster(k2, geom = "point", data = scaled_data) + ggtitle("k = 2")
c2 <- fviz_cluster(k3, geom = "point",  data = scaled_data) + ggtitle("k = 3")
c3 <- fviz_cluster(k4, geom = "point",  data = scaled_data) + ggtitle("k = 4")
```

Plot the graph below and showing the clusters in K from 2 to 5.

```
#let verify if the clusters are split reletive evenly.
table(k2$cluster, iris_data$Species)
```

```
##
##     setosa versicolor virginica
##   1      0         50        50
##   2     50          0         0
```

```
table(k3$cluster, iris_data$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         39        14
##   3      0         11        36
```

```
table(k4$cluster, iris_data$Species)
```

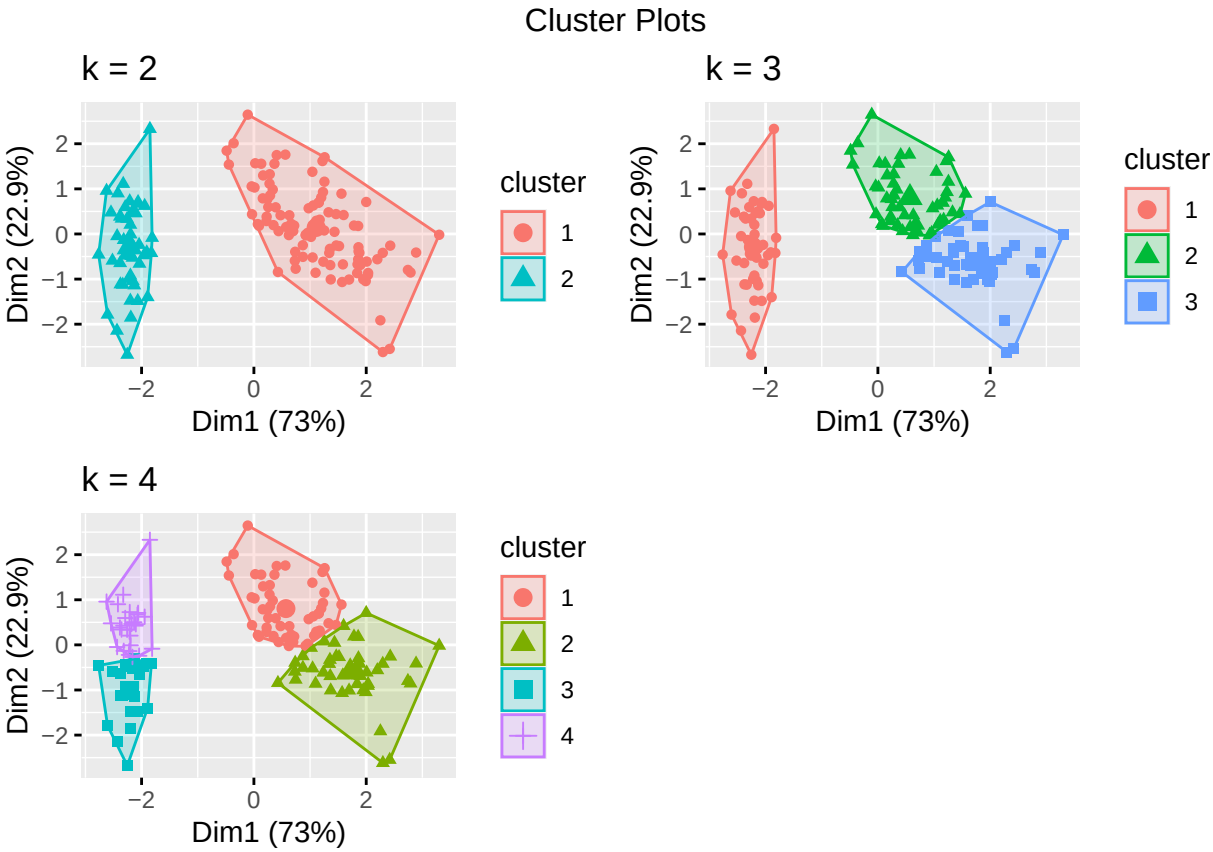
```
##
##     setosa versicolor virginica
##   1      0         39        14
##   2      0         11        36
##   3     25          0         0
##   4     25          0         0
```

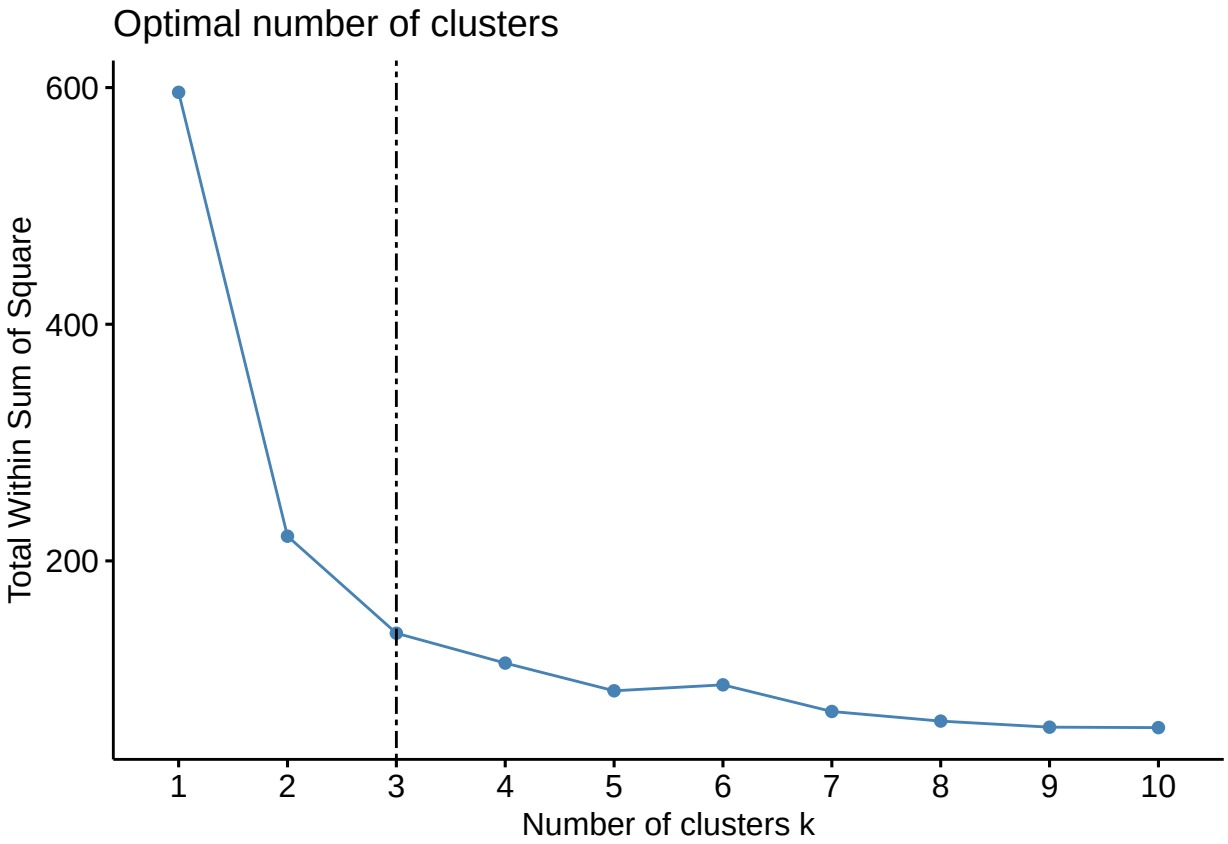
As we learned from the data set. We collected 50 data from each type of flowers. Based on the table we spited the data for each clusters. We can see when k=3 would have a better split on the data. We can separate setosa with 100% accuracy. About 78% accuracy on define versicolor and 72% accuracy on define virginica.

##

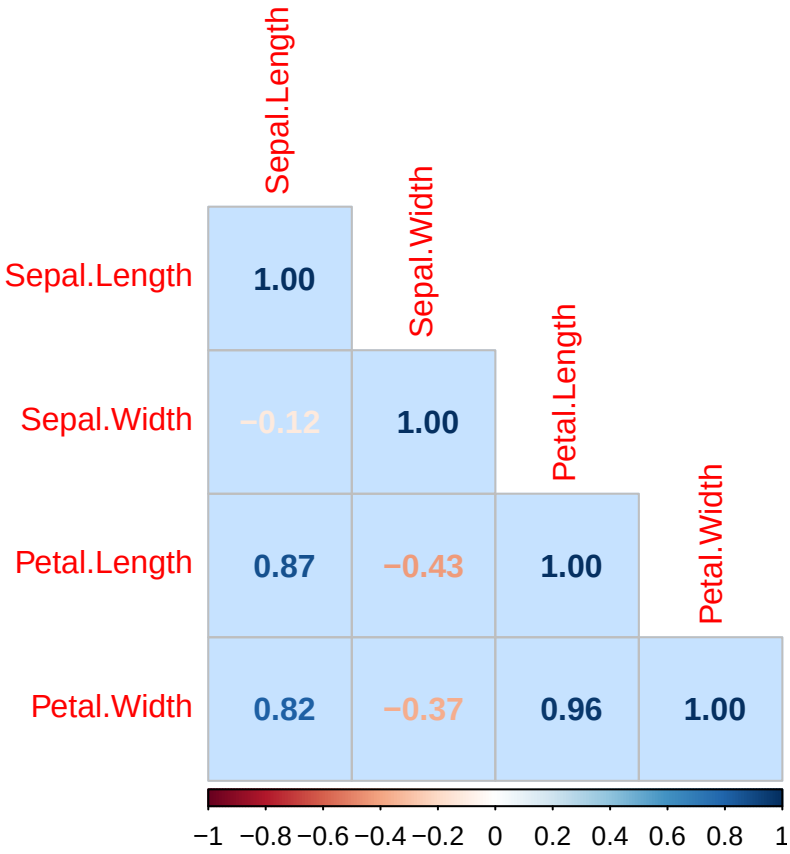
```
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```





We find out the best k from the elbow graph is 3. So I suggested categorize into 3 clusters is better for the iris data. After k=3, the more we had on creating clusters is gradually decrease on its cluster efficiency. Which means if we only create 3 difference clusters, the less misclassification we had for iris data.



Lastly, I am trying to find the correlation between Petal Length, Petal Width, Sepal Length, and Sepal Width. Here we can see from the graph. There are positive relationship between Petal Length, Petal Width, and Sepal Length.