Week2_Homework

Suman Thakuri

2022-09-06

==========

Question 3.1 (a)

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)

==========

```
# ------------------- Code for Question 3.1 part (a) ----------------------

# Clear environment
rm(list = ls())

# First, load the kknn library (which contains the kknn function) and read in
the data
library(kknn)
```

```
## Warning: package 'kknn' was built under R version 4.2.1
```

```
# Import Dataset
data <- read.table("C:/Users/Suman/OneDrive - Georgia Institute of
Technology/Georgia Institute of Technology/ISYE
6501/Week2/credit_card_data.txt", header = FALSE, sep = "")


# Check to make sure the data is read correctly
head(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
```

```
# random number generator: to generate random sampling/selection events
set.seed(1234)
```

```
model <- train.kknn(V11~., data, kmax =  20, scale = TRUE)


predicted <- round(fitted(model)[[20]][1:nrow(data)])
predicted
```

```
##   [1] 1 1 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1
1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0
1 1 0
##  [75] 1 1 0 1 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1
1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
1 1 1
## [186] 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0
1 0 1
## [223] 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 1
## [260] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0
## [297] 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0
## [334] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
## [371] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
## [408] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
## [445] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1
1 1 1
## [482] 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 1 0 1 1 1 0 0 0 1 0 1
0 1 0
## [519] 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1
1 0 0
## [556] 1 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0
## [593] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0
## [630] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
```

```
accuracy = sum(predicted == data[,11]) / nrow(data)

accuracy
```

```
## [1] 0.8501529
```

```
=====================================================================
```

Using kknn model cross-validation wiht kamx= 20, I got he model accuracy of 0.8501529

==========

Question 3.1 (b)

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

==========

```
# -------------------- Code for Question 3.1 part (b) ----------------------

# Clear environment
rm(list = ls())

# load the kernlab library and read in the data
library(kernlab)

# Import Data set
data <- read.table("C:/Users/Suman/OneDrive - Georgia Institute of
Technology/Georgia Institute of Technology/ISYE
6501/Week2/credit_card_data.txt", header = FALSE, sep = "")


# Check to make sure the data is read correctly
head(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

# random number generator: to generate random sampling/selection events
set.seed(1234)


# splitting data into 60% training, 20% validation, and 20% test.

# This method did not exactly split data into 60% 20% 20% => (382,137,135)

# split_data <- sample(3, nrow(data), replace = TRUE, prob = c(0.6,0.2,0.2))
#
# train_data = data[split_data==1,]
# valid_data = data[split_data==2,]
```

```r
# test_data = data[split_data==3,]

# Exploring alternative method for splitting data: referenced from youtube
video=>
https://www.youtube.com/watch?v=_PCebXBhnio&ab_channel=Dr.BharatendraRai

train_rows <- sample(rownames(data), dim(data)[1]*.6)

valid_rows <- sample(setdiff(rownames(data),train_rows), dim(data)[1]*.2)

test_rows <- setdiff(rownames(data), union(train_rows,valid_rows))

train_data <- data[train_rows,]
valid_data <- data[valid_rows,]
test_data <- data[test_rows,]

model <- ksvm(as.matrix(train_data[,1:10]),
              as.factor(train_data[,11]),
              type = "C-svc",
              kernel = "vanilladot",
              C = 10,
              scaled = TRUE)

##  Setting default kernel parameters

model

## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 10
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 115
##
## Objective Function Value : -1065.752
## Training error : 0.135204

pred <- predict(model, valid_data[,1:10])
accuracy <- sum(pred == valid_data[,11]) / nrow(valid_data)
accuracy

## [1] 0.8846154

pred <- predict(model, test_data[,1:10])
accuracy <- sum(pred == test_data[,11]) / nrow(test_data)
accuracy

## [1] 0.8409091
```

==========

Splitting our data set to 60% training, 20% validation and 20% test, for our trained model, our model accuracy is 88.46% on our validation data set and 84.09% accuracy on our test data set

==========

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.

==========

```
# -------------------- Code for Question 4.1  ----------------------
# No Code Require
```

==========

I am currently working in Energy Storage Industry. As part of my daily job, I have to troubleshoot system alarms/faults on our deployed systems which includes mainly our equipment and OEM equipment. We categorized the alarms/faults encountered into 5 categories:

1. Company Hardware
2. Company Software
3. OEM Hardware
4. OEM Software
5. Client Equipment
6. Site

A Clustering model would be appropriate to identify a Site with the max number of Alarms/faults that would prevent a Site from Operation.

==========

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.

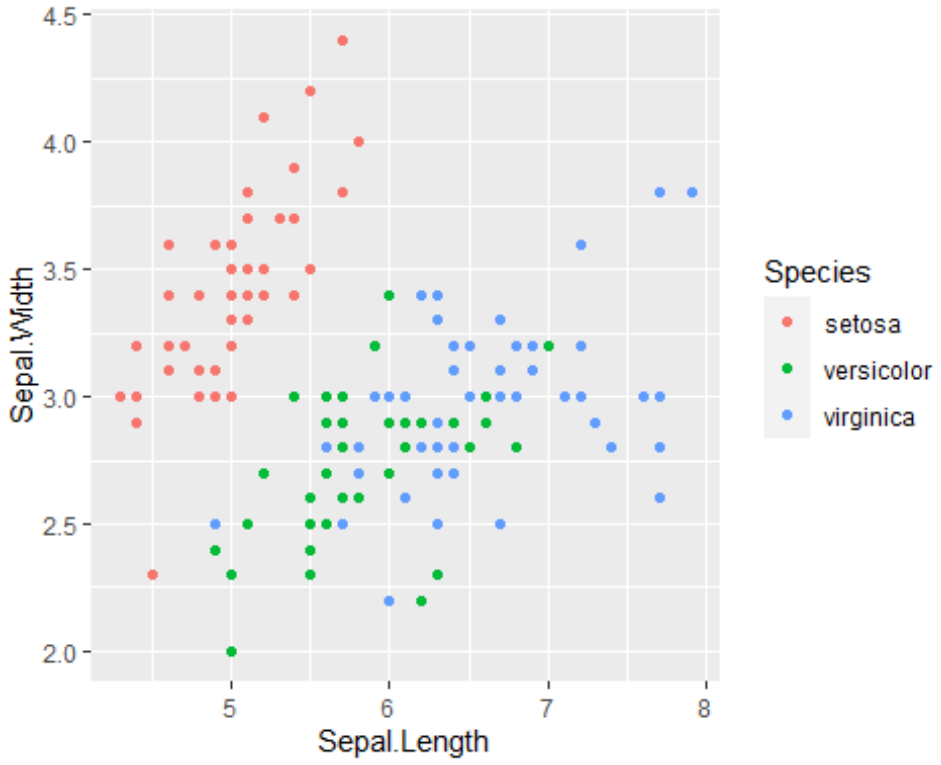
Use the R function kmeans to cluster the points as well as possible. Report the best combination of predictors, your suggested value of k, and how well your best clustering predicts flower type.

==========

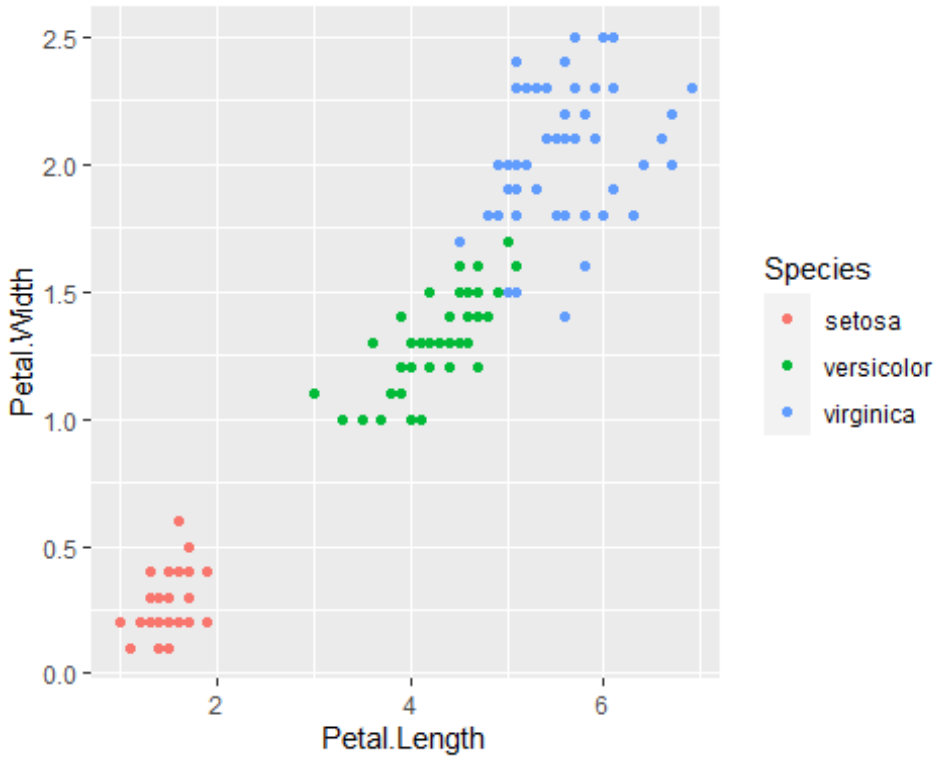
```
# -------------------- Code for Question 4.2 ----------------------------

# Clear environment
rm(list = ls())

# load the kernlab library and read in the data
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.2.1

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:kernlab':
##
##     alpha

library(cluster)
library(factoextra) # I needed to install factoextra package to get
fviz_cluster function

## Warning: package 'factoextra' was built under R version 4.2.1

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

# Import Data set
data <- read.table("C:/Users/Suman/OneDrive - Georgia Institute of
Technology/Georgia Institute of Technology/ISYE 6501/Week2/iris.txt", header
= TRUE, sep = "")


# Check to make sure the data is read correctly
head(data)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
```

```
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa

table(data$Species)

##
##     setosa versicolor  virginica
##         50         50         50
```

```
# plot to see data
ggplot(data, aes(Sepal.Length, Sepal.Width, color = Species)) + geom_point()
```

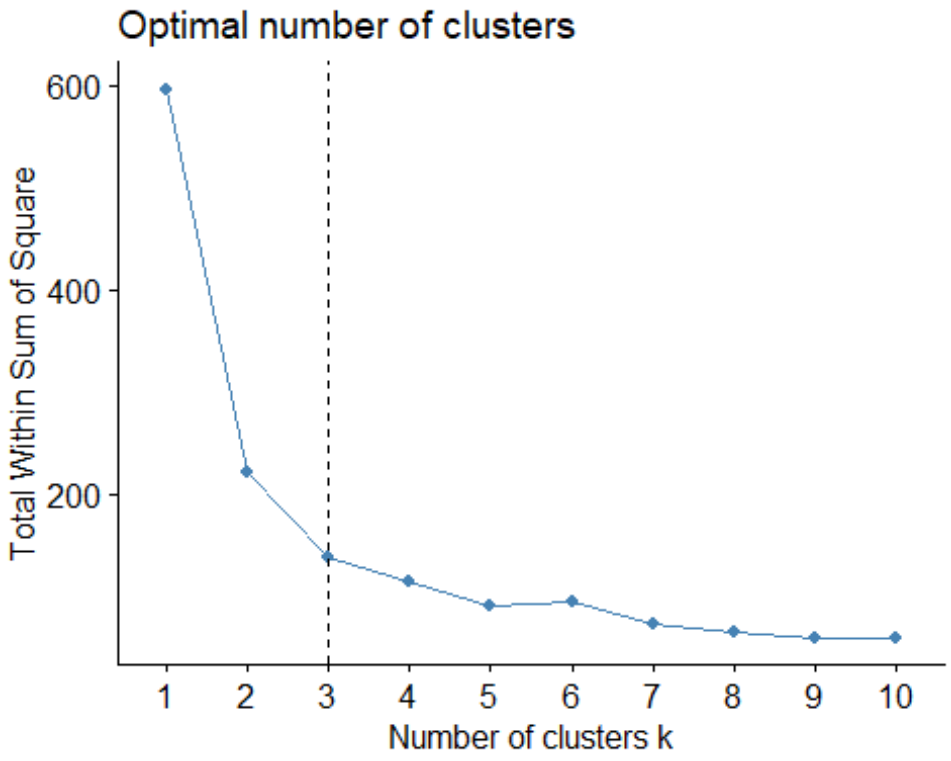


```
ggplot(data, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()
```




```
# Scale data
Scaled_data <- scale(data[,1:4])
```

```r
# calculate optimal cluster number based on the elbow method:
fviz_nbclust(Scaled_data, kmeans, method = c("wss")) +
  geom_vline(xintercept = 3, linetype = 2)
```




```r
# from elbow method optimal k value seems to be 3 but for the visualization
purpose, I am checking clustering for 2 and 3
# clustering
cluster1 <- kmeans(data[,1:4], centers = 2, nstart = 5)
cluster2 <- kmeans(data[,1:4], centers = 3, nstart = 10)


print(cluster1)
```

```
## K-means clustering with 2 clusters of sizes 53, 97
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     5.005660    3.369811     1.560377    0.290566
## 2     6.301031    2.886598     4.958763    1.695876
##
```

```
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
19  20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
39  40
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58
59  60
##   1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2   1
2   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78
79  80
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98
99 100
##   2   2   2   2   2   2   2   2   2   2   2   2   2   1   2   2   2   2
1   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118
119 120
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138
139 140
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
## 141 142 143 144 145 146 147 148 149 150
##   2   2   2   2   2   2   2   2   2   2
##
## Within cluster sum of squares by cluster:
## [1]  28.55208 123.79588
##  (between_SS / total_SS =  77.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"

print(cluster2)

## K-means clustering with 3 clusters of sizes 38, 50, 62
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     6.850000    3.073684     5.742105    2.071053
## 2     5.006000    3.428000     1.462000    0.246000
```

```
## 3     5.901613    2.748387     4.393548    1.433871
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
19  20
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
39  40
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58
59  60
##   2   2   2   2   2   2   2   2   2   2   3   3   1   3   3   3   3   3
3   3
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78
79  80
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   1
3   3
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98
99 100
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
3   3
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118
119 120
##   1   3   1   1   1   1   3   1   1   1   1   1   1   3   3   1   1   1
1   3
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138
139 140
##   1   3   1   3   1   1   3   3   1   1   1   1   1   3   1   1   1   1
3   1
## 141 142 143 144 145 146 147 148 149 150
##   1   1   3   1   1   1   3   1   1   3
##
## Within cluster sum of squares by cluster:
## [1] 23.87947 15.15100 39.82097
##  (between_SS / total_SS =  88.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# total within sum of squares : good metrics to make elbow diagram
cluster1$tot.withinss
```

```
## [1] 152.348
```

```r
cluster2$tot.withinss
```

```
## [1] 78.85144
```

```r
# Comparing our clustering model to data's true labels of species.
table(cluster1$cluster, data$Species)
```

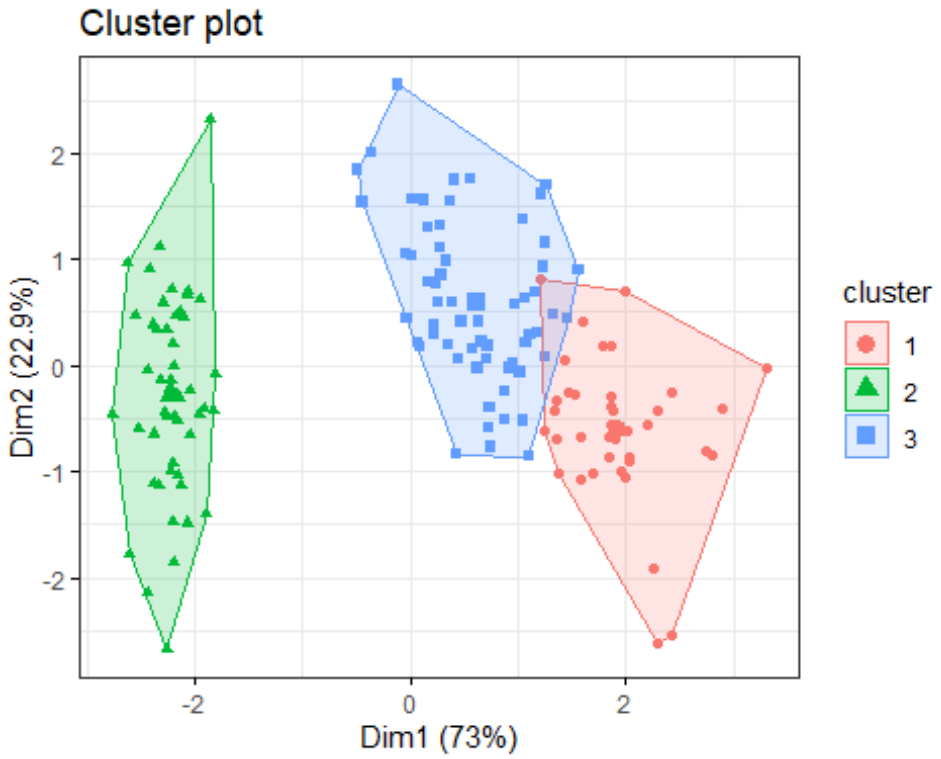
```
##
##     setosa versicolor virginica
##   1     50          3         0
##   2      0         47        50
```

```r
table(cluster2$cluster, data$Species)
```

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

```r
# Visualizing the clustering model
fviz_cluster(cluster1, data = data[,1:4],
             paleatte = c("#2E9FDF","#00AFBF","#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw())
```

```
fviz_cluster(cluster2, data = data[,1:4],
             paleatte = c("#2E9FDF","#00AFBF","#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw())
```



==========

I found 3 as the optimal k value for this data set from the elbow method. centers in the kmeans model denote the K value. The total within sum of squares for the Cluster1 model is 152.348 and for the Cluster2 model is 78.85144. As we can see, Cluster2 model has smaller within the sum of squares than Cluster1 model, hence a more compact and better clustering model.

==========