Assignment 4 – Solution

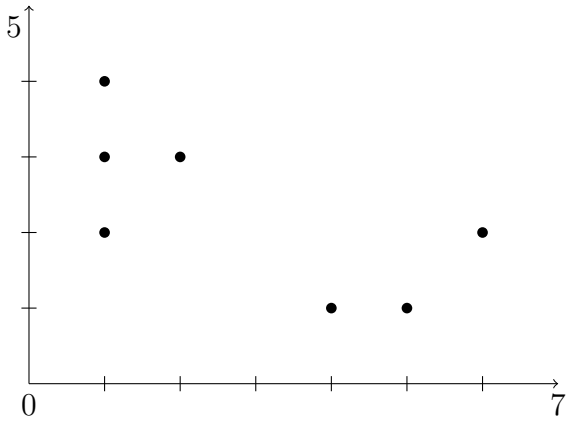
Machine Learning
MSc Business Analytics

Xiaocheng Li

1 Individual Assignment

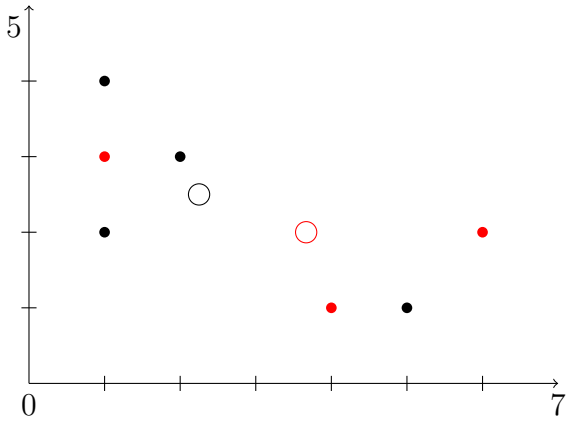
1. *Plot the observations in a two-dimensional graph.*

The graph looks as follows:



2. *Perform K-means clustering with $K = 2$ using the Euclidean norm. Toss a coin 7 times to initialise the algorithm.*

First we assign randomly $C_1 = \{2, 6, 7\}$ (red) and $C_2 = \{1, 3, 4, 5\}$ (black):



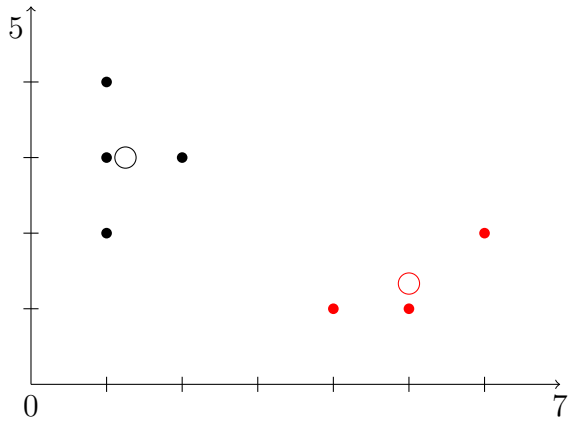
We compute the centroids of the two classes as $\boldsymbol{c}_1 = (\frac{11}{3}, 2)$ and $\boldsymbol{c}_2 = (\frac{9}{4}, \frac{5}{2})$. We now recompute the distances:

Obs. i	x_{i1}	x_{i2}	dist$(\boldsymbol{x}_i, \boldsymbol{c}_1)$	dist$(\boldsymbol{x}_i, \boldsymbol{c}_2)$
1	1	4	2.84	1.95
2	1	3	2.84	1.95
3	1	2	4.01	2.79
4	5	1	1.33	2.79
5	2	3	4.33	3.5
6	6	2	3.07	4.03
7	4	1	2.02	3.05

After reassignment, the new clusters are $C_1 = \{4, 6, 7\}$ and $C_2 = \{1, 2, 3, 5\}$. The new centroids of the two clusters are $\boldsymbol{c}_1 = (5, \frac{4}{3})$ and $\boldsymbol{c}_2 = (\frac{5}{4}, 3)$.

Obs. i	x_{i1}	x_{i2}	dist$(\boldsymbol{x}_i, \boldsymbol{c}_1)$	dist$(\boldsymbol{x}_i, \boldsymbol{c}_2)$
1	1	4	4.01	2.01
2	1	3	4.01	2.01
3	1	2	5.42	2.01
4	5	1	0.67	3.88
5	2	3	5.55	3.09
6	6	2	2.85	4.85
7	4	1	1.66	4.06

The clusters are still $C_1 = \{4, 6, 7\}$ and $C_2 = \{1, 2, 3, 5\}$. The algorithm thus terminates with the following result:



3. *Cluster the data using hierarchical clustering with complete linkage and the Euclidean norm. Draw the resulting dendrogram.*

We calculate the following pairwise distances between the observations:

	{1}	{2}	{3}	{4}	{5}	{6}	{7}
{1}	**0**						
{2}	1	**0**					
{3}	2	1	**0**				
{4}	5	4.47	4.12	**0**			
{5}	1.41	1	1.41	3.6	**0**		
{6}	5.38	5.09	5	1.41	4.12	**0**	
{7}	4.24	3.6	3.16	1	2.82	2.23	**0**

(Empty cells can be inferred from symmetry.) We first merge the 'clusters' {1} and {2}:

	{1, 2}	{3}	{4}	{5}	{6}	{7}
{1, 2}	**0**					
{3}	2	**0**				
{4}	5	4.12	**0**			
{5}	1.41	1.41	3.6	**0**		
{6}	5.38	5	1.41	4.12	**0**	
{7}	4.24	3.16	1	2.82	2.23	**0**

We now merge the 'clusters' {4} and {7}:

	{1, 2}	{3}	{5}	{6}	{4, 7}
{1, 2}	**0**				
{3}	2	**0**			
{5}	1.41	1.41	**0**		
{6}	5.38	5	4.12	**0**	
{4, 7}	5	4.12	3.6	2.23	**0**

We now merge the 'clusters' {5} and {1, 2}:

	{1, 2, 5}	{3}	{6}	{4, 7}
{1, 2, 5}	**0**			
{3}	2	**0**		
{6}	5.38	5	**0**	
{4, 7}	5	4.12	2.23	**0**

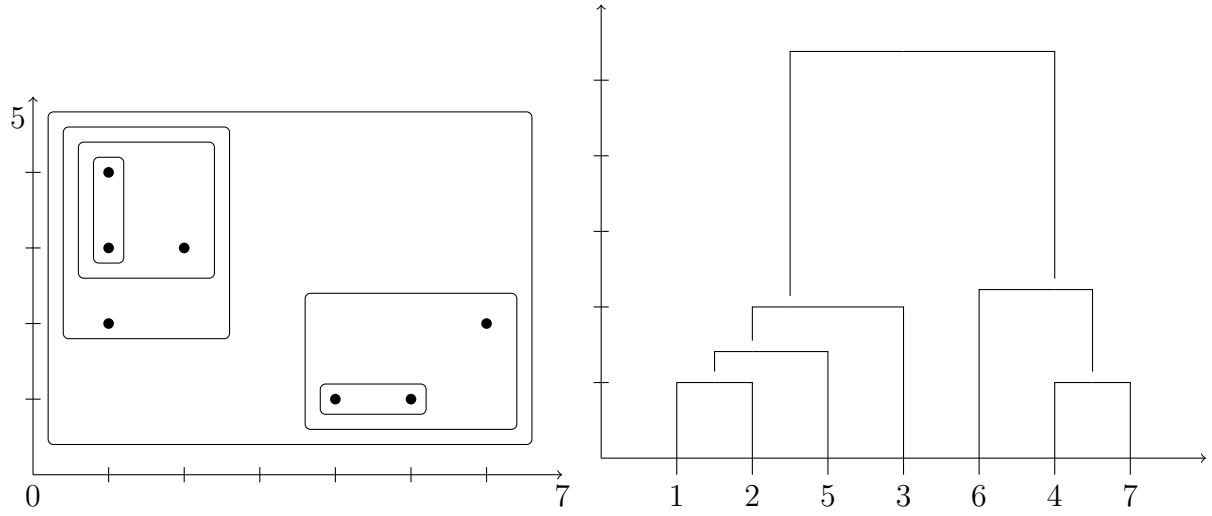
We now merge the 'clusters' {3} and {1, 2, 5}:

	{1, 2, 3, 5}	{6}	{4, 7}
{1, 2, 3, 5}	**0**		
{6}	5.38	**0**	
{4, 7}	5	2.23	**0**

We now merge the 'clusters' {6} and {4, 7}:

	{1, 2, 3, 5}	{4, 6, 7}
{1, 2, 3, 5}	**0**	
{4, 6, 7}	5.38	**0**

After merging the clusters $\{1, 2, 3, 5\}$ and $\{4, 6, 7\}$, we obtain the following result:



2 Group Assignment

1. *Load the customers.csv dataset. Apply a Z-score normalisation on the numerical features, i.e. age, income and score*

We start by loading the dataset and creating a new dataframe containing the normalised numerical features.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
df = pd.read_csv('data/customers.csv')
df.drop('ID', axis = 1, inplace=True)
scaler    = StandardScaler()
df_norm   = pd.DataFrame(data=scaler.fit_transform(df[['Age','Income',
    'Score']]),columns=df.columns[1:])
```

2. *Perform K-means with different k values, k = 2, 3, .., 10. In computing the distance, use only the normalised features from the previous point. Use a heuristic measure to find the best k.*

To evaluate the best k, we create the following function implementing the elbow method using both the *sklearn* and *yellowbrick* modules, in order to have a double check on the best k value.

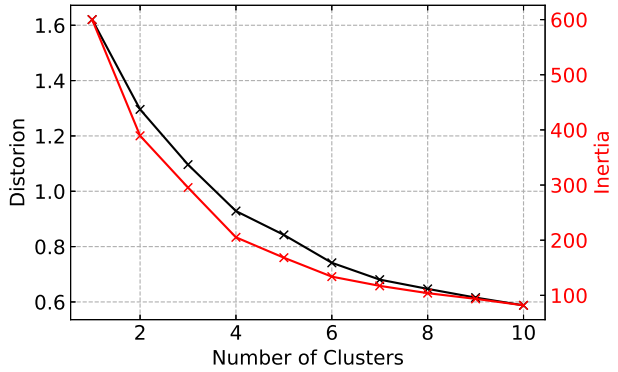
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist

def elbow(df_norm, labels):
        ## df_norm : normalised dataframe
        ## labels  : labels of interest, e.g. ['Age','Score']
        X  = df_norm[labels].values
        K = range(1,11)
        distortion = []
        inertia    = []

        for k in K:
                km = KMeans(n_clusters = k, init = 'k-means++',
                    max_iter = 300, n_init = 10, random_state = 0)
                km.fit(X)
                d  = sum(np.min(cdist(X, km.cluster_centers_, '
                    euclidean'),axis=1)) / X.shape[0]
                distortion.append(d)
                i  = km.inertia_
                inertia.append(i)

        # Plot and save figure
```

Using the function above with all labels, we obtain the following figure.



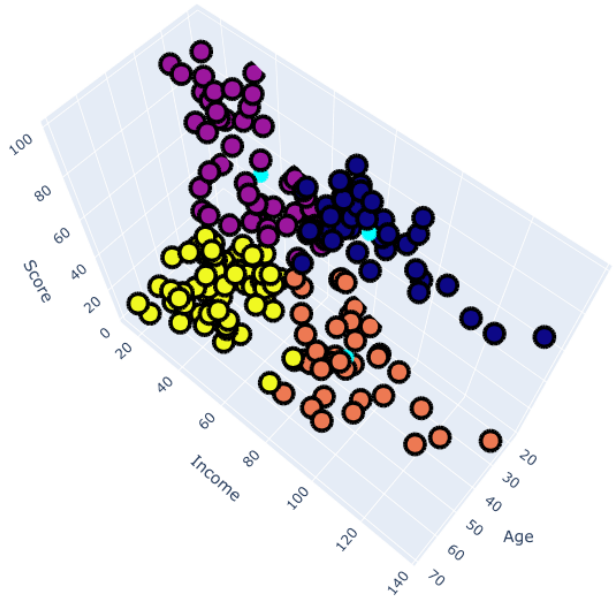
We can see an elbow at $k = 4$, although $k = 5$ would also be a suitable value.

3. *Cluster the samples using K-means with the best k. Plot the clusters and centroids (in 3D with denormalised axes). Can you find any meaningful results? Can you identify customer segments?*

We now perform K-Means using a value of $k = 4$ and plot the clusters.

```
X                = df_norm.values
km               = KMeans(n_clusters = 4, init = 'k-means++', max_iter
    = 300, n_init = 10, random_state = 0)
y_means = km.fit_predict(X)
centroids = scaler.inverse_transform(km.cluster_centers_)
# Plot clusters
```

The four clusters with centroids are visualised below.



At this point, we can investigate the clusters and draw some conclusions about customers segments:

- Cluster 1, in yellow, represents customers with low to mid income, low to mid score and mid to high age. This cluster might be composed of grocery shoppers, habitual customers whose low income renders them careful shoppers and in turn keeps their score in the low end. Most likely composed of working to middle class people, this group might be receptive to offers on everyday basic products.

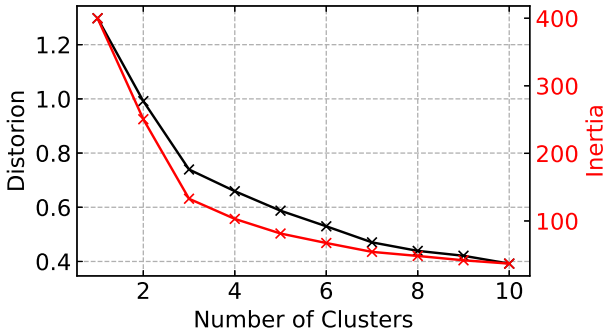
- Cluster 2, in purple, represents customers with low to mid income, mid to high score and low to mid age. This cluster is composed of young people, at most in their 30s. Most likely, these are mall customers that, unlike the previous group, are attracted by other product lines rather than groceries (e.g. clothing, entertainment). Their low income does not stop them from spending, therefore this group might be the most receptive to new shops and products.

- Cluster 3, in blue, represents customers with mid to high income, high score, low to mid age. This group is likely to be composed of the proverbial "young professionals". Their spending habit is likely to remain unchanged as they already classify with high score and income is not a problem for them.

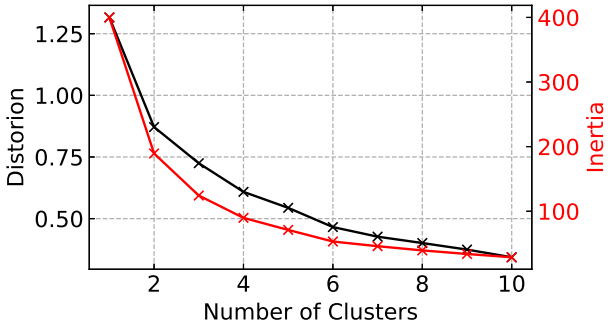
- Cluster 4, in orange, ranges across age and represent customers with mid to high income and low score. These are probably the occasional/passer-by customers.

4. *Create three different datasets, each with two out of the three features you normalised, i.e. (age,income), (age, score), (income, score). Perform K-means and find the best k for each of them.*

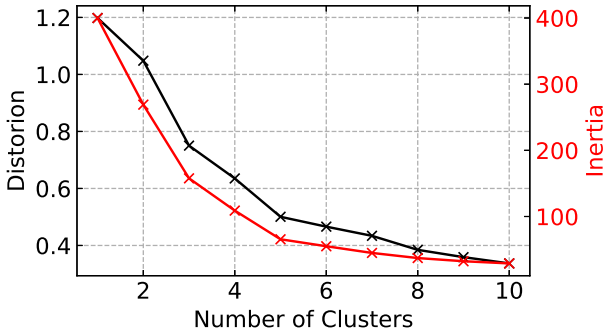
We use the function implemented in point 2 to find the best k for each dataset. Starting with (age, income) we obtain the following



We can see that there is a big change in steepness of the curve after $k = 3$. We can use a value of either $k = 3$ or $k = 4$, as after that the inertia value does not change much. We go for $k = 4$ and we'll discuss this choice when plotting the clusters.
Moving on to (age, score), we select $k = 4$.

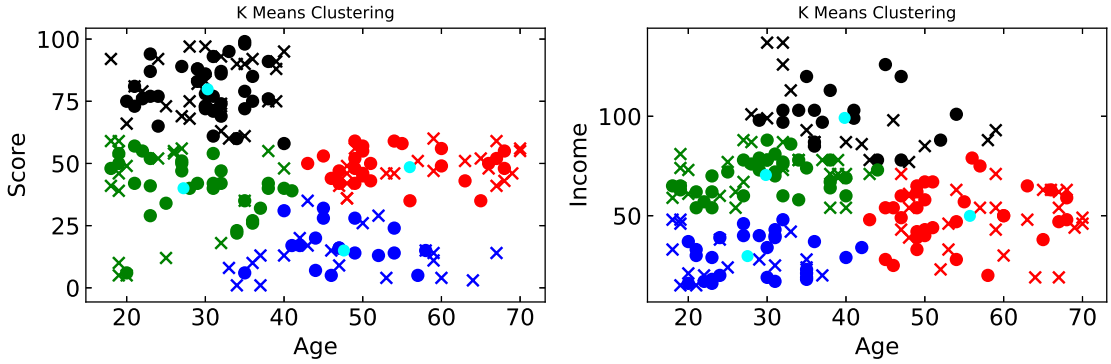


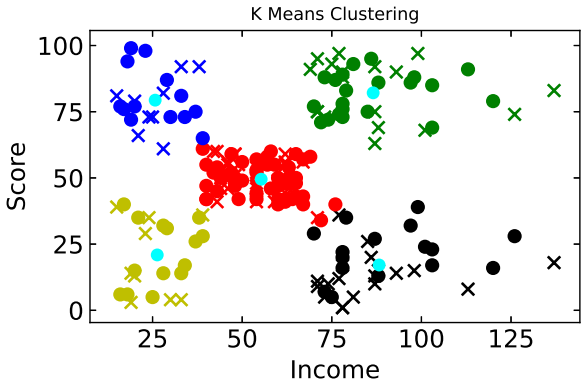
Finally for (income, score), we choose $k = 5$



5. *Plot the clusters and distinguish the data points based on the 'Gender' categorical feature. (You can use the visualisation method you prefer, e.g scatter plots with different markers, bar plots, etc.) Can you recognise customer subsegments that you did not identify earlier? Is it worth converting 'Gender' into a binary variable and including it in the K-means?*

We distinguish between *male* and *female* in the clusters, by using, respectively a cross and a dot. The clusters are plotted below





We can see that there is no distinguishable difference between genders, therefore we expect that introducing it as a variable in the clustering process will not enhance our predictions; as a matter of fact it might even deteriorate the process. We can see that the clusters are quite well separated for all pairs of features. The pairs (age,score) and (age, income) are not much more insightful than what we saw earlier when taking into account all features. On the other hand, we can see 5 very well distinct clusters on the (income, score) pair. We can, again, recognise classes such as *working class* with low income/low score, and *occasional shopper* with high income/low score, but we also notice a very compact cluster around the centre of the plot, most likely representing purely middle class customers. This strong division was not as clear in point 3, due to the conservative value of $k = 4$. With this more in depth analysis, we would ideally go back and change our number of clusters from $k = 4$ to $k = 5$ and add *middle class* as a stand-alone segment.

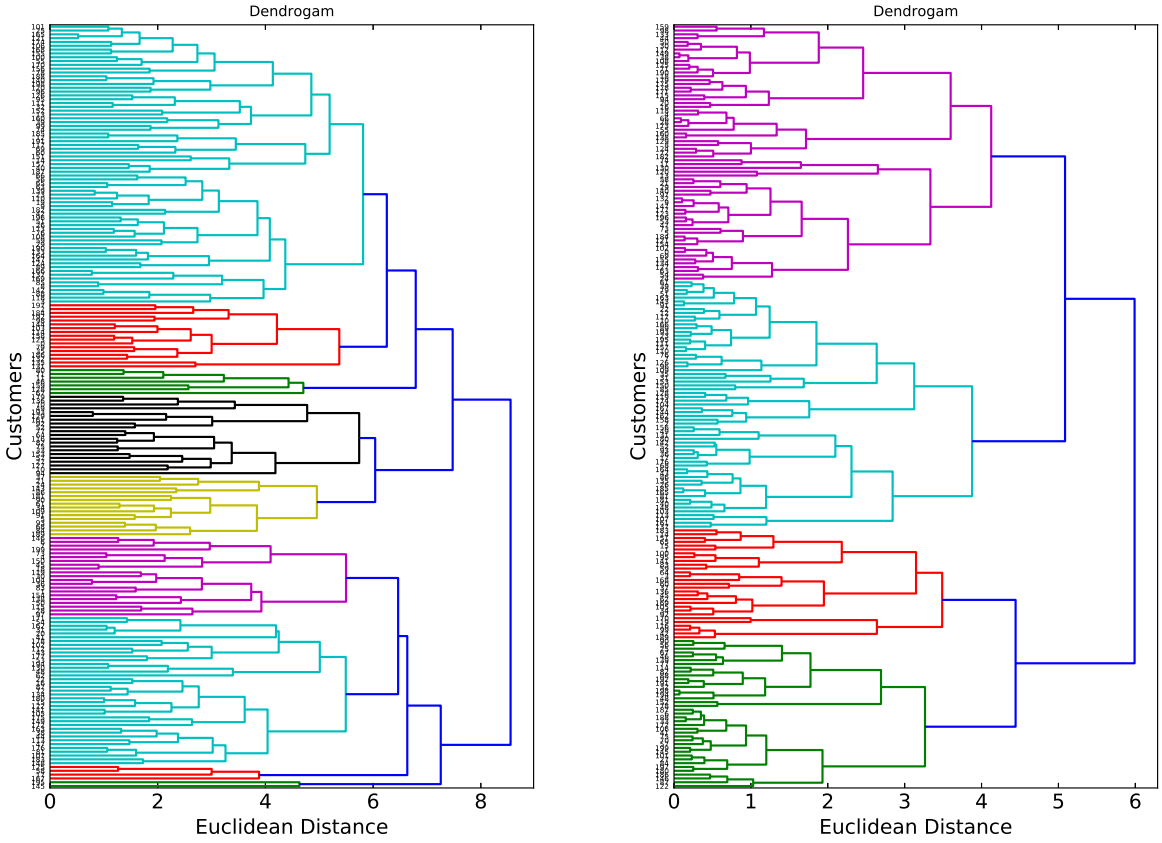
6. *Load the customers noisy.csv dataset. This dataset contains the four original features ('Gender' is now a binary feature) plus four noisy new ones. Perform hierarchical clustering on all features, plot the dendrogram and explain what you find.*

To evaluate the impact of noisy features on clustering, we plot the dendrograms for our original data (with gender), against the noisy dataset, using the same complete linkage.

```
import scipy.cluster.hierarchy as sch
df_noisy = pd.read_csv('data/customers_noisy.csv')
X_unproc = df_noisy.values
scaler   = StandardScaler().fit(X_unproc)
X_noisy  = scaler.transform(X_unproc)
df_orig  = df_noisy.iloc[:,:4]
X_unproc = df_orig.values
scaler   = StandardScaler().fit(X_unproc)
X_orig   = scaler.transform(X_unproc)

dendrogram_noisy = sch.dendrogram(sch.linkage(X_noisy, method = '
    complete', metric='euclidean'),orientation='right')
# Save figure
dendrogram_orig  = sch.dendrogram(sch.linkage(X_orig, method = '
    complete', metric='euclidean'),orientation='right')
# Save figure
```

We plot the dendrograms and note that, first of all, the complete linkage is extremely sensitive and picks up the noisy data, creating many more clusters. Moreover, we notice that the distance between leaves and their relative clades has increased and so has the distance between clades themselves. Take, for example, the green branch at the bottom of the dendrogram including the noisy data, we can see that this branch has only two leaves which have been clustered together although their distance is far greater than the average distance in the dataset.



(a) Noisy data

(b) Original