DSC 520: 11.2 Exercises

Rahul Rajeev

2023-02-23

Introduction to Machine Learning

First I read the files into dataframes.

```
bi_data <- read.csv('data/binary-classifier-data.csv')
tri_data <- read.csv('data/trinary-classifier-data.csv')
head(bi_data)
```

```
##   label        x        y
## 1     0 70.88469 83.17702
## 2     0 74.97176 87.92922
## 3     0 73.78333 92.20325
## 4     0 66.40747 81.10617
## 5     0 69.07399 84.53739
## 6     0 72.23616 86.38403
```

```
head(tri_data)
```

```
##   label        x        y
## 1     0 30.08387 39.63094
## 2     0 31.27613 51.77511
## 3     0 34.12138 49.27575
## 4     0 32.58222 41.23300
## 5     0 34.65069 45.47956
## 6     0 33.80513 44.24656
```

Then I plotted the data sets on a scatter plot.

Binary Data

As an example as to how I fit the k nearest neighbors to the binary data, here is my code:

```
split <- sample.split(bi_data, SplitRatio = 0.7)
bi_train <- subset(bi_data, split == "TRUE")
bi_test <- subset(bi_data, split == "FALSE")

# creating list of k values and initializing accuracy list
k_val <- c(3, 5, 10, 15, 20, 25)
acc_list <- c()

# iterating through each k value for the model, calculating accuracy,
# and storing it into a list
# printing the accuracy each iteration as well
```

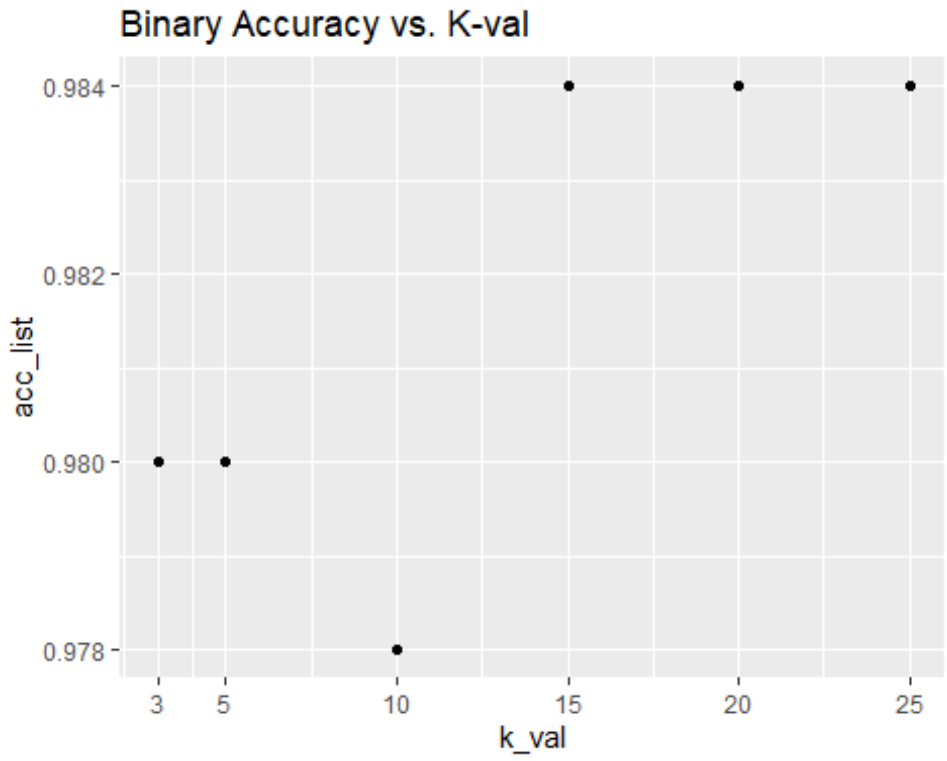
```r
for (k in k_val){
  k_class <- knn(train = bi_train,
                 test = bi_test,
                 cl = bi_train$label,
                 k = k)
  k_error <- mean(k_class != bi_test$label)
  accuracy <- 1-k_error
  resp <- paste('The accuracy for k =', k, 'was', accuracy)
  print(resp)
  acc_list <- append(acc_list, accuracy)
}
```

```
## [1] "The accuracy for k = 3 was 0.98"
## [1] "The accuracy for k = 5 was 0.98"
## [1] "The accuracy for k = 10 was 0.978"
## [1] "The accuracy for k = 15 was 0.984"
## [1] "The accuracy for k = 20 was 0.984"
## [1] "The accuracy for k = 25 was 0.984"
```

```r
# storing the values and accuracy into a data frame
results_df <- data.frame(k_val, acc_list)
head(results_df)
```

```
##   k_val acc_list
## 1     3    0.980
## 2     5    0.980
## 3    10    0.978
## 4    15    0.984
## 5    20    0.984
## 6    25    0.984
```

```r
# plotting results
ggplot(results_df, aes(x=k_val, y=acc_list)) + geom_point() +
  ggtitle('Binary Accuracy vs. K-val') +
  scale_x_continuous('k_val', labels = as.character(k_val), breaks = k_val)
```



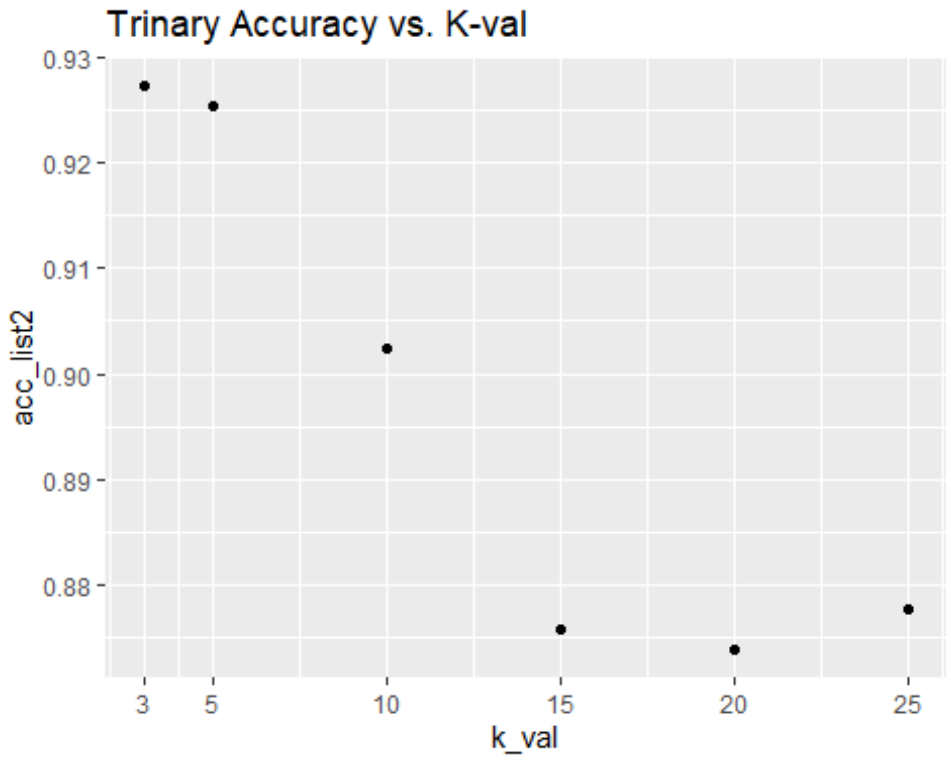
Thoughts

Looking back at the plots of the data, a linear classifier would not work here because the accuracy is way better than the one we used before. And compared to my previous accuracy of around a 57% in the 10.2 exercise, I would say using the KNN model has substantially improved accuracy by an increase of 40%.

Trinary Data

I repeated the procedure for trinary data. Here is the results of my analysis:

```
ggplot(results_df2, aes(x=k_val, y=acc_list2)) + geom_point() +
  ggtitle('Trinary Accuracy vs. K-val') +
  scale_x_continuous('k_val', labels = as.character(k_val), breaks = k_val)
```



Thoughts

From this particular run of knn model for the trinary data set, the accuracy seems to lower for each increasing k-value which is interesting, but the accuracy is still well above 90% which is good news.

Clustering

Introduction

I was having many issues with this part of the assignment, so I decided to work on as much as I could and ask questions about it to peers later.

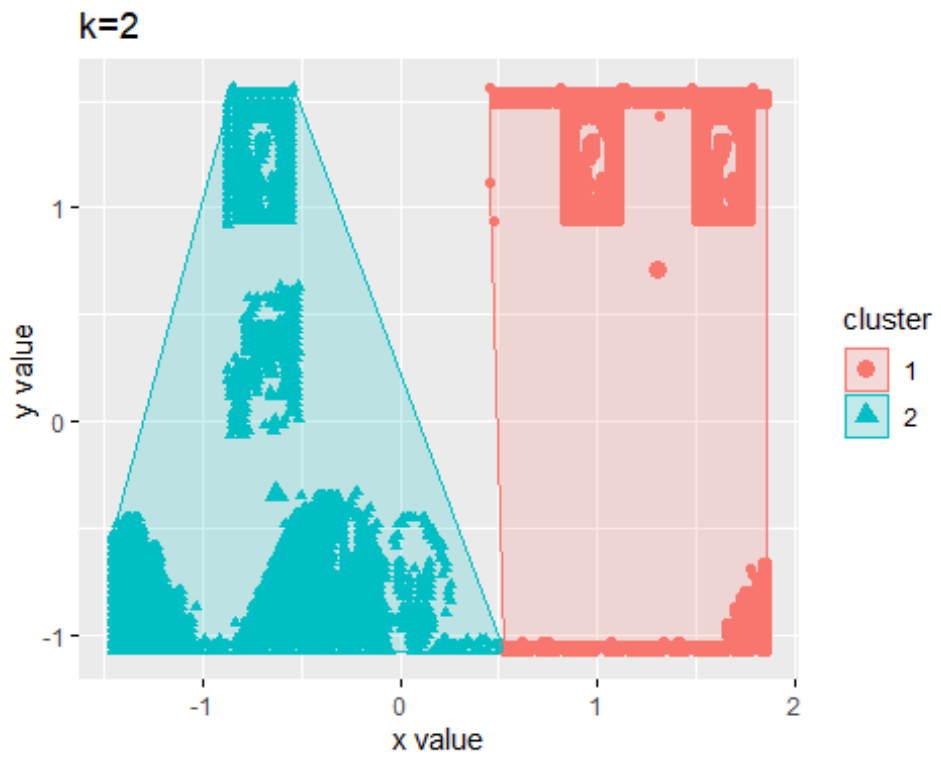
I was able to load the data and plot the data set, which looks awfully familiar to a level of Mario! I laughed out loud when I saw it, and the k means clustering visualizations looked amazing on the picture.

```
raw <- read.csv('data/clustering-data.csv')
ggplot(raw, aes(x=x, y=y)) + geom_point() + ggtitle('Clustering Data')
```

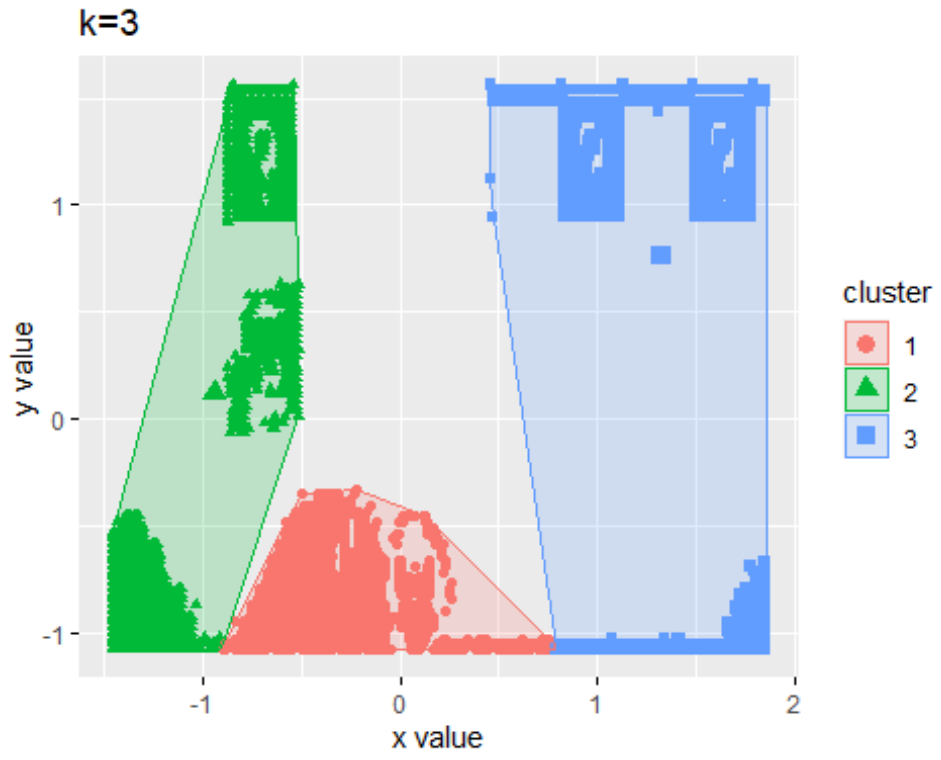
Fitting the Datasets using k means algorithm

Then I was also successful with plotting the visualizations of each kmeans clustering algorithm from 2 to 12, although iterating through a for loop didn't seem to work, so I had about 22 lines of code, each creating the kmeans objecting and then using a package called factoextra to plot using fviz_cluster(). Here are the results of the visualizations. It looks very beautiful to say the least. One of the coolest plots I've seen.

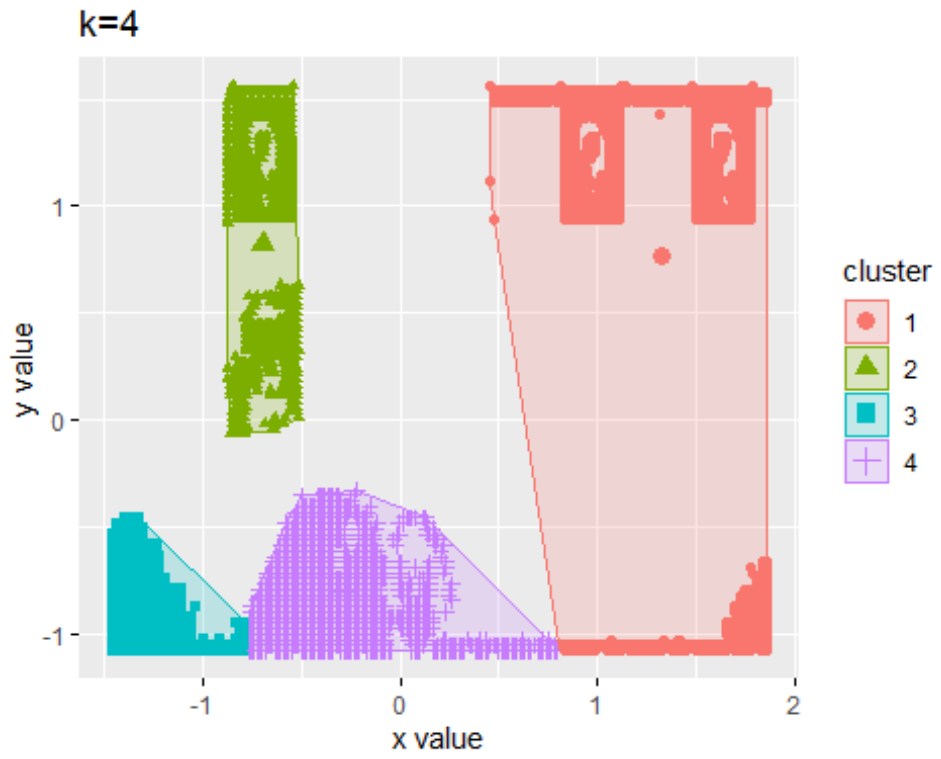
```
kmcluster2 <- kmeans(raw, 2, n=25)
fviz_cluster(kmcluster2, raw, geom = 'point') + ggtitle('k=2')
```



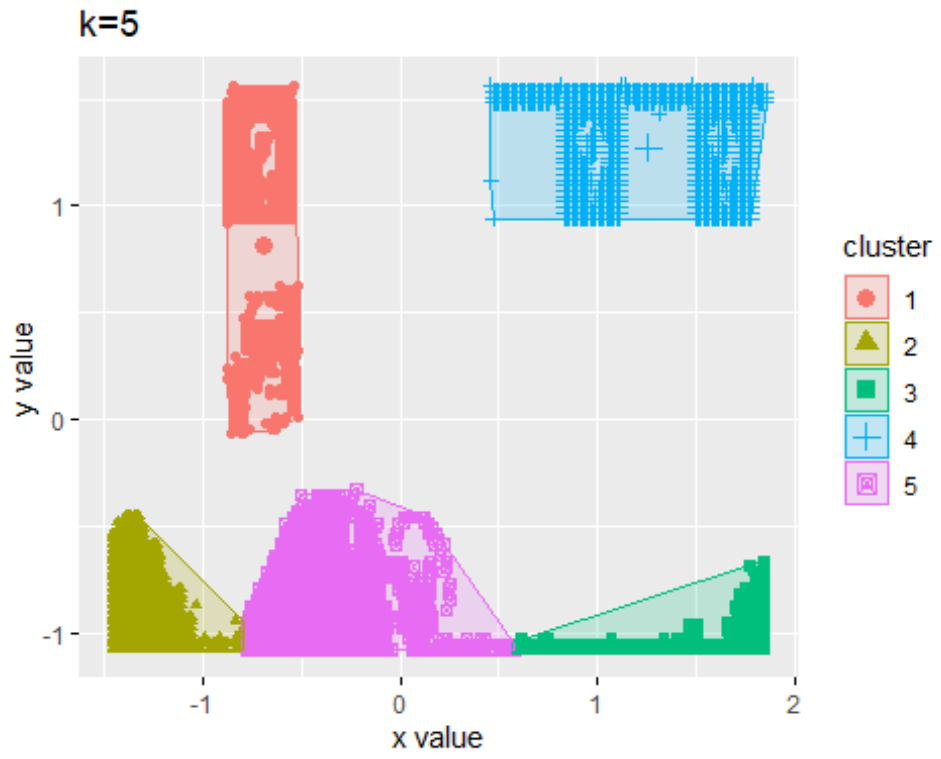

```
kmcluster3 <- kmeans(raw, 3, n=25)
fviz_cluster(kmcluster3, raw, geom = 'point') + ggtitle('k=3')
```



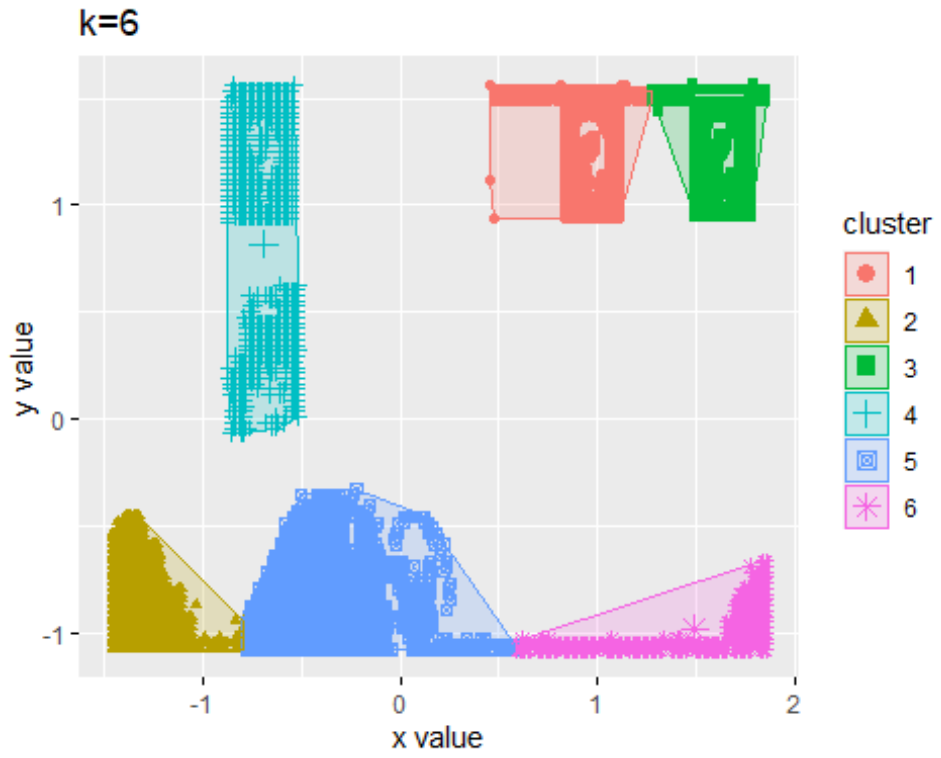
```
kmcluster4 <- kmeans(raw, 4, n=25)
fviz_cluster(kmcluster4, raw, geom = 'point') + ggtitle('k=4')
```



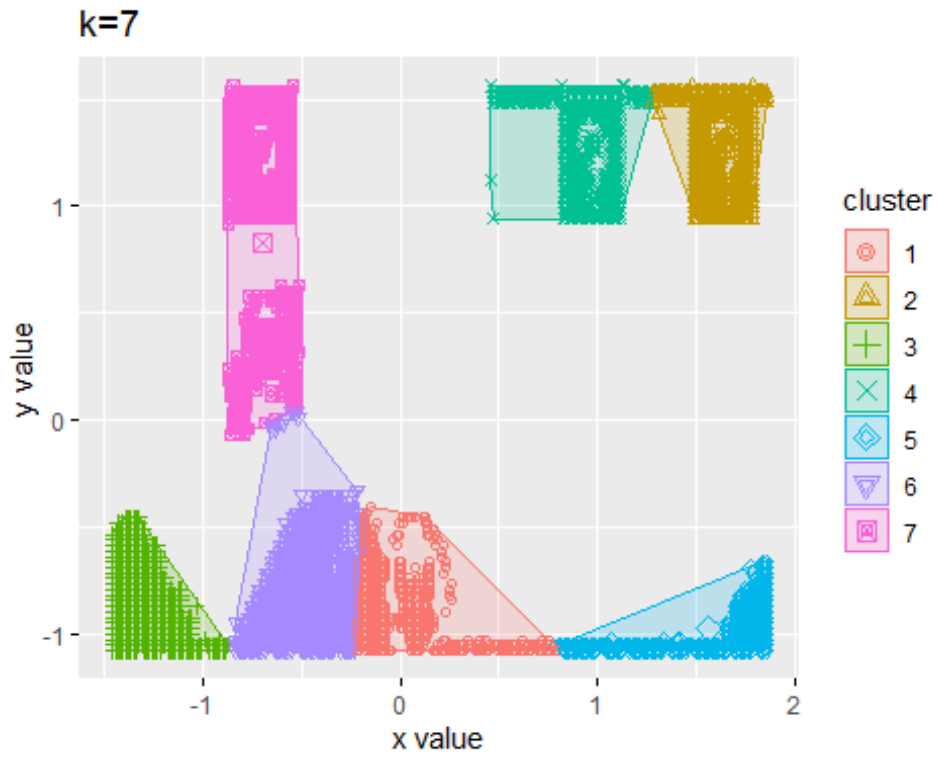

```
kmcluster5 <- kmeans(raw, 5, n=25)
fviz_cluster(kmcluster5, raw, geom = 'point') + ggtitle('k=5')
```



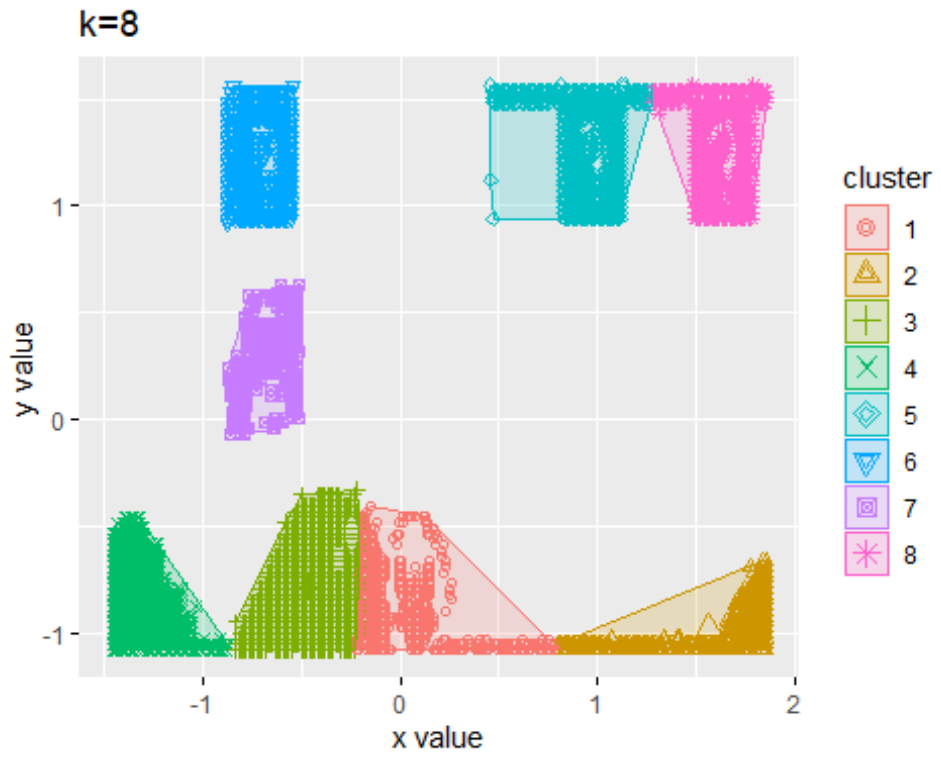

```
kmcluster6 <- kmeans(raw, 6, n=25)
fviz_cluster(kmcluster6, raw, geom = 'point') + ggtitle('k=6')
```



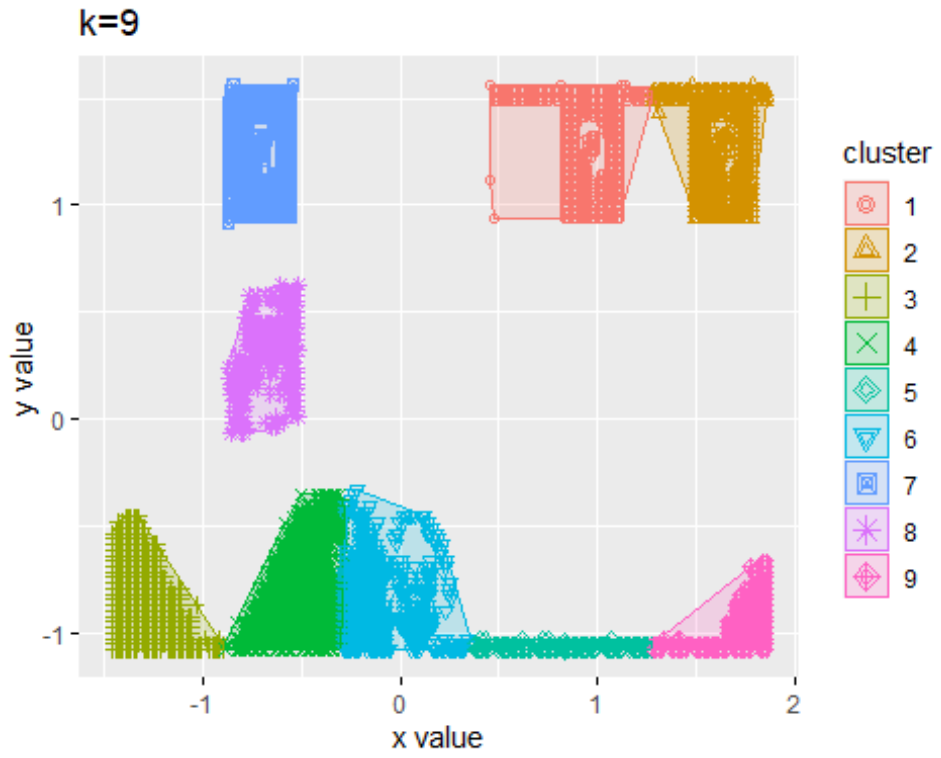
```
kmcluster7 <- kmeans(raw, 7, n=25)
fviz_cluster(kmcluster7, raw, geom = 'point') + ggtitle('k=7')
```



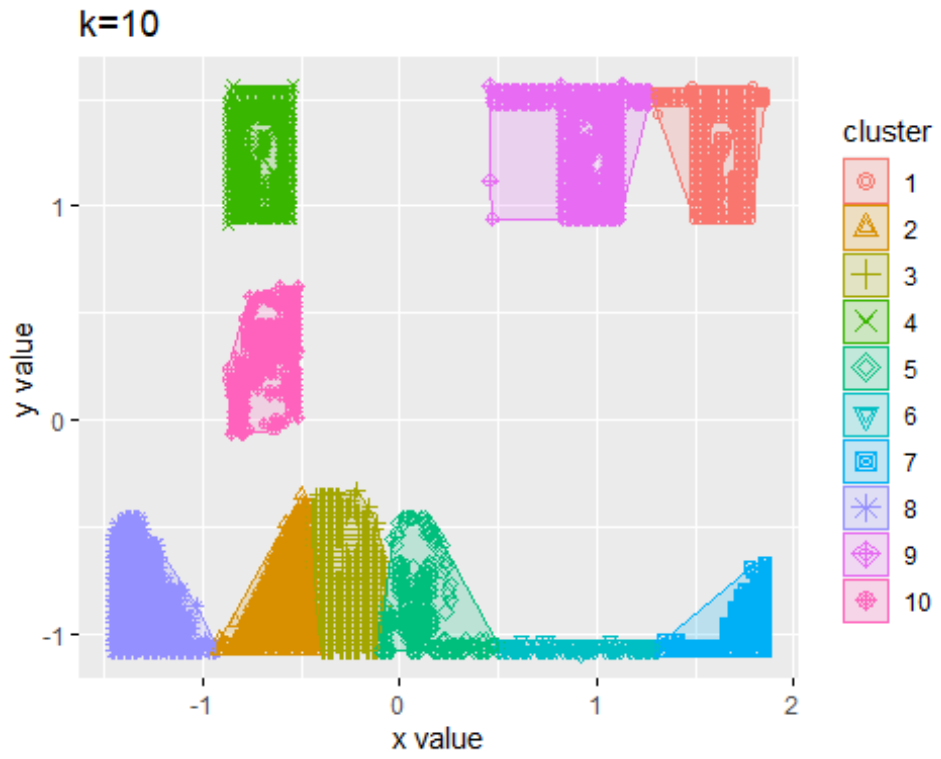

```
kmcluster8 <- kmeans(raw, 8, n=25)
fviz_cluster(kmcluster8, raw, geom = 'point') + ggtitle('k=8')
```



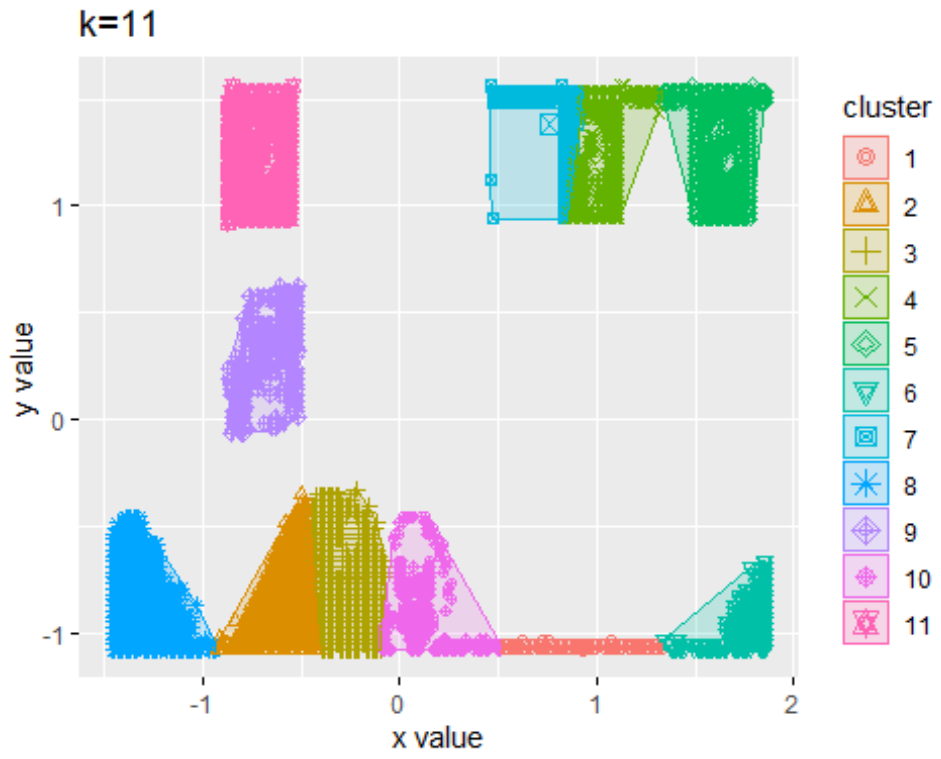

```
kmcluster9 <- kmeans(raw, 9, n=25)
fviz_cluster(kmcluster9, raw, geom = 'point') + ggtitle('k=9')
```



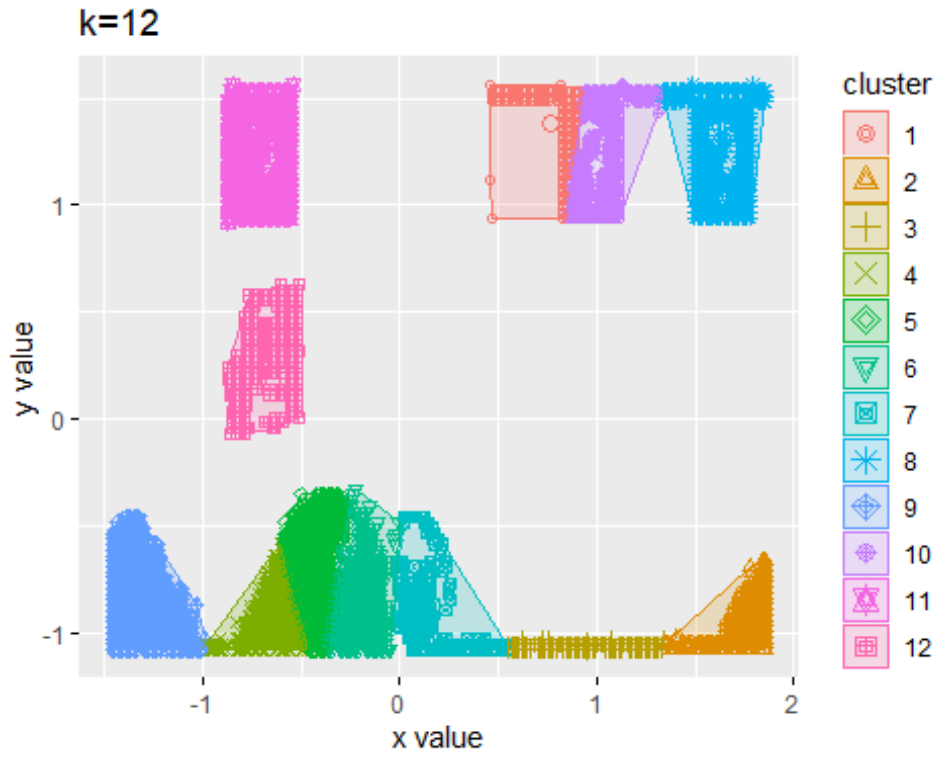
```
kmcluster10 <- kmeans(raw, 10, n=25)
fviz_cluster(kmcluster10, raw, geom = 'point') + ggtitle('k=10')
```




```
kmcluster11 <- kmeans(raw, 11, n=25)
fviz_cluster(kmcluster11, raw, geom = 'point') + ggtitle('k=11')
```




```
kmcluster12 <- kmeans(raw, 12, n=25)
fviz_cluster(kmcluster12, raw, geom = 'point') + ggtitle('k=12')
```



Calculating the average distance

It was here where I began to feel overwhelmed. Since I couldn't find a successful way to iterate through the kmeans objects, I couldn't also find a way to store the values of the results and also iterate through them to say the least at the moment. Since my knowledge of k means is growing, I wanted to instead include the procedure of which I will conduct the calculations once I get better. I managed to store my results of cluster label to the dataset in a merged dataset of which I will include the results of below.

```
labeled_k <- cbind(raw, k2 = kmcluster2$cluster, k3 = kmcluster3$cluster,
                   k4 = kmcluster4$cluster, k5 = kmcluster5$cluster,
                   k6 = kmcluster6$cluster, k7 = kmcluster7$cluster,
                   k8 = kmcluster8$cluster, k8 = kmcluster8$cluster,
                   k9 = kmcluster9$cluster, k10 = kmcluster10$cluster,
                   k11 = kmcluster11$cluster, k12 = kmcluster12$cluster)
head(labeled_k)
```

```
##     x   y k2 k3 k4 k5 k6 k7 k8 k8 k9 k10 k11 k12
## 1  46 236  2  2  2  1  4  7  6  6  7   4  11  11
## 2  69 236  2  2  2  1  4  7  6  6  7   4  11  11
## 3 144 236  1  3  1  4  1  4  5  5  1   9   7   1
## 4 171 236  1  3  1  4  1  4  5  5  1   9   7   1
## 5 194 236  1  3  1  4  1  4  5  5  1   9   4  10
## 6 195 236  1  3  1  4  1  4  5  5  1   9   4  10
```

1. Store the xy values for each k and the center matrix for each k using the cluster parameter on the kmeans object.
2. Then calculate the distance between each xy value and the center and average them.
3. Once I find the average distance for one, then repeat for the remaining cluster labels for that k-val. For example for k = 2, there are two clusters 1 and 2. I would have to average the distances for each point in cluster 1 and average the distances for each point in cluster 2, then average those two together, and that would be my result for k=2.
4. Repeat steps 1-3 for each kclusters in the data frame for each label
5. Plot the averaged distance for each k-val set onto a plot with k as the x and the averaged distance for y.

Elbow

Because the previous part was left incomplete, I will also leave my general thoughts about what the elbow means and how I would analyze it if I actually got to that point. The elbow signifies where the average distance between points levels off, so it would be optimal to use that particular number of clusters for future analysis.