Classification of Rice Dataset

Saurabh Shinde

11/18/2020

Background of data

- **The data in the dataset was extracted from rice seeds**
- **The physical properties of rice seeds are given in the dataset like:**
- Area of seed
- Major axis length
- Minor axis length
- Eccentricity
- Convex area
- Equivalent diameter
- Extent
- Perimeter
- Roundness
- **The dataset is available on kaggle:**

  https://www.kaggle.com/seymasa/rice-dataset-gonenjasmine?select=Rice-Gonen+andJasmine.csv (https://www.kaggle.com/seymasa/rice-dataset-gonenjasmine?select=Rice-Gonen+andJasmine.csv)
- **Based on the provided information we will classify the dataset into clusters with unsupervised learning by using k-means cluster analysis**

Data Importing

Import the packages necessary for analysis and read the data file

```
#Clear Workspace
rm(list = ls(all = TRUE))

#Load Libraries
library(tinytex)
library(tidyverse)
library(cluster)
library(factoextra)
library(gridExtra)
library(kableExtra)
library(grid)

#Read .csv files
df <- read_csv("data/project_dataset.csv")
#Omit all na in dataset
df_tidy <- na.omit(df)
#Dropping id and class columns
df_trim <- select(df_tidy, 2:11)
```

Scaling the data frame

```
#Scaling the data
df_scale <- scale(df_trim)
#Table showing original dataset
kableExtra::kable(head(df), caption = "Table 1: Original Dataset" ) %>%
  kable_styling()
```

Table 1: Original Dataset

id	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	AspectRation

1	4537	92.22932	64.01277	0.7199162	4677	76.00453	0.6575362	273.085	0.7645096	1.440796
2	2872	74.69188	51.40045	0.7255527	3015	60.47102	0.7130089	208.317	0.8316582	1.453137
3	3048	76.29316	52.04349	0.7312109	3132	62.29634	0.7591532	210.012	0.8684336	1.465950
4	3073	77.03363	51.92849	0.7386387	3157	62.55130	0.7835288	210.657	0.8702031	1.483456
5	3693	85.12478	56.37402	0.7492816	3802	68.57167	0.7693750	230.332	0.8747433	1.510000
6	2990	77.41707	50.95434	0.7528609	3080	61.70078	0.5848983	216.930	0.7984391	1.519342

```
#Table showing scaled dataset
kableExtra::kable(head(df_scale), caption = "Table 2: Scaled Dataset") %>%
  kable_styling()
```

Table 2: Scaled Dataset

Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	AspectRatio
-1.703584	-4.803612	0.4179152	-6.393762	-1.696942	-1.829999	0.3916432	-2.661705	0.8395653	-2.66372
-2.838400	-6.220618	-0.8355881	-6.209412	-2.803462	-3.398050	0.9230448	-4.857184	1.8371648	-2.63534
-2.718444	-6.091236	-0.7716784	-6.024354	-2.725566	-3.213790	1.3650842	-4.799728	2.3835218	-2.60587
-2.701405	-6.031408	-0.7831084	-5.781419	-2.708922	-3.188053	1.5985909	-4.777864	2.4098111	-2.56562
-2.278830	-5.377651	-0.3412790	-5.433330	-2.279496	-2.580319	1.4630043	-4.110929	2.4772625	-2.50457
-2.757975	-6.000426	-0.8799257	-5.316264	-2.760186	-3.273910	-0.3041931	-4.565224	1.3436426	-2.48309

K-means Cluster Analysis

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity.

Comparing k-means cluster plots for clusters in range of 1 to 6

```
#k-means clusters for k(1:6)
k1 <- kmeans(df_scale, centers = 1, nstart = 25)
k2 <- kmeans(df_scale, centers = 2, nstart = 25)
k3 <- kmeans(df_scale, centers = 3, nstart = 25)
k4 <- kmeans(df_scale, centers = 4, nstart = 25)
k5 <- kmeans(df_scale, centers = 5, nstart = 25)
k6 <- kmeans(df_scale, centers = 6, nstart = 25)

#Plots to compare
p1 <- fviz_cluster(k1, geom = "point",  data = df_scale) + ggtitle("k = 1")+ theme_bw()
p2 <- fviz_cluster(k2, geom = "point", data = df_scale) + ggtitle("k = 2") + theme_bw()
p3 <- fviz_cluster(k3, geom = "point",  data = df_scale) + ggtitle("k = 3")+ theme_bw()
p4 <- fviz_cluster(k4, geom = "point",  data = df_scale) + ggtitle("k = 4")+ theme_bw()
p5 <- fviz_cluster(k5, geom = "point",  data = df_scale) + ggtitle("k = 5")+ theme_bw()
p6 <- fviz_cluster(k6, geom = "point",  data = df_scale) + ggtitle("k = 6")+ theme_bw()
#Plotting a grid
grid.arrange(p1, p2, p3, p4, p5, p6, nrow = 3, bottom = textGrob("Figure 1: Plots with k value 1 to 6"))
```

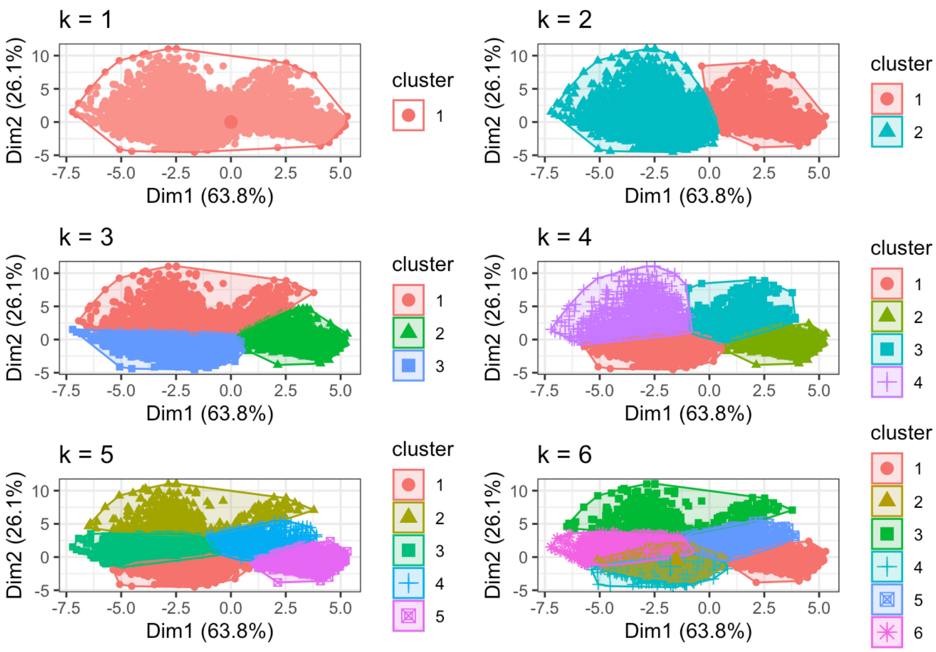

Figure 1: Plots with k value 1 to 6

Determining Optimal Clusters - Elbow Method

Using the elbow method to determine the optimal number of clusters for k-means clustering

Elbow method gives us an idea on what a good k number of clusters would be based on the sum of squared distance (SSE) between data points and their assigned clusters' centroids. We pick k at the spot where SSE starts to flatten out and forming an elbow. We'll use the geyser dataset and evaluate SSE for different values of k and see where the curve might form an elbow and flatten out.

```
set.seed(123)

#Function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(df_scale, k, nstart = 10 )$tot.withinss
}
#Compute and plot wss for k = 1 to k = 15
k.values <- 1:15
#Extract wss for 2-15 clusters
wss_values <- map_dbl(k.values, wss)
#Plotting the elbow bent diagram
plot(k.values, wss_values,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters K", sub="Figure 2: Elbow Bent",
       ylab="Total within-clusters sum of squares", caption = 'Figure 2: Elbow Method')
```

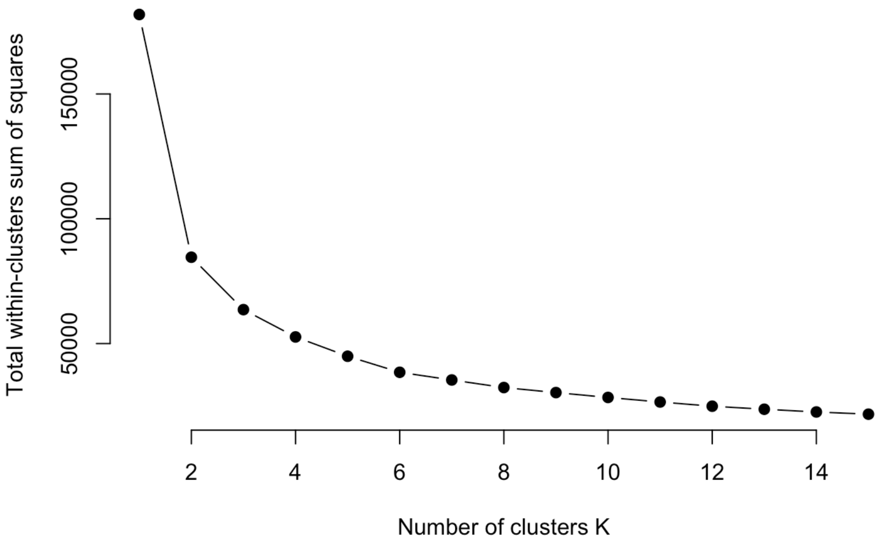

Figure 2: Elbow Bent

Result of Elbow Method

- The graph above shows that k=2 is not a bad choice and the original dataset has 2 classes Jasmine and Gonen, so it confirms that Elbow Method is accurate.
- Sometimes it's still hard to figure out a good number of clusters to use because the curve is monotonically decreasing and may not show any elbow or has an obvious point where the curve starts flattening out.

Final plot

Kmean clustering result

```
#k-means clustering with k = 2
k <- kmeans(df_scale, centers = 2, nstart = 25)
#Plotting the kmean cluster plot
fviz_cluster(k, data = df_scale, caption = 'Figure 3: Clusters plot')+ theme_bw()
```

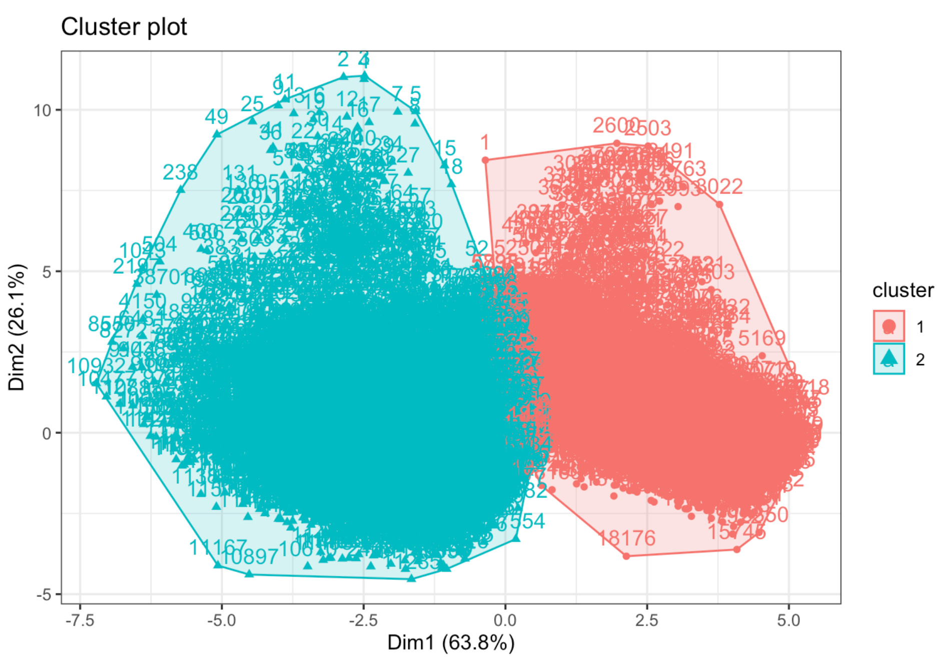

Figure 3: Clusters plot

```
#Table with cluster 1 and 2 with mean of each column (scaled dataset)
kableExtra::kable(k$centers, caption = "Table 3: Centroid of clusters for scaled dataset") %>%
  kable_styling()
```

Table 3: Centroid of clusters for scaled dataset

Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	AspectRat
0.9699032	0.2202755	1.0686118	-0.8983218	0.9674666	0.9595447	0.3484196	0.6567423	0.9399144	-0.9392
-0.7291325	-0.1655938	-0.8033375	0.6753206	-0.7273008	-0.7213455	-0.2619273	-0.4937113	-0.7065882	0.7061

```
#Table with cluster 1 and 2 with mean of each column (original dataset)
kableExtra::kable(df_trim %>%
  mutate(Cluster = k$cluster) %>%
  group_by(Cluster) %>%
  summarise_all("mean"), caption = "Table 4: Centroid of clusters for original dataset") %>%
  kable_styling()
```

Table 4: Centroid of clusters for original dataset

Cluster	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	Aspe
1	8459.532	154.4070	70.55985	0.8879400	8678.959	103.63844	0.6530242	370.9813	0.7712641	
2	5966.712	149.6313	51.72495	0.9360544	6133.407	86.98712	0.5893104	337.0422	0.6604378	

Result of K-means Analysis

- **We have classsified the given dataset with the help of unsupervised learning. The 10 parameters have been clustered into 2 clusters. The cluster 1 has bigger seed size and the cluster 2 has smaller seed size. We can observe the mean values of parameters for each parameter in table 4.**
- **Kmeans algorithm is good in capturing structure of the data if clusters have a spherical-like shape. It doesn’t work well when clusters are in different shapes such as elliptical clusters.**