CS5542 Big Data Apps and Analytics

In Class Programming –7 Report
(Jongkook Son)

Project Overview:

Use a different data and use the model provided in ICP7 to perform clustering. You must try 5 different number of clusters (for example n_clusters= 5 or n_clusters=6,7,8,or 9 etc) based on elbow curve and for each cluster visualize the clustering results and report your findings in detail.

Requirements/Task(s):

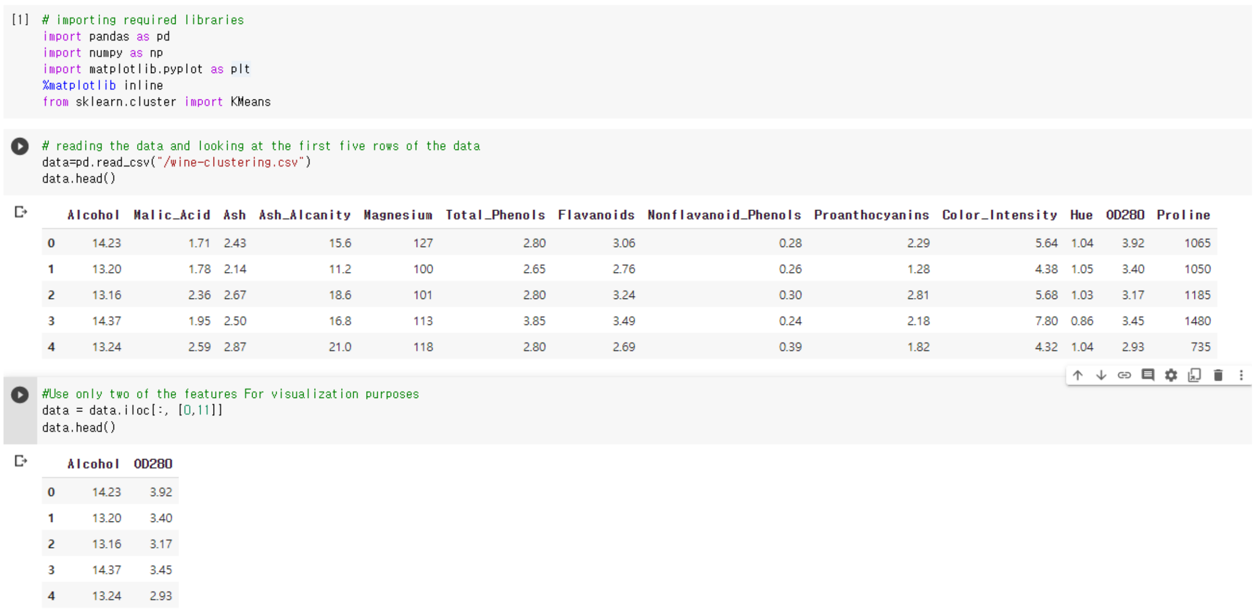
1) Successfully executing the code and trying 5 different number of clusters based on elbow curve and visualizing those clusters using python plotting libraries (75 points)
2) Using a new and good dataset (5 points)
3) overall code quality (10 points)
4) Pdf Report quality, video explanation (10 points)

What I learned in ICP:

I could have learned how to cluster data with K means algorithm. It was a little different kind of task because it was the unsupervised learning. All of the task before were supervised learning which have labels for the data. However it seems more easier than before. Thanks to this ICP7 I could get some insight how to get the optimized number of clustering group by using elbow method. It was basically draw a elbow graph adjusting cluster group number then find the point that is leveled off. And the start point of the leveled off is likely to be an optimized point. By doing this process the result of the clustering could get better.

ICP description what was the task you were performing and Screen shots that shows the successful execution of each required step of your code

Use a new data set which contains wine informations.



```
[1] # importing required libraries
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    %matplotlib inline
    from sklearn.cluster import KMeans
```

```
# reading the data and looking at the first five rows of the data
data=pd.read_csv("/wine-clustering.csv")
data.head()
```

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensity	Hue	OD280	Proline
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

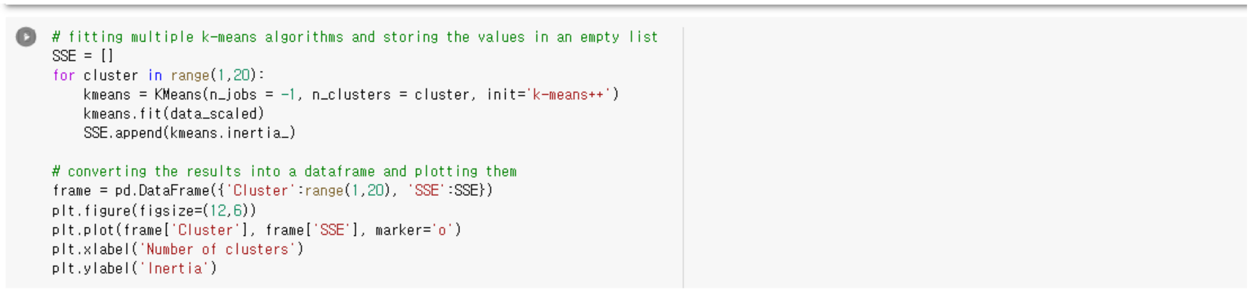
```
#Use only two of the features For visualization purposes
data = data.iloc[:, [0,11]]
data.head()
```

	Alcohol	OD280
0	14.23	3.92
1	13.20	3.40
2	13.16	3.17
3	14.37	3.45
4	13.24	2.93

⇨ **For visualization purposes, I chose to include only two of the features. Two features that I decided to include is Alcohol and OD280.**

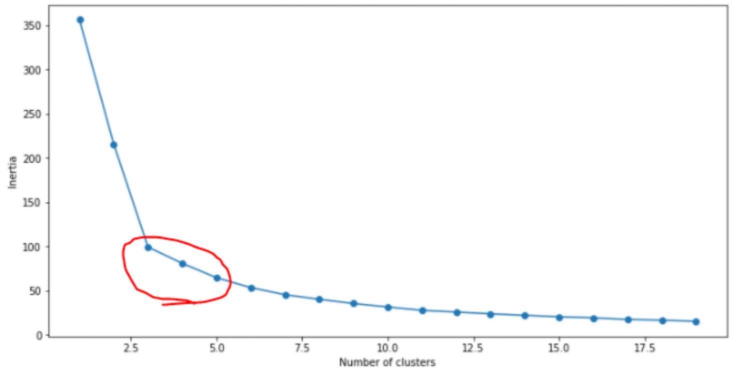
Trying 5 different number of clusters based on elbow curve and visualizing those clusters using python plotting libraries

1. Graph a Elbow Curve



```
# fitting multiple k-means algorithms and storing the values in an empty list
SSE = []
for cluster in range(1,20):
    kmeans = KMeans(n_jobs = -1, n_clusters = cluster, init='k-means++')
    kmeans.fit(data_scaled)
    SSE.append(kmeans.inertia_)

# converting the results into a dataframe and plotting them
frame = pd.DataFrame({'Cluster':range(1,20), 'SSE':SSE})
plt.figure(figsize=(12,6))
plt.plot(frame['Cluster'], frame['SSE'], marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
```

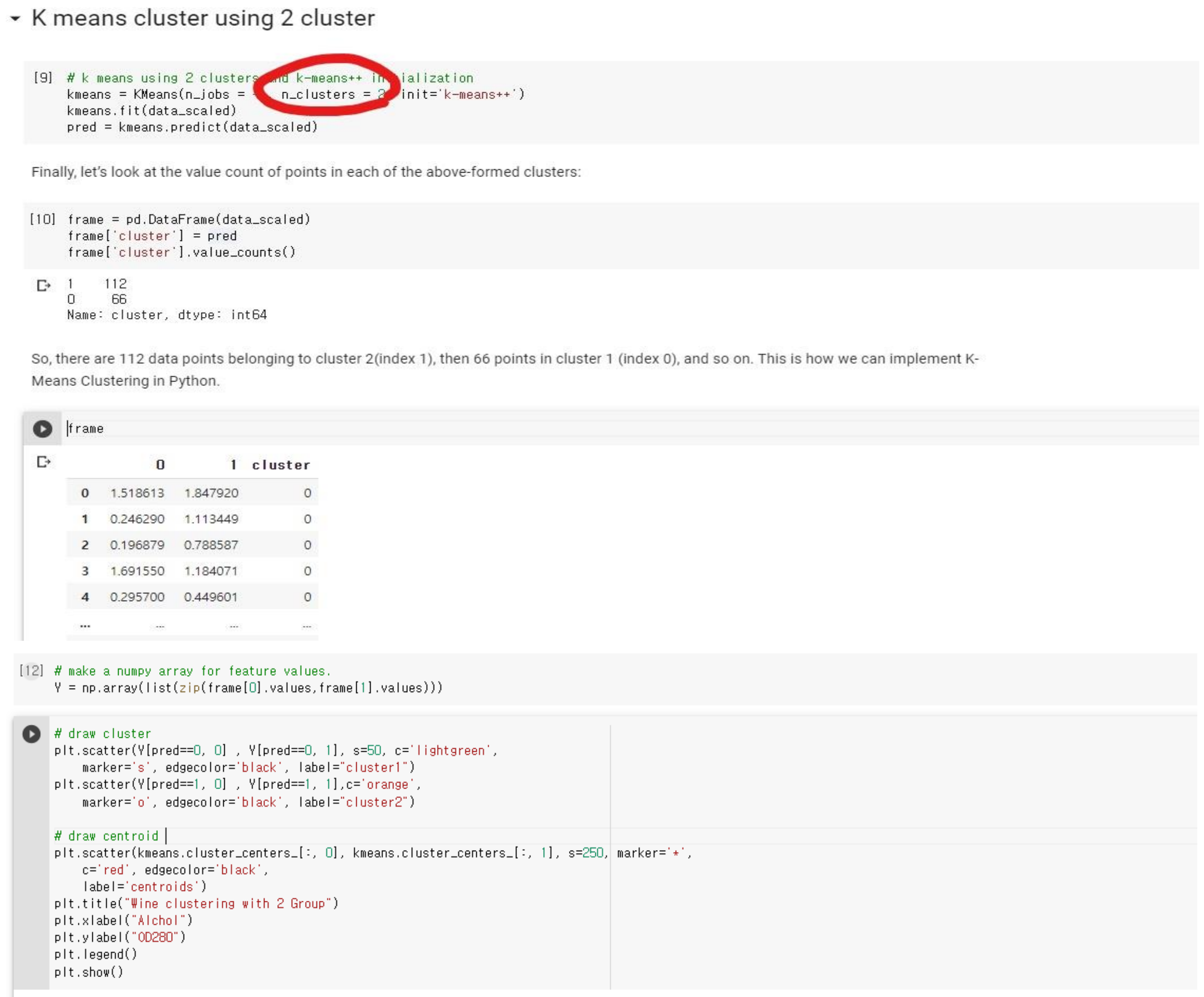
```
Text(0, 0.5, 'Inertia')
```



⇨ **We graph the relationship between the number of clusters and Inertia then we select the number of clusters where the change in Inertia begins to level off (elbow method). In this case, Inertial value get leveled off after 2.5. So we can guess we need to start K with 2 or 3.**

2. Executing the code and trying 5 different number of clusters based on elbow curve and visualizing those clusters

1) k means using 2 clusters and k-means++ initialization



K means cluster using 2 cluster

```
[9]  # k means using 2 clusters and k-means++ initialization
     kmeans = KMeans(n_jobs = -1, n_clusters = 2, init='k-means++')
     kmeans.fit(data_scaled)
     pred = kmeans.predict(data_scaled)
```

Finally, let's look at the value count of points in each of the above-formed clusters:

```
[10] frame = pd.DataFrame(data_scaled)
     frame['cluster'] = pred
     frame['cluster'].value_counts()
```

```
1    112
0     66
Name: cluster, dtype: int64
```

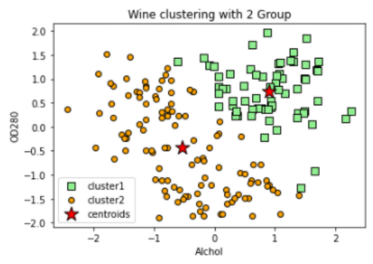
So, there are 112 data points belonging to cluster 2(index 1), then 66 points in cluster 1 (index 0), and so on. This is how we can implement K-Means Clustering in Python.

```
frame
```

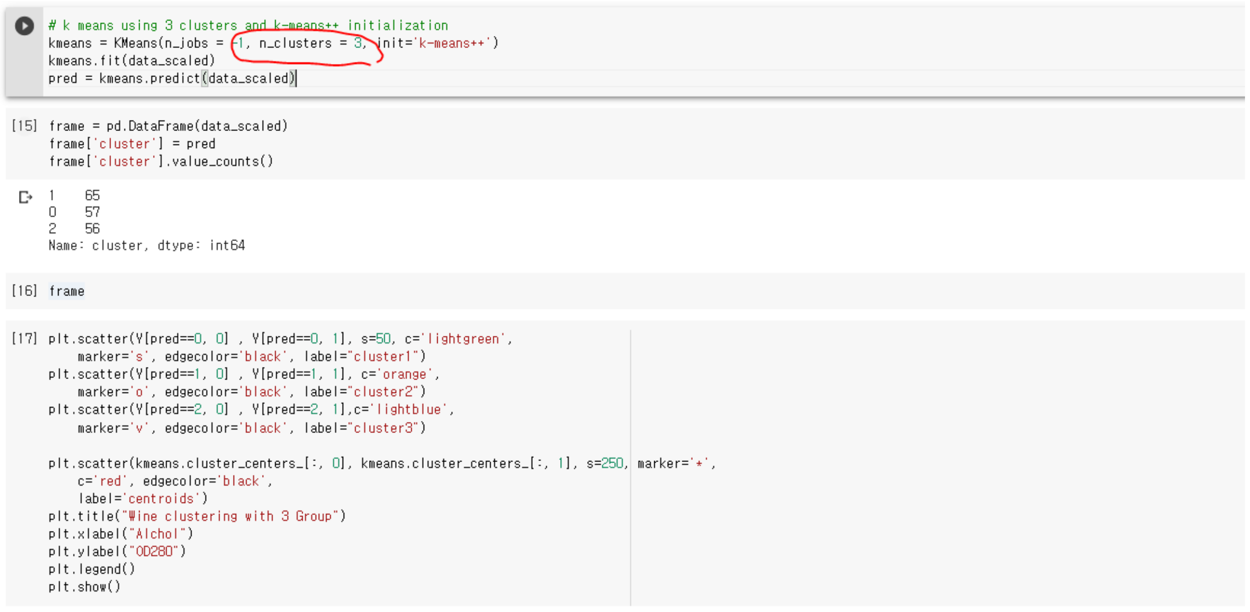
	0	1	cluster
0	1.518613	1.847920	0
1	0.246290	1.113449	0
2	0.196879	0.788587	0
3	1.691550	1.184071	0
4	0.295700	0.449601	0
...	...	...	...

```
[12] # make a numpy array for feature values.
     Y = np.array(list(zip(frame[0].values,frame[1].values)))
```

```
# draw cluster
plt.scatter(Y[pred==0, 0] , Y[pred==0, 1], s=50, c='lightgreen',
    marker='s', edgecolor='black', label="cluster1")
plt.scatter(Y[pred==1, 0] , Y[pred==1, 1],c='orange',
    marker='o', edgecolor='black', label="cluster2")

# draw centroid
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=250, marker='*',
    c='red', edgecolor='black',
    label='centroids')
plt.title("Wine clustering with 2 Group")
plt.xlabel("Alchol")
plt.ylabel("OD280")
plt.legend()
plt.show()
```



2) k means using 3 clusters and k-means++ initialization



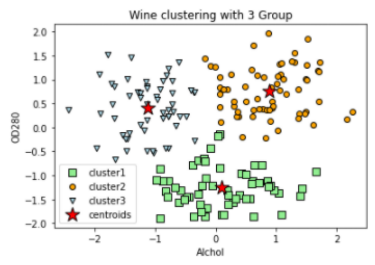
```
# k means using 3 clusters and k-means++ initialization
kmeans = KMeans(n_jobs = -1, n_clusters = 3, init='k-means++')
kmeans.fit(data_scaled)
pred = kmeans.predict(data_scaled)
```

```
[15] frame = pd.DataFrame(data_scaled)
     frame['cluster'] = pred
     frame['cluster'].value_counts()
```

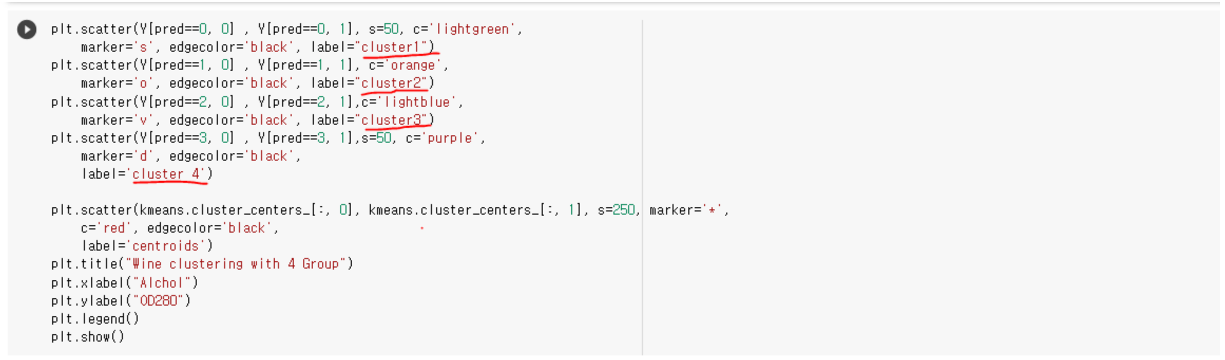
```
1    65
0    57
2    56
Name: cluster, dtype: int64
```

```
[16] frame
```

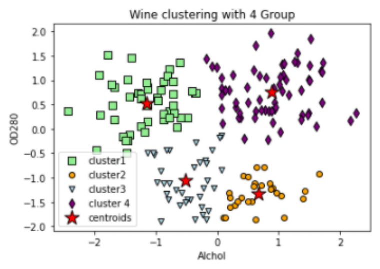
```
[17] plt.scatter(Y[pred==0, 0] , Y[pred==0, 1], s=50, c='lightgreen',
         marker='s', edgecolor='black', label="cluster1")
     plt.scatter(Y[pred==1, 0] , Y[pred==1, 1], c='orange',
         marker='o', edgecolor='black', label="cluster2")
     plt.scatter(Y[pred==2, 0] , Y[pred==2, 1],c='lightblue',
         marker='v', edgecolor='black', label="cluster3")

     plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=250, marker='*',
         c='red', edgecolor='black',
         label='centroids')
     plt.title("Wine clustering with 3 Group")
     plt.xlabel("Alchol")
     plt.ylabel("OD280")
     plt.legend()
     plt.show()
```



3) k means using 4 clusters and k-means++ initialization

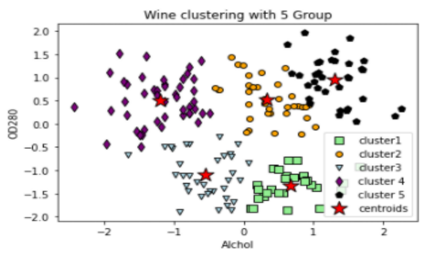


```
plt.scatter(Y[pred==0, 0] , Y[pred==0, 1], s=50, c='lightgreen',
    marker='s', edgecolor='black', label="cluster1")
plt.scatter(Y[pred==1, 0] , Y[pred==1, 1], c='orange',
    marker='o', edgecolor='black', label="cluster2")
plt.scatter(Y[pred==2, 0] , Y[pred==2, 1],c='lightblue',
    marker='v', edgecolor='black', label="cluster3")
plt.scatter(Y[pred==3, 0] , Y[pred==3, 1],s=50, c='purple',
    marker='d', edgecolor='black',
    label='cluster 4')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=250, marker='*',
    c='red', edgecolor='black',
    label='centroids')
plt.title("Wine clustering with 4 Group")
plt.xlabel("Alchol")
plt.ylabel("OD280")
plt.legend()
plt.show()
```



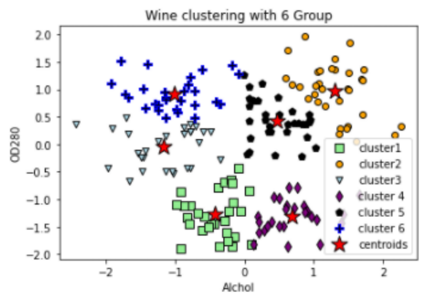
4) k means using 5 clusters and k-means++ initialization

```
plt.scatter(Y[pred==0, 0] , Y[pred==0, 1], s=50, c='lightgreen',
    marker='s', edgecolor='black', label="cluster1")
plt.scatter(Y[pred==1, 0] , Y[pred==1, 1], c='orange',
    marker='o', edgecolor='black', label="cluster2")
plt.scatter(Y[pred==2, 0] , Y[pred==2, 1],c='lightblue',
    marker='v', edgecolor='black', label="cluster3")
plt.scatter(Y[pred==3, 0] , Y[pred==3, 1],s=50, c='purple',
    marker='d', edgecolor='black',
    label='cluster 4')
plt.scatter(Y[pred==4, 0] , Y[pred==4, 1],s=50, c='black',
    marker='p', edgecolor='black',
    label='cluster 5')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=250, marker='*',
    c='red', edgecolor='black',
    label='centroids')
plt.title("Wine clustering with 5 Group")
plt.xlabel("Alchol")
plt.ylabel("OD280")
plt.legend()
plt.show()
```



5) k means using 6 clusters and k-means++ initialization

```
plt.scatter(Y[pred==0, 0] , Y[pred==0, 1], s=50, c='lightgreen',
    marker='s', edgecolor='black', label="cluster1")
plt.scatter(Y[pred==1, 0] , Y[pred==1, 1], c='orange',
    marker='o', edgecolor='black', label="cluster2")
plt.scatter(Y[pred==2, 0] , Y[pred==2, 1],c='lightblue',
    marker='v', edgecolor='black', label="cluster3")
plt.scatter(Y[pred==3, 0] , Y[pred==3, 1],s=50, c='purple',
    marker='d', edgecolor='black',
    label='cluster 4')
plt.scatter(Y[pred==4, 0] , Y[pred==4, 1],s=50, c='black',
    marker='p', edgecolor='black',
    label='cluster 5')
plt.scatter(Y[pred==5, 0] , Y[pred==5, 1],s=50, c='black',
    marker='P', edgecolor='blue',
    label='cluster 6')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=250, marker='*',
    c='red', edgecolor='black',
    label='centroids')
plt.title("Wine clustering with 6 Group")
plt.xlabel("Alchol")
plt.ylabel("OD280")
plt.legend()
plt.show()
```



Conclusion:
Based on elbow curve, I guessed k number for clustering from 2 to 6. Actually it is hard to accurately select k number in k means clustering. However, When You see Elbow curve, inertia value is leveled off after 2.5. So we can guess the best number for k is 3. As You can see when k=3, it is clustered well visually. So one can decide 3 is the appropriate number for k.

Challenges that I faced:

The most difficult challenge that I faced was that how to find optimized number for k means clustering. I want prove it with some number but It was hard to do that. I could have guessed the best number for the clustering number based on elbow curve finally. It would be better if I would use different kind of cluster algorithm like hierarchical clustering.

Video link
https://www.youtube.com/watch?v=teqxvmg1gKQ