ISYE HW2

2024-01-23

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)

Q3.1A

```
#Clear environment
rm(list=ls())

#Load package
library(kknn)

#Read data
data <- read.table("credit_card_data-headers.txt", header = TRUE)

#check data
head(data)
```

```
##   A1    A2    A3   A8 A9 A10 A11 A12 A14 A15 R1
## 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
```

```
tail(data)
```

```
##     A1    A2     A3   A8 A9 A10 A11 A12 A14 A15 R1
## 649  1 40.58  3.290 3.50  0   1   0   0 400   0  0
## 650  1 21.08 10.085 1.25  0   1   0   1 260   0  0
## 651  0 22.67  0.750 2.00  0   0   2   0 200 394  0
## 652  0 25.25 13.500 2.00  0   0   1   0 200   1  0
## 653  1 17.92  0.205 0.04  0   1   0   1 280 750  0
## 654  1 35.00  3.375 8.29  0   1   0   0   0   0  0
```

```
#Set seeds
set.seed(1)

#Set kmax
kmax <- 25

#Run the model and use train.kknn for leave-one-out cross validation
model <- train.kknn(R1~A1+A2+A3+A8+A9+A10+A11+A12+A14+A15, data, kmax = kmax,
```

```
scale = TRUE)

accuracy_percent <- rep(0, kmax)

for (k in 1: kmax){
  predicted <- as.integer(fitted(model)[[k]][1: nrow(data)] + 0.5)
  accuracy_percent[k] <- sum(predicted == data$R1)/ nrow(data) * 100
}

accuracy_percent
```

```
##  [1] 81.49847 81.49847 81.49847 81.49847 85.16820 84.55657 84.70948
84.86239
##  [9] 84.70948 85.16820 85.16820 85.32110 85.16820 85.16820 85.32110
85.32110
## [17] 85.32110 85.16820 85.01529 85.01529 84.86239 84.70948 84.40367
84.55657
## [25] 84.55657
```

```
max(accuracy_percent)
```

```
## [1] 85.3211
```

```
#Found that the lowest values of k (k<5) had the lowest percentage, therefore
the worst choice.
#12, 15, 16, and 17 had the highest accuracy percentage at 85.3211%.
```

(Q3.1B)

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

```
#Clear environment
rm(list=ls())

#Load package
library(kknn)

#Read data
data <- read.table("credit_card_data-headers.txt", header = TRUE)

#Set seeds, random number generator so results are reproducible
set.seed(1)

#split the data using 70% for training
#sample function randomly selects 70% of the data points
random_1 = sample(1:nrow(data), as.integer(0.7*nrow(data)))

#Assign the training data set to 70% of the data
train_set = data[random_1,]

#Assign the remaining 30% of the data to random set
remaining_data = data[-random_1,]
```

```r
#split in half the remaining data and generate a randomized sample
random_2 = sample(1:nrow(remaining_data),
as.integer(0.5*nrow(remaining_data)))

#Assign half of the remaining data to validation set
validation_set = remaining_data[random_2,]

#Assign the other half to the test set
test_set = remaining_data[-random_2,]

#Check the data
head(train_set)
```

```
##     A1    A2    A3    A8 A9 A10 A11 A12 A14  A15 R1
## 129  1 36.25 5.000 2.500  1   0   6   1   0  367  1
## 509  0 24.92 1.250 0.000  1   1   0   1  80    0  0
## 471  1 19.17 9.500 1.500  1   1   0   1 120 2206  1
## 299  1 19.00 1.750 2.335  0   1   0   0 112    6  0
## 270  1 23.92 0.585 0.125  0   1   0   1 240    1  0
## 187  1 42.00 0.205 5.125  1   1   0   1 400    0  1
```

```r
head(validation_set)
```

```
##     A1    A2     A3    A8 A9 A10 A11 A12 A14  A15 R1
## 278  1 23.00  0.750 0.500  0   1   0   0 320    0  0
## 184  0 28.67  1.040 2.500  1   0   5   0 300 1430  1
## 502  1 43.25 25.210 0.210  1   0   1   1 760   90  0
## 174  0 18.42  9.250 1.210  1   0   4   1  60  540  1
## 12   1 29.92  1.835 4.335  1   1   0   1 260  200  1
## 383  1 20.75  5.085 0.290  0   1   0   1 140  184  0
```

```r
head(test_set)
```

```
##    A1    A2    A3    A8 A9 A10 A11 A12 A14   A15 R1
## 9   1 54.42  0.50 3.960  1   1   0   1 180   314  1
## 23  0 47.75  8.00 7.875  1   0   6   0   0  1260  1
## 24  0 27.42 14.50 3.085  1   0   1   1 120    11  1
## 30  1 42.08  1.04 5.000  1   0   6   0 500 10000  1
## 52  1 26.00  1.00 1.750  1   1   0   0 280     0  1
## 54  1 34.92  2.50 0.000  1   1   0   0 239   200  1
```

```r
#Check # of rows in each data set to confirm split amount
nrow(train_set)
```

```
## [1] 457
```

```r
nrow(validation_set)
```

```
## [1] 98
```

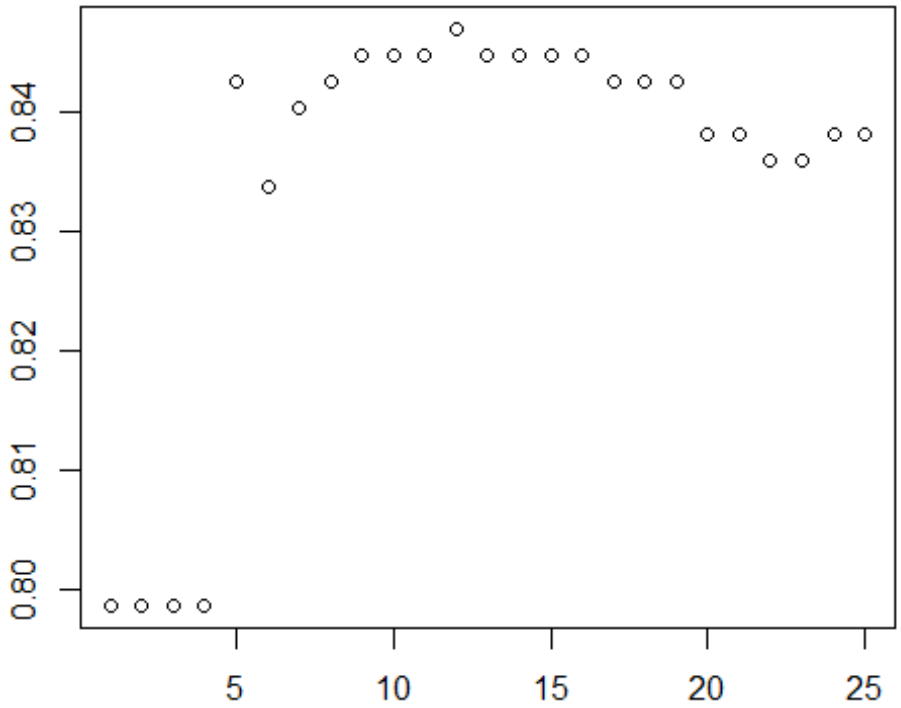
```r
nrow(test_set)
```

```
## [1] 99
```

```r
#test & build kknn models on training set
predicted_train = rep(0,(nrow(train_set)))
train_accuracy = 0
X = 0

accuracy = data.frame(matrix(nrow = 25, ncol = 2))

for (X in 1:25){
  for (i in 1: nrow(train_set)){
    model = kknn(R1~A1+A2+A3+A8+A9+A10+A11+A12+A14+A15, train_set[-i,],
train_set[i,], k = X, scale = TRUE)
    predicted_train[i] = as.integer(fitted(model)+0.5)
  }
#calculate fraction of correct predictions
  train_accuracy = sum(predicted_train == train_set[,11]) / nrow(train_set)
  accuracy[X, 1] = X
  accuracy[X, 2] = train_accuracy
}

#add titles to the accuracy table
colnames(accuracy) = c("k","accuracy")
accuracy
```

```
##     k  accuracy
## 1   1 0.7986871
## 2   2 0.7986871
## 3   3 0.7986871
## 4   4 0.7986871
## 5   5 0.8424508
## 6   6 0.8336980
## 7   7 0.8402626
## 8   8 0.8424508
## 9   9 0.8446389
## 10 10 0.8446389
## 11 11 0.8446389
## 12 12 0.8468271
## 13 13 0.8446389
## 14 14 0.8446389
## 15 15 0.8446389
## 16 16 0.8446389
## 17 17 0.8424508
## 18 18 0.8424508
## 19 19 0.8424508
## 20 20 0.8380744
## 21 21 0.8380744
## 22 22 0.8358862
## 23 23 0.8358862
## 24 24 0.8380744
## 25 25 0.8380744
```

```r
#plot accuracy table and adjust margin to fit
par(mar= c(2,2,2,2))
plot(accuracy[,1],accuracy[,2])
```




```r
#12 is the best fit for the training data. K values of 13-15 also performed
well.

#Testing different models
predicted_validate = rep(0,(nrow(validation_set)))
validate_accuracy = 0
X = 0

#Accuracy
accuracy_validate_table = data.frame(matrix(nrow =4, ncol = 2))
counter = 0

for (X in 12:15){
  counter = counter +1
  for (i in 1: nrow(validation_set)){
    model = kknn(R1~A1+A2+A3+A8+A9+A10+A11+A12+A14+A15, validation_set[-i,],
validation_set[i,], k = X, scale = TRUE)
    predicted_validate[i] = as.integer(fitted(model)+0.5)
  }
  #calculate fraction of correct predictions
  validate_accuracy = sum(predicted_validate == validation_set[,11]) /
nrow(validation_set)
  accuracy_validate_table[counter, 1] = X
```

```
  accuracy_validate_table[counter, 2] = train_accuracy
}

#Create table for k values and corresponding accuracy
colnames(accuracy_validate_table) = c('k', 'validate_accuracy')

accuracy_validate_table
```

```
##    k validate_accuracy
## 1 12         0.8380744
## 2 13         0.8380744
## 3 14         0.8380744
## 4 15         0.8380744
```

```
##the output shows the 4 models perform the same at 83.8%

#Testing accuracy on test data
predicted_test = rep(0,(nrow(test_set)))
test_accuracy = 0

for (i in 1: nrow(test_set)){
  model = kknn(R1~A1+A2+A3+A8+A9+A10+A11+A12+A14+A15, test_set[-i,],
test_set[i,], k = 12, scale = TRUE)
  predicted_test[i] = as.integer(fitted(model)+0.5)
}

#calculate fraction of correct predictions
test_accuracy = sum(predicted_test == test_set[,11]) / nrow(test_set)
test_accuracy
```

```
## [1] 0.7676768
```

```
#The actual accuracy turned out to be 77% which is lower than the accuracy
tested on the training data set
#This is likely due to random effects in the training data.
```

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.

A clustering model could be used to tailor my marketing strategies and improve guest satisfaction for my vacation home rental, by better identifying my customer base. I would use predictors such as guest demographics (couples, families, business travelers, solo-travelers), length of stay, location of customers, and property amenities. By applying a clustering model, I can use the information to improve marketing strategies, tailor the property by offering additional amenities and services, and pricing optimization.

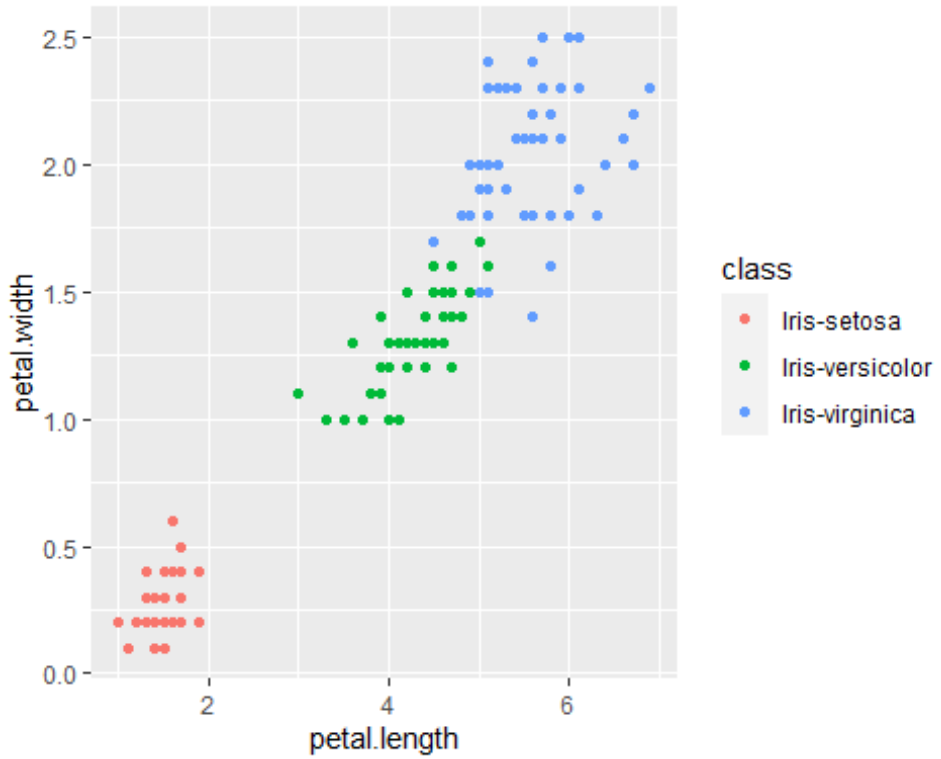
Question 4.2

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

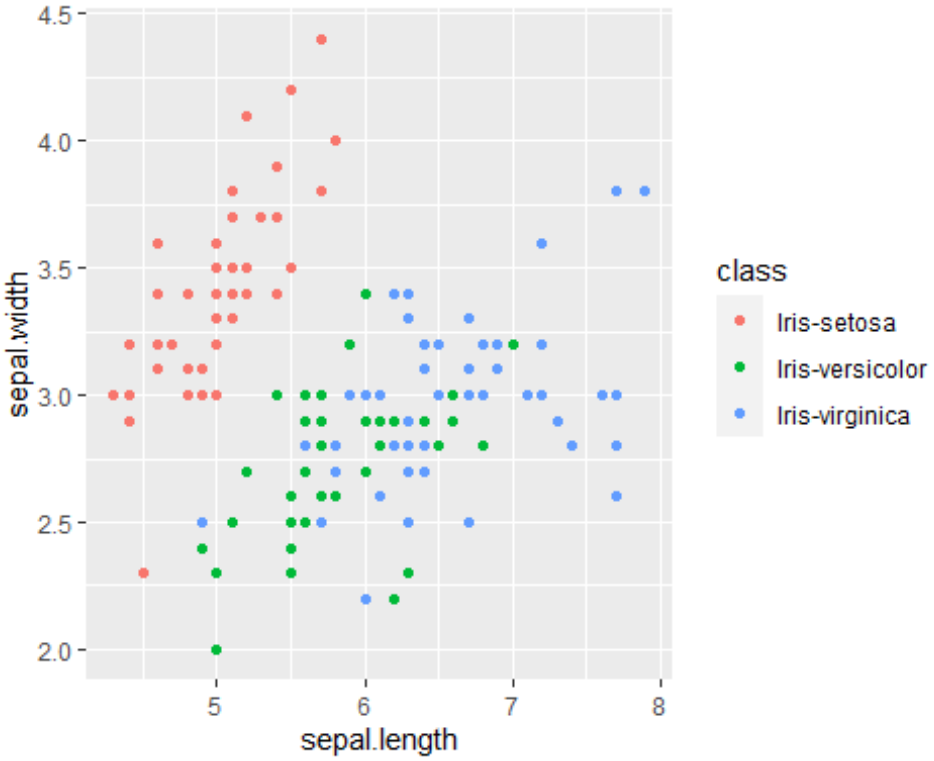
and the response is the type of flower. The data is available from the R library datasets and can be accessed with iris once the library is loaded. It is also available at the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Iris). The response values are only given to see how well a specific method performed and should not be used to build the model. Use the R function k-means 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.

```
#Clear environment
rm(list=ls())

#Read in the data
data <- read.table("iris.data", header = FALSE, sep = ",")

#name the columns according to the isris.names data file
v1 <- c("sepal.length", "sepal.width", "petal.length", "petal.width",
"class")
colnames(data) <- v1

#check features to make sure they return the same values in the iris.names
summary
min(data$"sepal.length")

## [1] 4.3

max(data$"petal.width")

## [1] 2.5

head(data)

##   sepal.length sepal.width petal.length petal.width       class
## 1          5.1         3.5          1.4         0.2 Iris-setosa
## 2          4.9         3.0          1.4         0.2 Iris-setosa
## 3          4.7         3.2          1.3         0.2 Iris-setosa
## 4          4.6         3.1          1.5         0.2 Iris-setosa
## 5          5.0         3.6          1.4         0.2 Iris-setosa
## 6          5.4         3.9          1.7         0.4 Iris-setosa

#check how many of each species there are
table(data$class)

##
##     Iris-setosa Iris-versicolor  Iris-virginica
##              50              50              50

#Find number of unique results which form a cluster
numClusters <- length(unique(data$class))
numClusters

## [1] 3
```

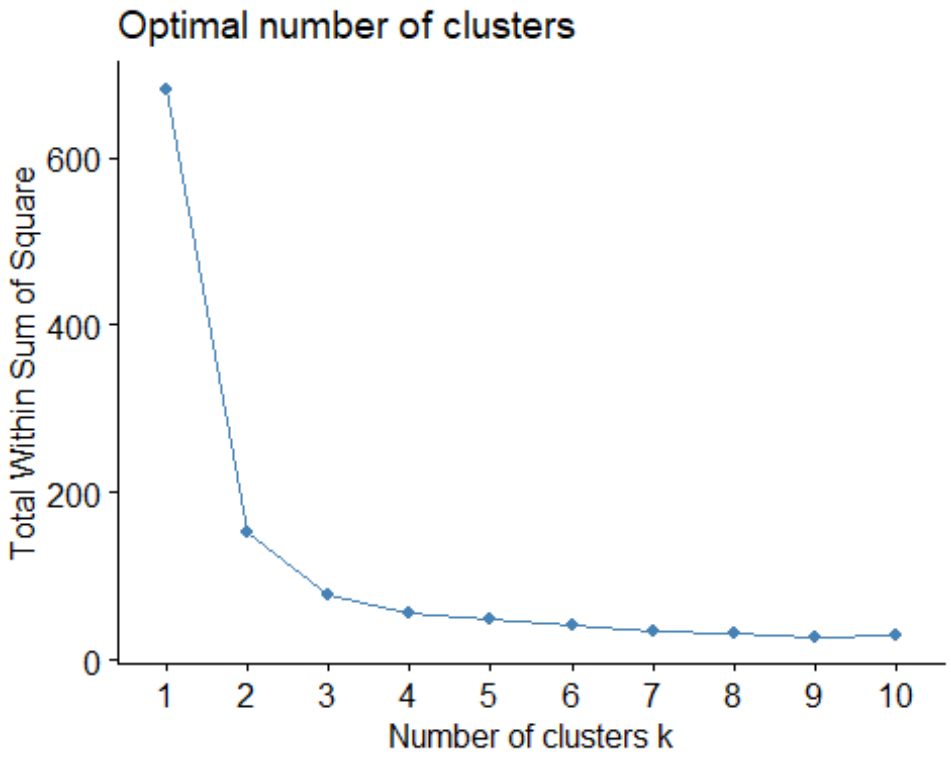
```
library(ggplot2)
ggplot(data, aes(petal.length, petal.width, color = class)) + geom_point()
```




```
ggplot(data, aes(sepal.length, sepal.width, color = class)) + geom_point()
```




```
#load package
library(factoextra)

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

#Remove column 5 which contains the class
data2 <- data[,1:4]

#use this function to evaluate and find the best number of clusters to test
plot <- fviz_nbclust(data2, kmeans, method ="wss")
plot
```




```r
#Test the k values that performed the best k = 2,3,4,5
#nstart=20 to ensure at least 20 random sets are chosen
cluster2 <- kmeans(data2, centers = 2, nstart=20)
cluster3 <- kmeans(data2, centers = 3, nstart=20)
cluster4 <- kmeans(data2, centers = 4, nstart=20)
cluster5 <- kmeans(data2, centers = 5, nstart=20)

#Compare the clusters with the different classes
table(cluster2$cluster, data$class)
```

```
##
##     Iris-setosa Iris-versicolor Iris-virginica
##   1           0              47             50
##   2          50               3              0
```

```r
table(cluster3$cluster, data$class)
```

```
##
##     Iris-setosa Iris-versicolor Iris-virginica
##   1          50               0              0
##   2           0              48             14
##   3           0               2             36
```

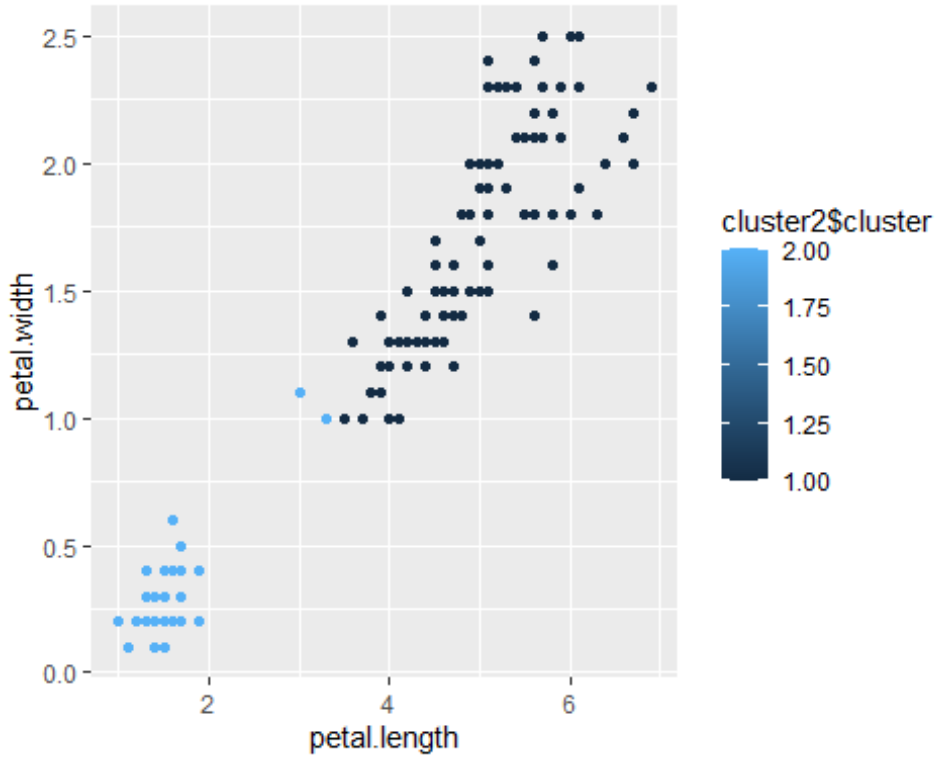
```r
#this model performed the best, all of species setosa was classified in one
cluster. There were some overlapping for the species versicolor and
virginica.

table(cluster4$cluster, data$class)
```

```
##
##     Iris-setosa Iris-versicolor Iris-virginica
##   1           0               0              32
##   2           0              27               1
##   3           0              23              17
##   4          50               0               0
```

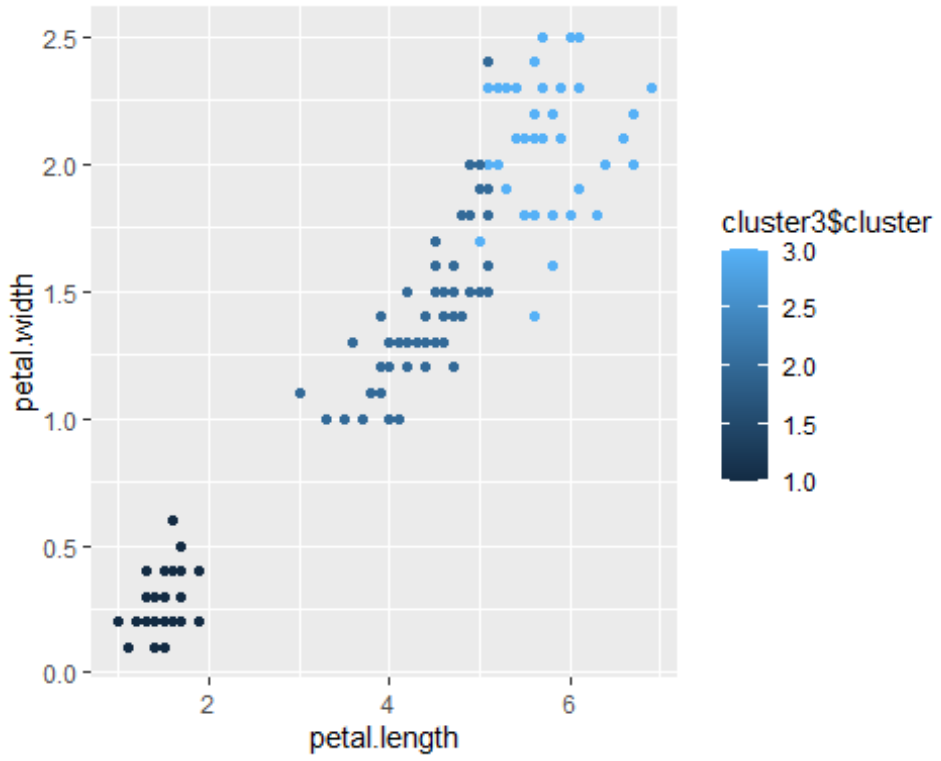
```
table(cluster5$cluster, data$class)
```

```
##
##     Iris-setosa Iris-versicolor Iris-virginica
##   1           0               0              12
##   2          50               0               0
##   3           0              26              13
##   4           0               0              24
##   5           0              24               1
```

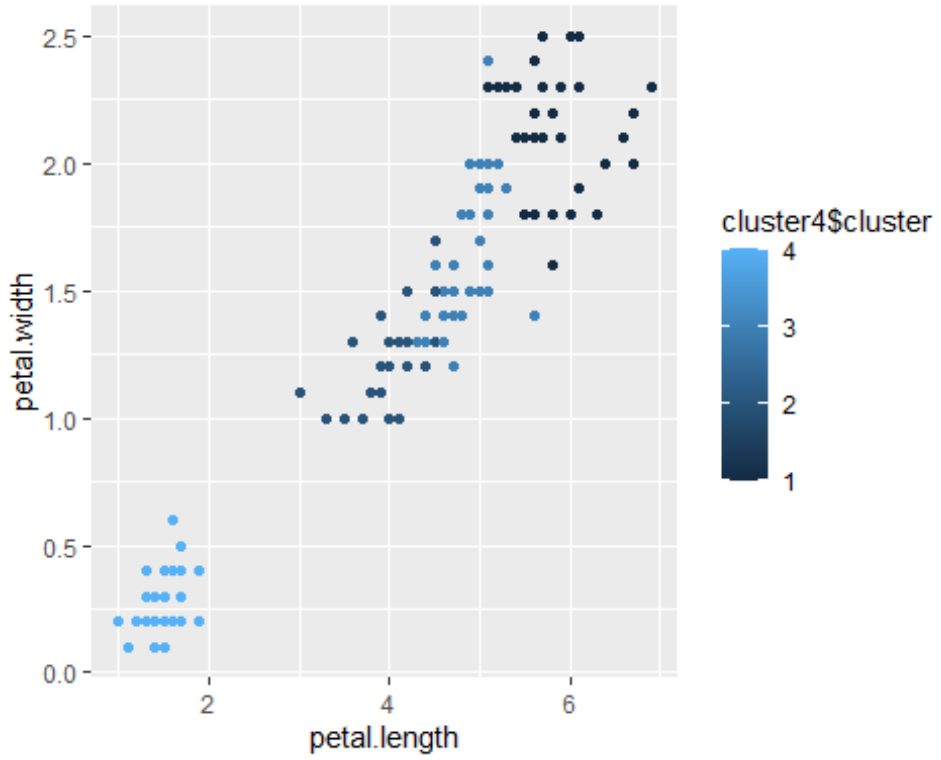
```
#Plot each k value
ggplot(data, aes(petal.length, petal.width, color = cluster2$cluster)) +
geom_point()
```



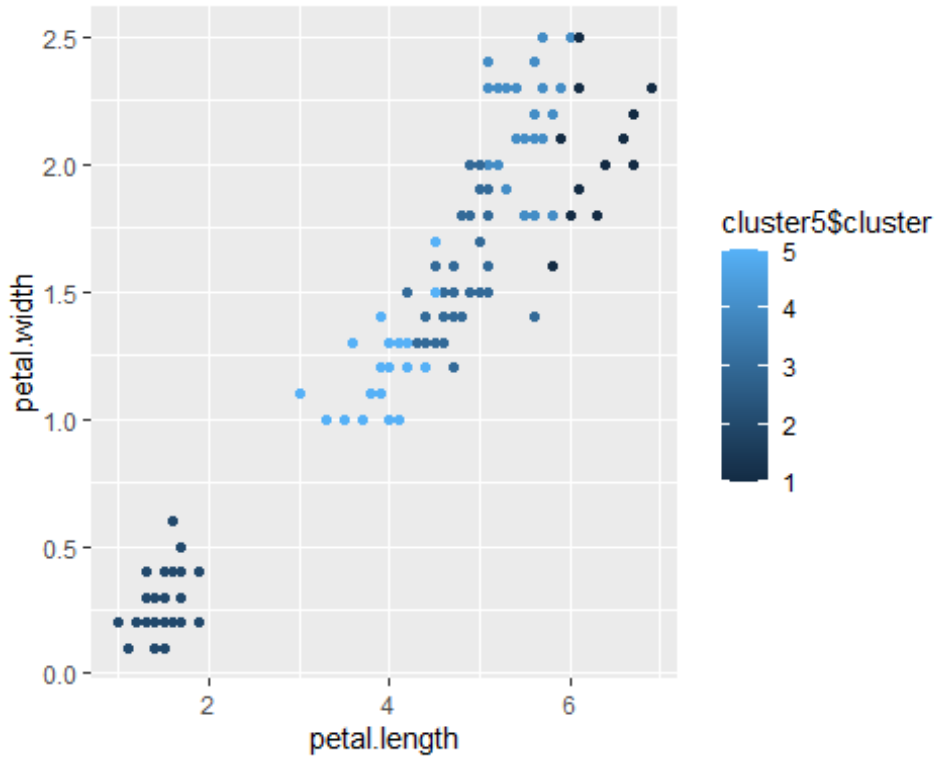

```
ggplot(data, aes(petal.length, petal.width, color = cluster3$cluster)) +
geom_point()
```



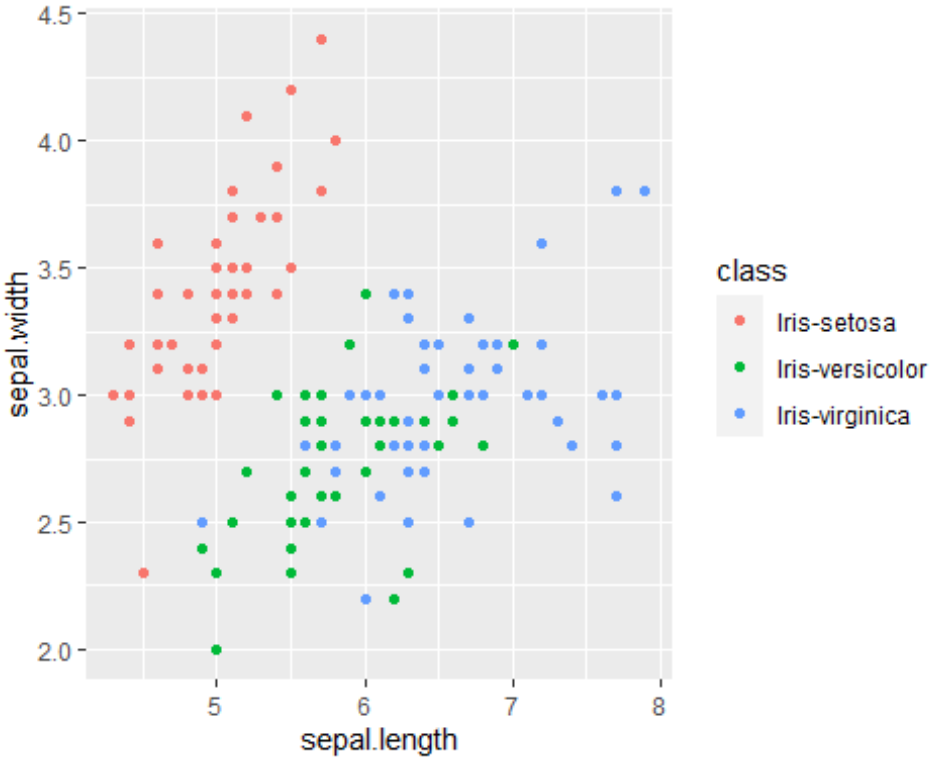

```
ggplot(data, aes(petal.length, petal.width, color = cluster4$cluster)) +
geom_point()
```



```
ggplot(data, aes(petal.length, petal.width, color = cluster5$cluster)) +
geom_point()
```




```
#the graph shows that k=3 is the ideal number of clusters. Model 3 performed
the best and we know that the actual number of classes, which is 3.

#Scale data
data_scaled <- scale(data2)
data_scaled <- data.frame(data_scaled, class = data$class)
ggplot(data, aes(sepal.length, sepal.width, color = class)) + geom_point()
```


4.5
4.0
3.5
3.0
2.5
2.0
sepal.width
5
6
7
8
sepal.length
class
Iris-setosa
Iris-versicolor
Iris-virginica