

CHAPTER 2

Unsupervised Learning

(Determine Number of Clusters with Python)

Content

- Introduction to Unsupervised Learning
- K-means clustering
- Probabilistic clustering via EM algorithm
- Hierarchical clustering
- **Determine Number of Clusters with Python**
- Unsupervised Learning with Python

Tools

Tools to estimate the number of clusters for unsupervised learning problems

1. Silhouette analysis

References:

1. Elbow Curve Method
2. AIC, BIC
3. Hierarchical clustering

SILHOUETTE ANALYSIS

Idea of Silhouette Analysis

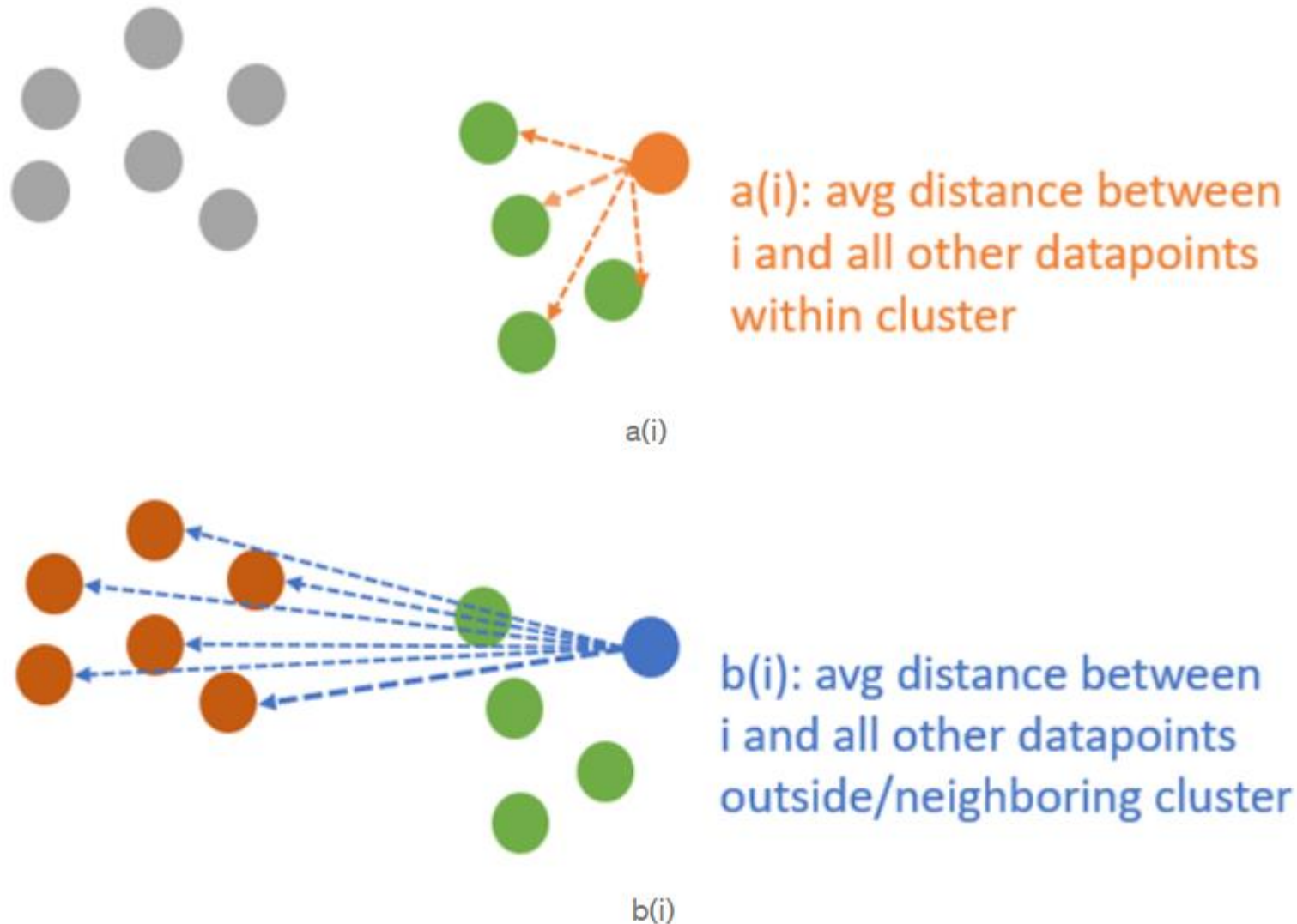
- The silhouette coefficient is a measure of between-cluster distance and within-cluster distance.

- The silhouette coefficient is defined as

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

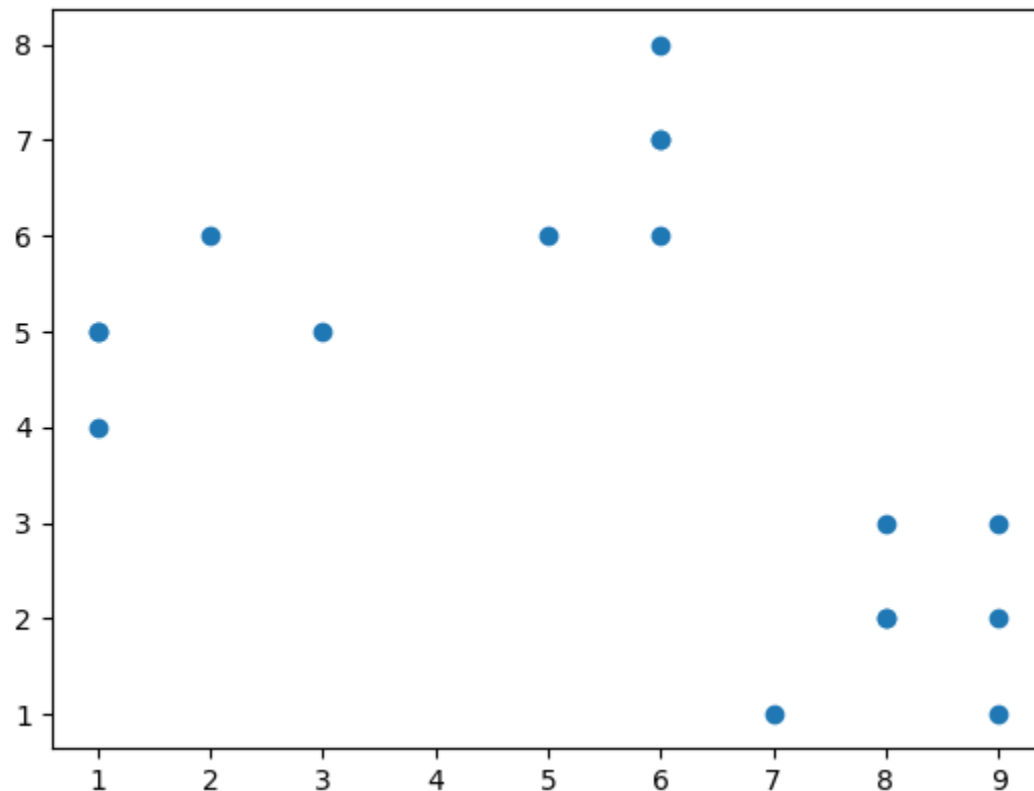
- The score is defined as the averaged silhouette coefficients.
- $S(i)$ is the silhouette coefficient of the data point i .
- $a(i)$ is the average distance between i and all the other data points in the cluster to which i belongs.
- $b(i)$ is the average distance from i to all clusters to which i does not belong.

Idea of Silhouette Analysis



Idea of Silhouette Analysis

- Consider the following data again. It has three clusters.



Idea of Silhouette Analysis

```
#Silhouette analysis
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans

X1 = [3, 1, 1, 2, 1, 6, 6, 6, 5, 6, 7, 8, 9, 8, 9, 9, 8]
X2 = [5, 4, 5, 6, 5, 8, 6, 7, 6, 7, 1, 2, 1, 2, 3, 2, 3]
data_frame = np.array([X1,X2]).T
plt.figure(); plt.scatter(X1,X2)

range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
silhouette_avg = []
for num_clusters in range_n_clusters:

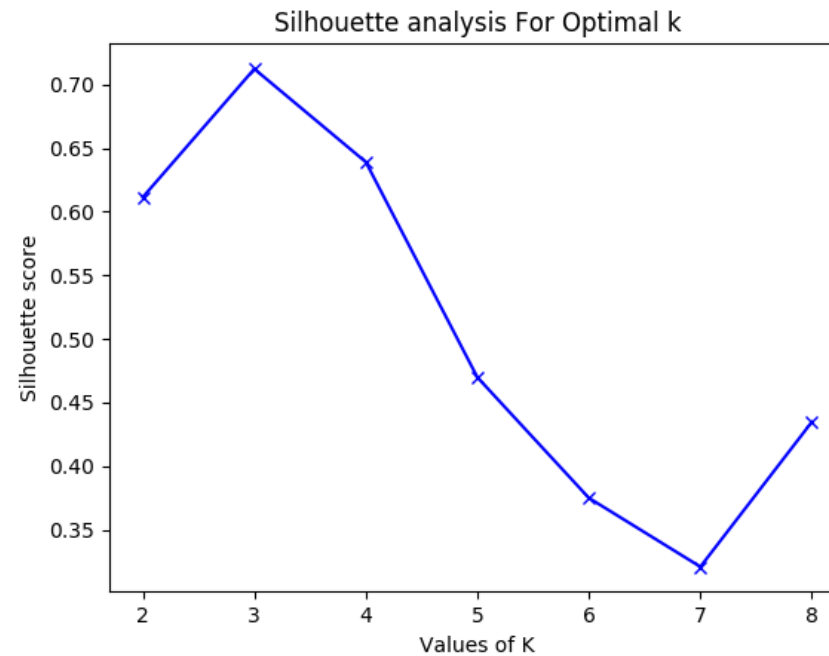
    # initialise kmeans
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(data_frame)
    cluster_labels = kmeans.labels_

    # silhouette score
    silhouette_avg.append(silhouette_score(data_frame, cluster_labels));

plt.figure();
plt.plot(range_n_clusters,silhouette_avg,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis For Optimal k')
```

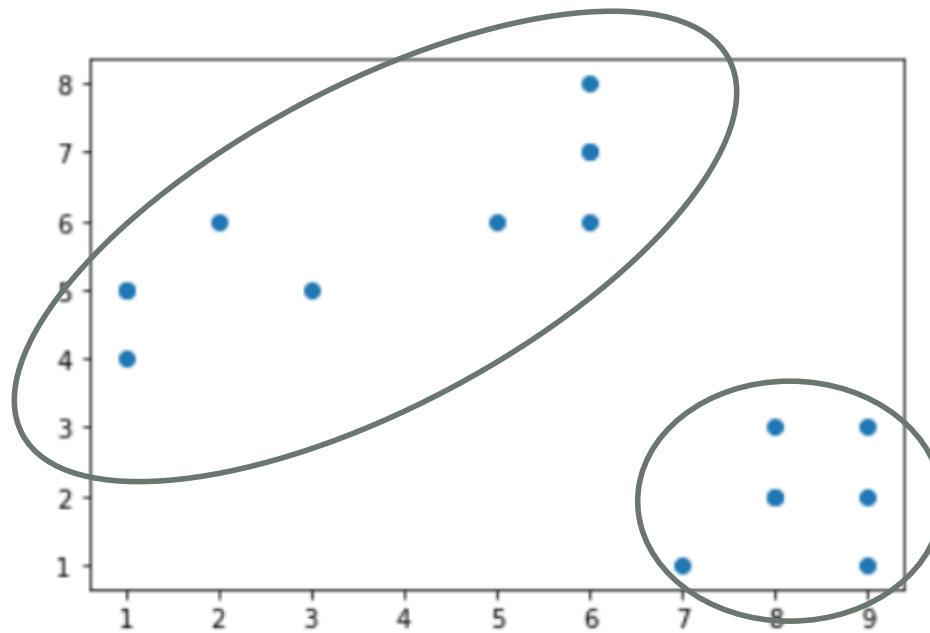

Idea of Silhouette Analysis

- Average silhouette coefficients are shown in the following figure.
- The largest value is the number of clusters. It has three clusters.



Idea of Silhouette Analysis

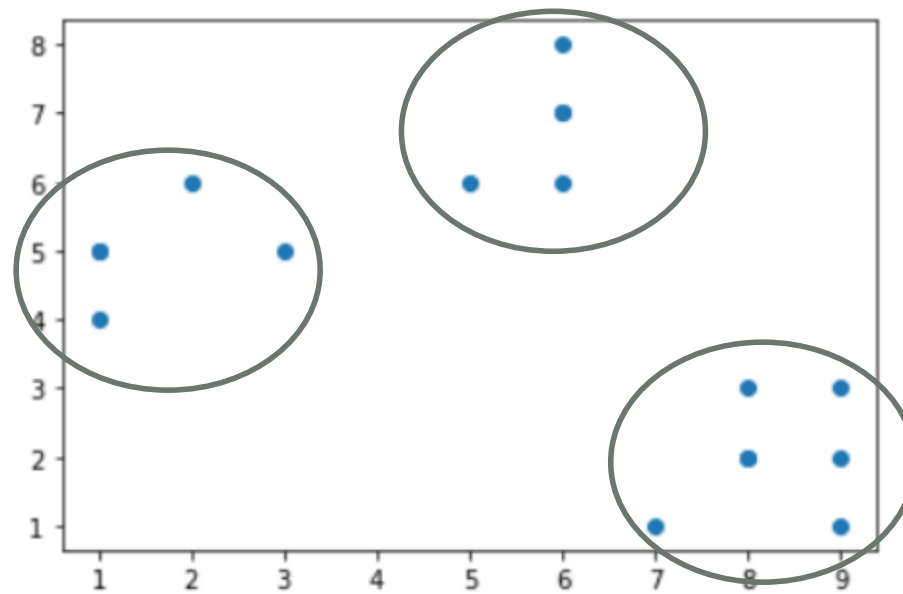
- Why the method works?
- When $K = 2$, the variance of the big group is large. The denominator of $S(i)$ is big.



Scatter Plot between X1 and X2

Idea of Silhouette Analysis

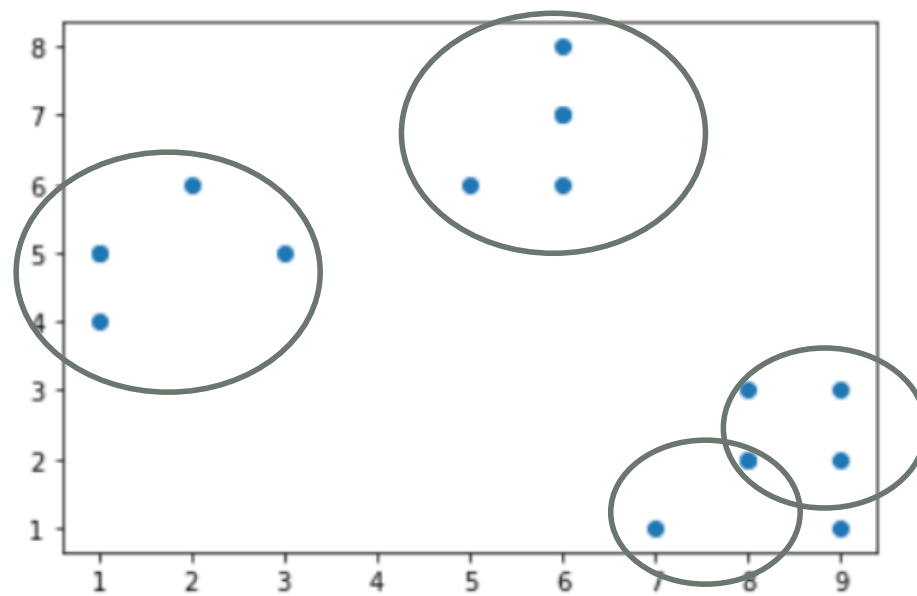
- When $K = 3$, it is the sum of the three variances. It fits for the data well. The degree of separability is the best. That's why the score reaches maximum.



Scatter Plot between X1 and X2

Idea of Silhouette Analysis

- When $K = 4$, it is the sum of the four variances. Two clusters have relatively small variances. The degree of separability is worse for the two nearby clusters. This reduces the separability and thus makes the scores smaller.



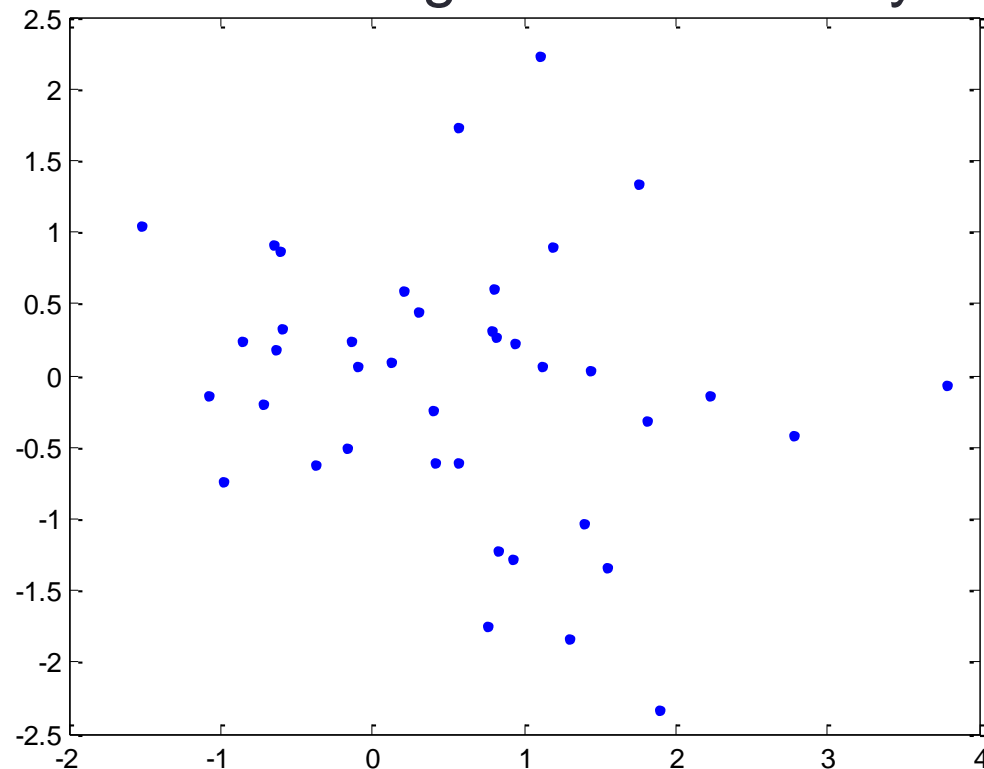
Scatter Plot between X1 and X2

Idea of Silhouette Analysis

- Similar to elbow curve method, silhouette analysis works well with well-separated clusters.
- It is usually used with K-means clustering method.
- However, for the data with not well-separated clusters, the method may not work well.
- But compared with elbow curve method, silhouette analysis is more capable to deal with data with not well-separated clusters.

Example 1 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data1.csv).
- Apply K-means clustering algorithm to partition the data into different clusters using silhouette analysis.



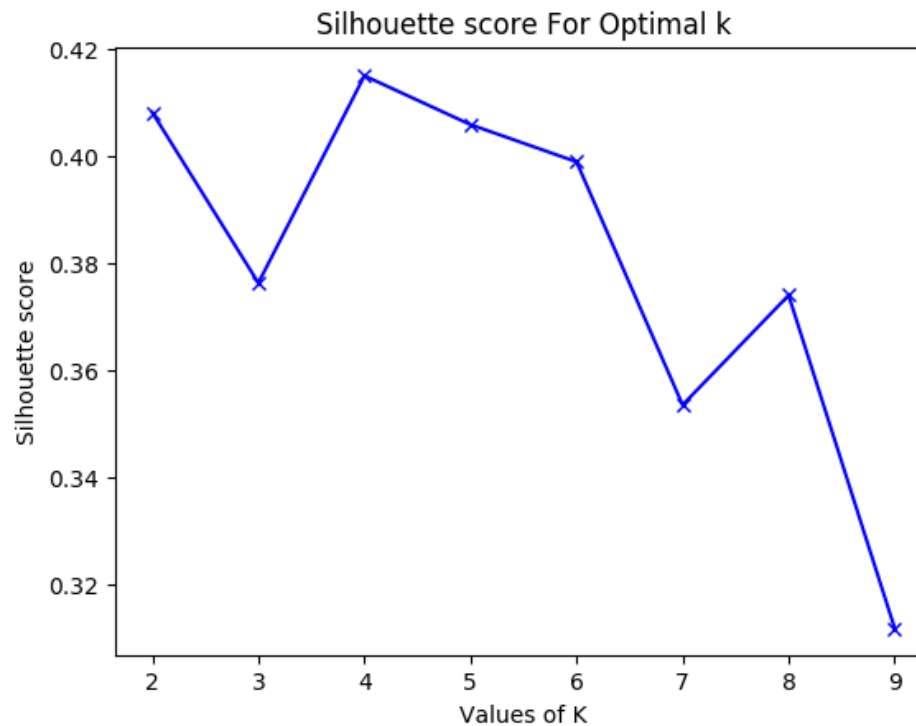
Example 1 - Question

```
#Example 1
from sklearn.metrics import silhouette_score
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
data = pd.read_csv('Data\\kmeans_data1.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data

K = range(2,11); silhouette_avg = []
for num_clusters in K :
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(data)
    cluster_labels = kmeans.labels_
    silhouette_avg.append(silhouette_score(data, cluster_labels));
plt.figure();
plt.plot(K,silhouette_avg,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette score For Optimal k')
```

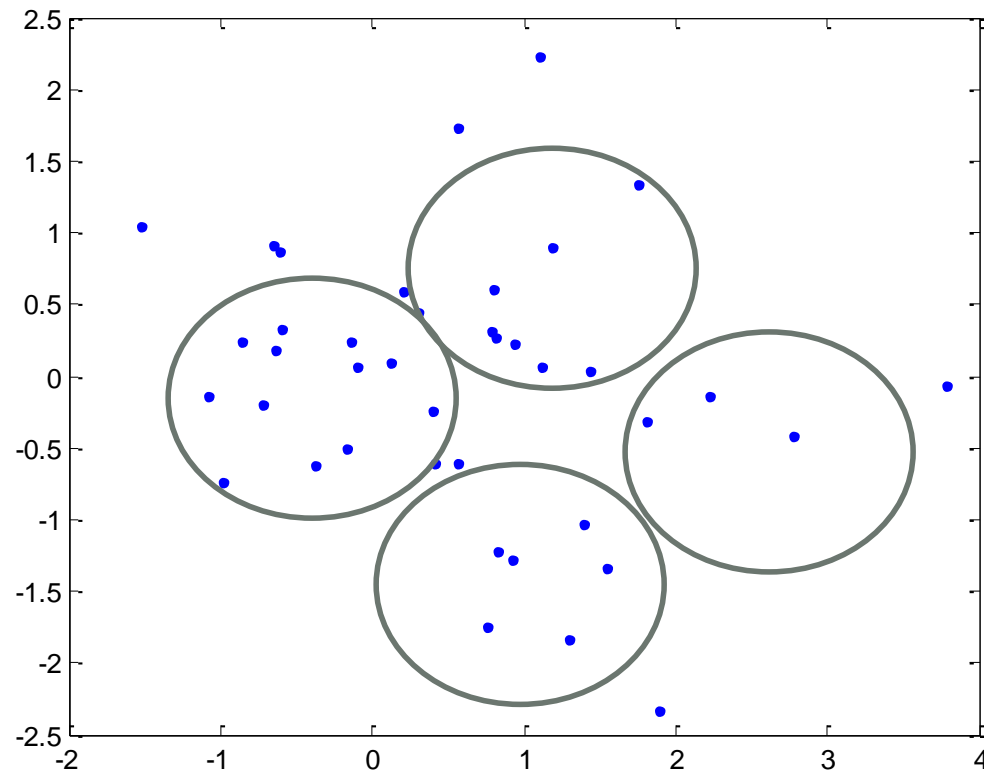
Example 1 - Question

- The sum of squares are shown as below.
- However, the scores reach maximum at $K = 4$.



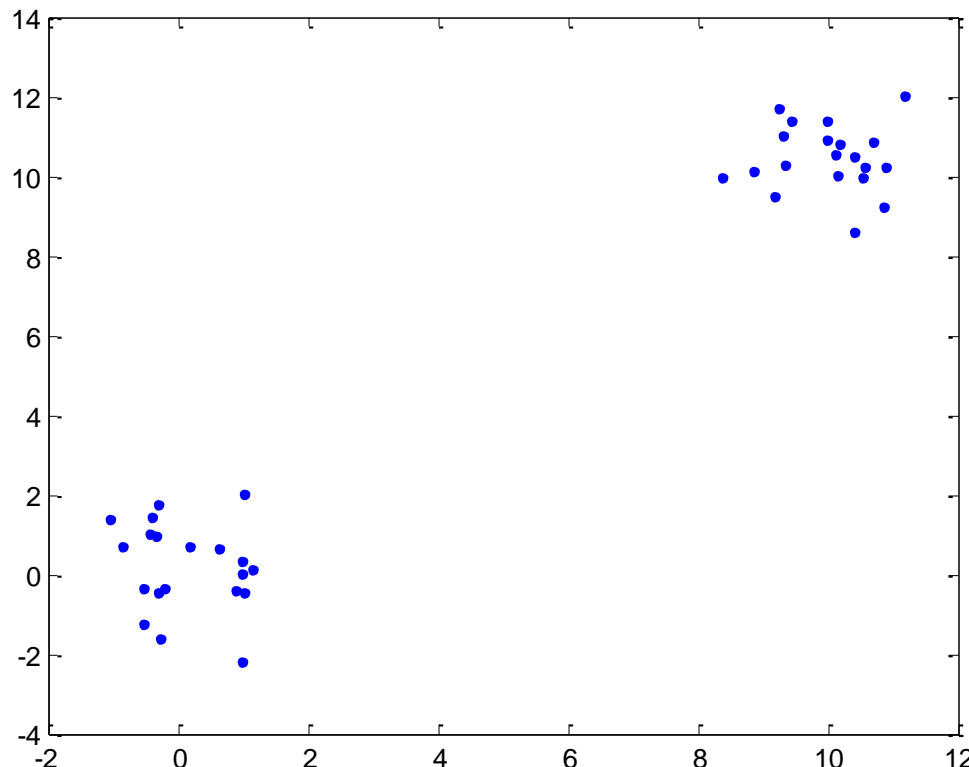
Example 1 - Question

- In the data, it seems to be reasonable.



Example 2 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data2.csv).
- Apply K-means clustering algorithm to partition the data into different clusters using silhouette analysis.



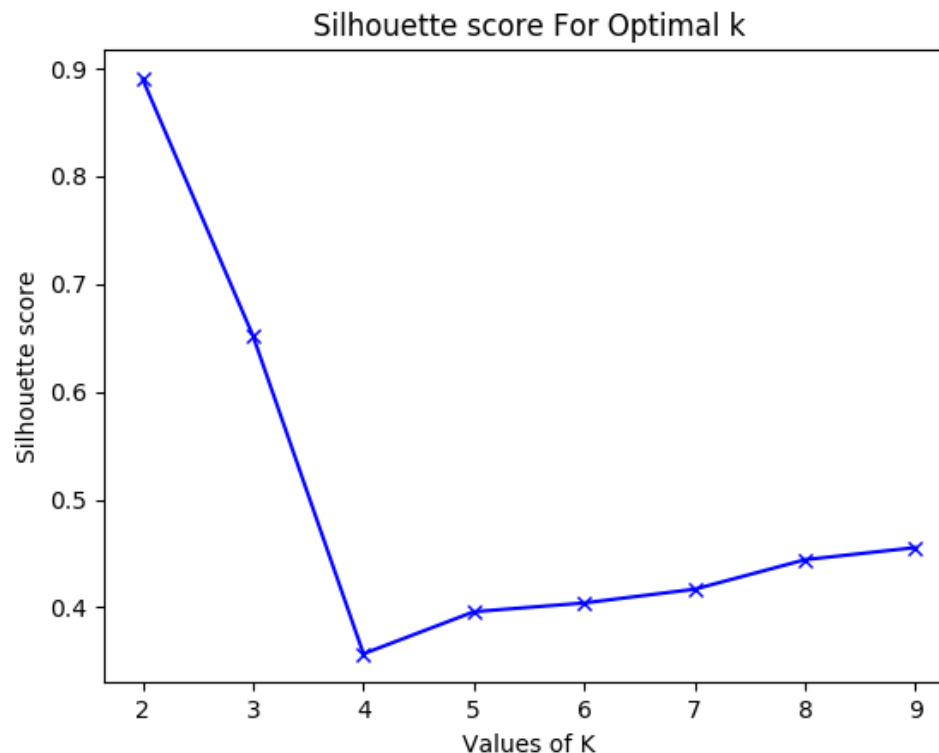
Example 2 - Question

```
#Example 2
from sklearn.metrics import silhouette_score
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
data = pd.read_csv('Data\\kmeans_data2.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data

K = range(2,11); silhouette_avg = []
for num_clusters in K :
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(data)
    cluster_labels = kmeans.labels_
    silhouette_avg.append(silhouette_score(data, cluster_labels));
plt.figure();
plt.plot(K,silhouette_avg,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette score For Optimal k')
```

Example 2 - Question

- The silhouette scores are shown as below.
- The maximum is reached at $K = 2$. There are two clusters.
- In the data, the two clusters are well-separated.



Example 3 - Question

- Cluster the Iris dataset with K-means clustering algorithm using silhouette analysis.
- There are 3 groups with 150 samples and 4 attributes.
- The three groups are Iris-setosa, Iris-versicolor and Iris-virginica.
- Suppose we do not know that the dataset has 3 groups.

Example 3 - Question

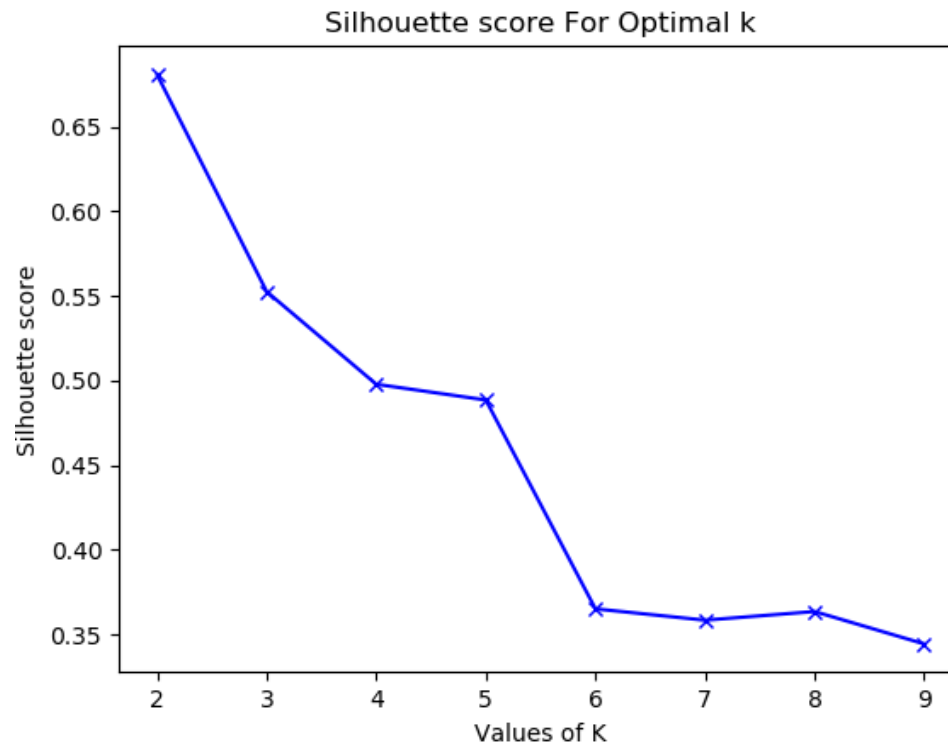
#Example 3

```
from sklearn.metrics import silhouette_score
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
data = pd.read_csv('Data\\iris.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
```

```
K = range(2,11); silhouette_avg = []
for num_clusters in K :
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(data.iloc[:, :-1])
    cluster_labels = kmeans.labels_
    silhouette_avg.append(silhouette_score(data.iloc[:, :-1], cluster_labels));
plt.figure();
plt.plot(K,silhouette_avg,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette score For Optimal k')
```

Example 3 - Question

- The scores are shown in the graph below.
- Obviously, the maximum is reached at $K = 2$.



Remark

- Similar to the elbow curve method, silhouette analysis can also be used to evaluate the quality of clusters.
- For K-means, the clustering results are depended on the initial cluster centers. We can apply K-means clustering (with fixed K) with different initial cluster centers. The one with the largest silhouette score is the best clustering results.
- Silhouette analysis is easier to use than the elbow curve method because it only requires the class labels and the data.
- Silhouette analysis can be used with K-means, probabilistic clustering, hierarchical clustering and ensemble clustering.

- The following slides are for reference only.

ELBOW CURVE METHOD

Review of K-means Clustering

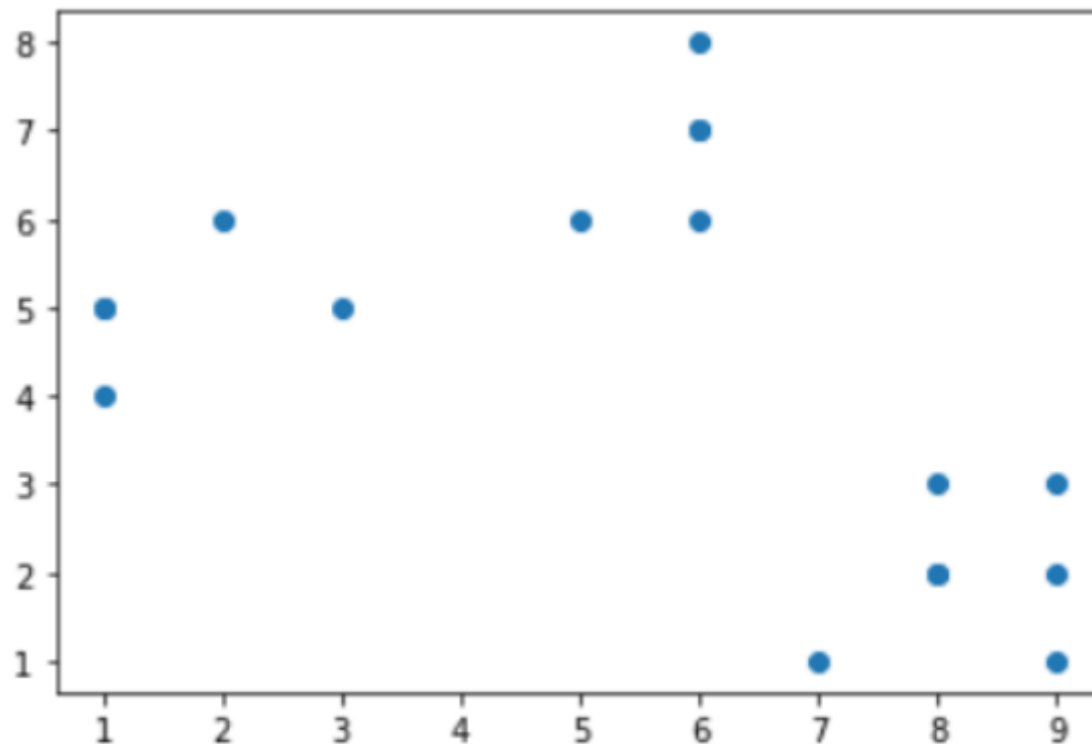
- The goal of K-means clustering is to minimize the following objective function

$$\min_{I_{ik}, \mathbf{c}_k} J(I_{ik}, \mathbf{c}_k), \text{ where } J(I_{ik}, \mathbf{c}_k) = \sum_{i=1}^n \sum_{k=1}^c I_{ik} \|\mathbf{x}_i - \mathbf{c}_k\|^2$$

where $I_{ik} = \{0, 1\}$ are binary variables and \mathbf{c}_k are the cluster centers.

Idea of Elbow Curve Method

- Consider the following data.
- It has three clusters.



Scatter Plot between X1 and X2

Idea of Elbow Curve Method ²⁹

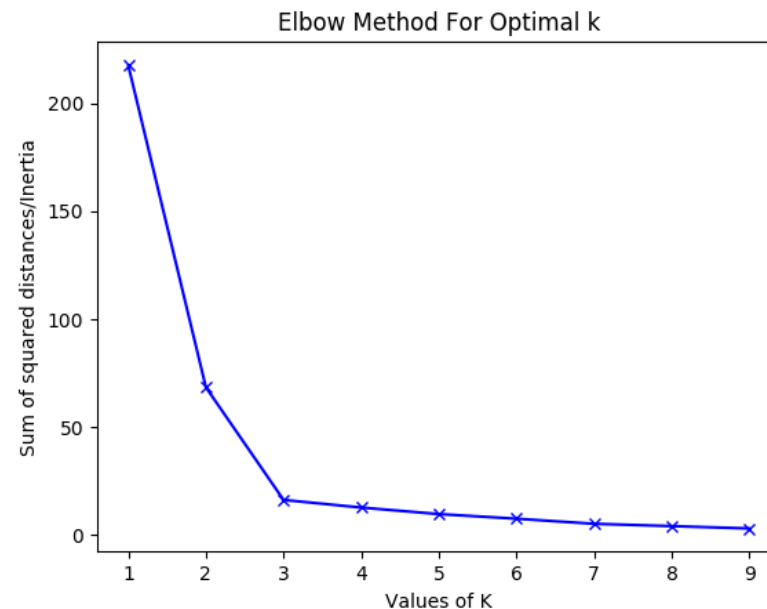
- We can perform K-means clustering to the dataset with number of clusters, 1, 2, 3, ... 10.

```
import numpy as np
import matplotlib.pyplot as plt
#import seaborn as sns
from sklearn.cluster import KMeans
X1 = [3, 1, 1, 2, 1, 6, 6, 6, 5, 6, 7, 8, 9, 8, 9, 9, 8]
X2 = [5, 4, 5, 6, 5, 8, 6, 7, 6, 7, 1, 2, 1, 2, 3, 2, 3]
plt.figure(); plt.scatter(X1,X2)

data_frame = np.array([X1,X2]).T
Sum_of_squared_distances = []
K = range(1,11)
for num_clusters in K :
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(data_frame)
    Sum_of_squared_distances.append(kmeans.inertia_)
plt.figure();
plt.plot(K,Sum_of_squared_distances,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Sum of squared distances/Inertia')
plt.title('Elbow Method For Optimal k')
```

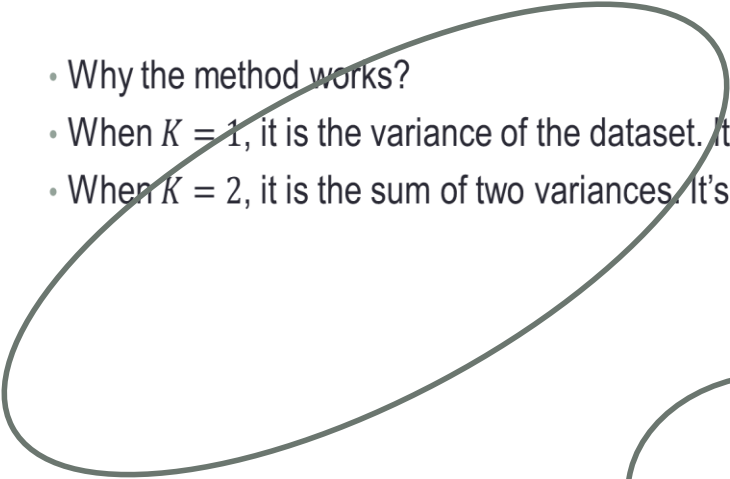
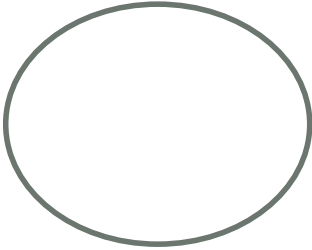
Idea of Elbow Curve Method

- We find the sum of squared distance with different number of clusters.
- The graph is shown as below.
- There is sharp decrease at $k = 3$. It means there is a high chance that there are totally three clusters in this data.



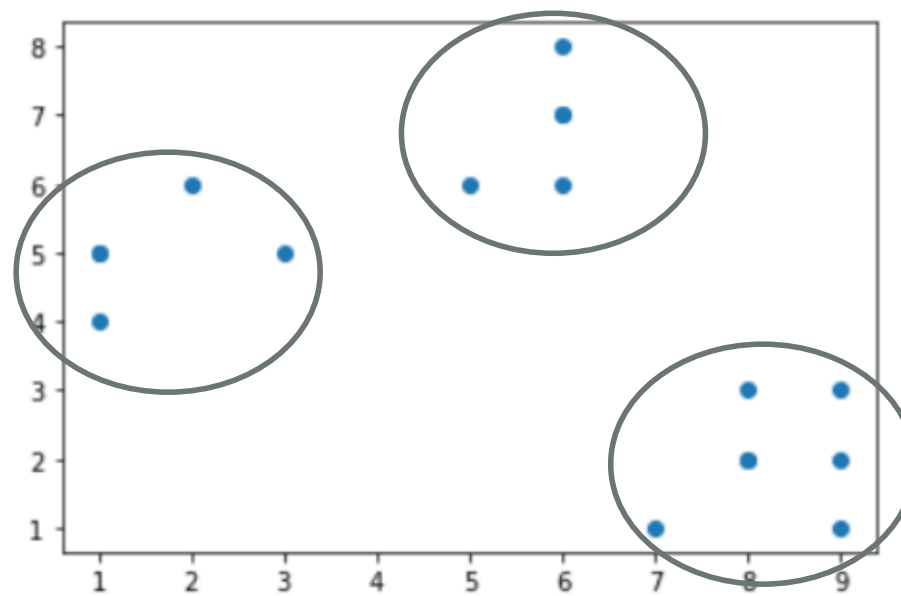
Idea of Elbow Curve Method

- Why the method works?
- When $K = 1$, it is the variance of the dataset. It's very big.
- When $K = 2$, it is the sum of two variances. It's smaller.

- 
- 
- Why the method works?
 - When $K = 1$, it is the variance of the dataset. It's very big.
 - When $K = 2$, it is the sum of two variances. It's smaller.

Idea of Elbow Curve Method

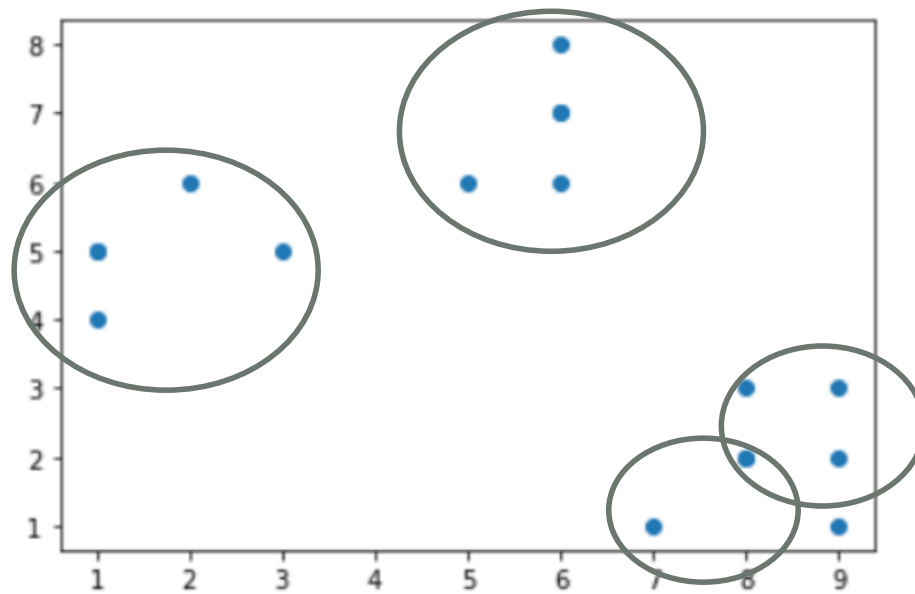
- When $K = 3$, it is the sum of the three variances. It fits for the data well. The variance is greatly reduced.



Scatter Plot between X1 and X2

Idea of Elbow Curve Method

- When $K = 4$, it is the sum of the four variances. Two clusters have relatively small variances. The sum of these four variances is smaller than three's but no sharp decrease.



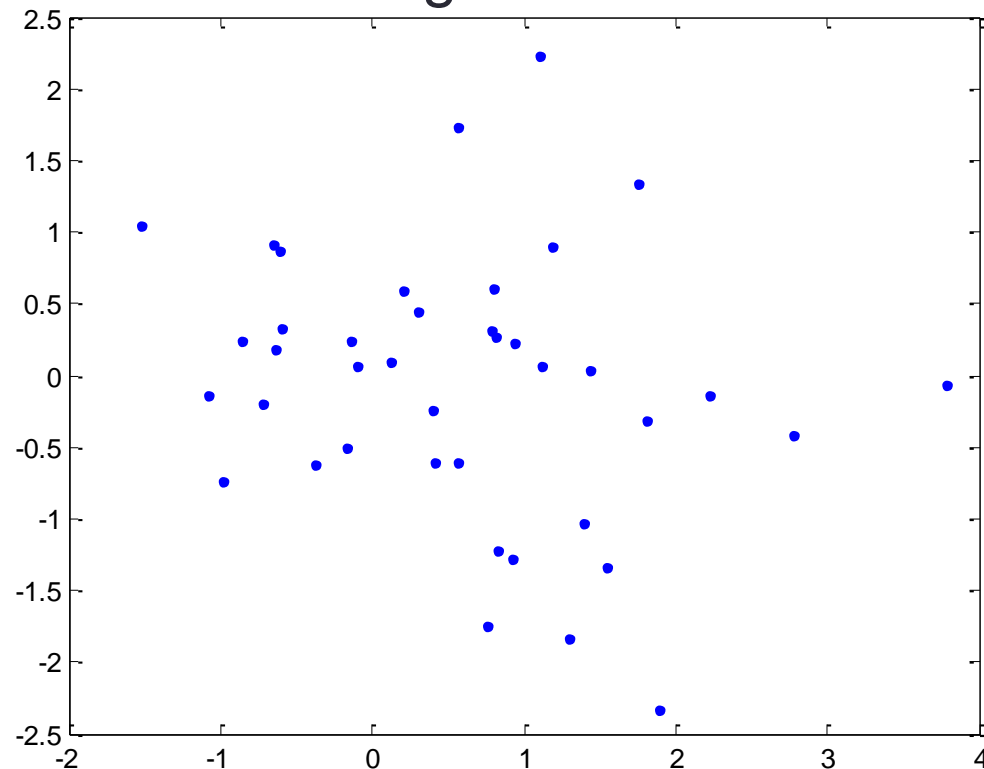
Scatter Plot between X1 and X2

Idea of Elbow Curve Method

- Elbow curve method works well with well-separated clusters.
- It is usually used with K-means clustering method.
- However, for the data with not well-separated clusters, the method may not work well.

Example 1 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data1.csv).
- Apply K-means clustering algorithm to partition the data into different clusters using elbow method.



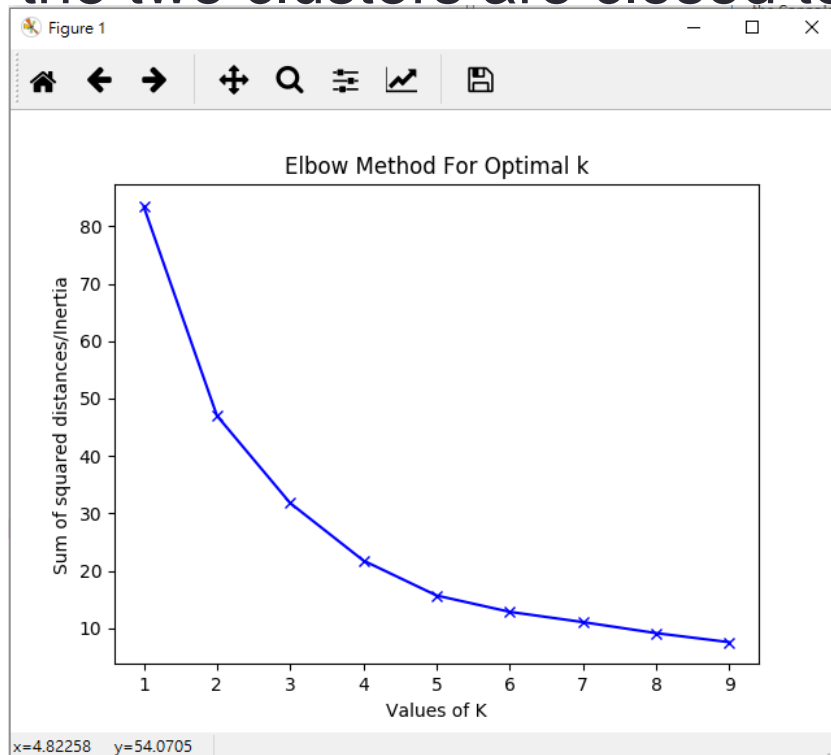
Example 1 - Question

```
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
data = pd.read_csv('Data\\kmeans_data1.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data

K = range(1,11); Sum_of_squared_distances = []
for num_clusters in K :
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(data)
    Sum_of_squared_distances.append(kmeans.inertia_)
plt.figure();
plt.plot(K,Sum_of_squared_distances,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Sum of squared distances/Inertia')
plt.title('Elbow Method For Optimal k')
```

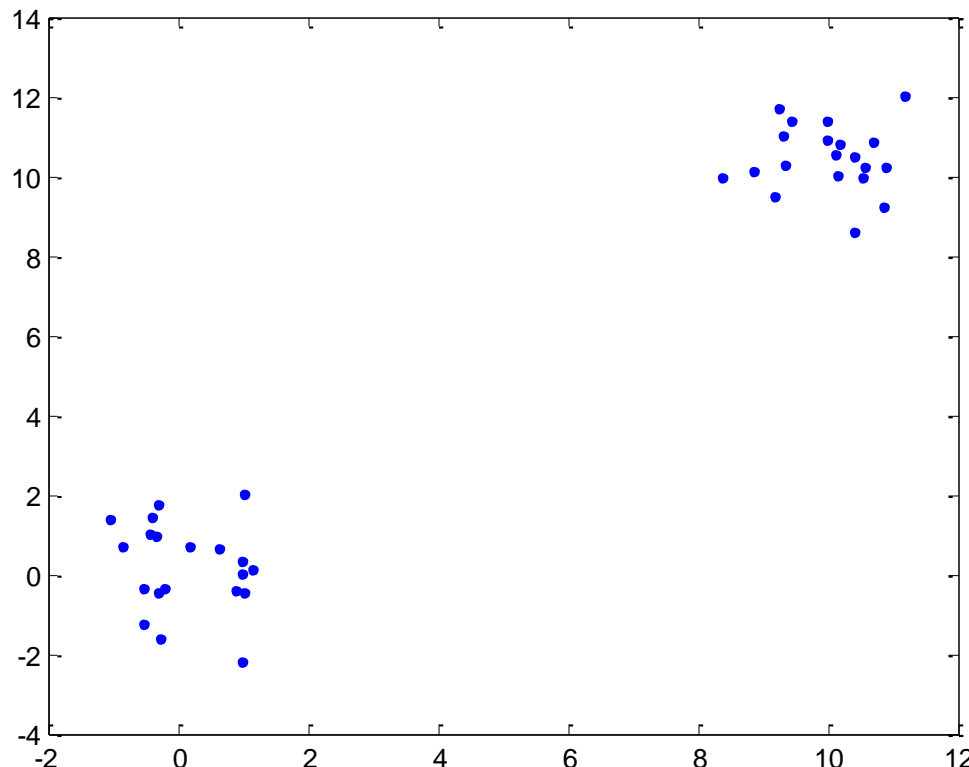
Example 1 - Question

- The sum of squares are shown as below.
- However, the drop is not significant. It's hard to estimate the number of clusters.
- In the data, the two clusters are closed to each other.



Example 2 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data2.csv).
- Apply K-means clustering algorithm to partition the data into two different clusters.



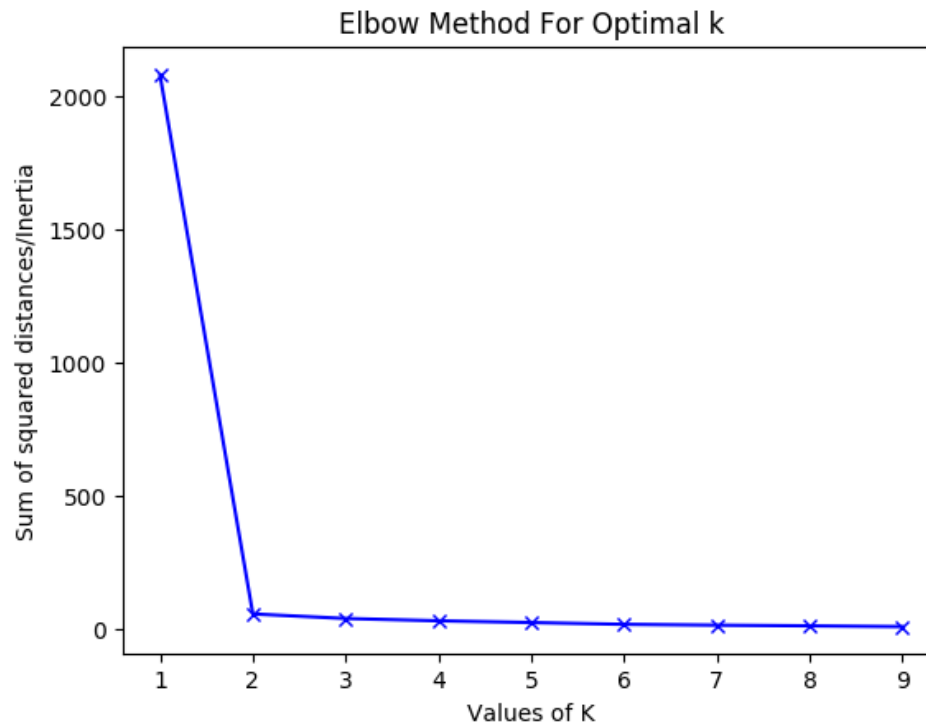
Example 2 - Question

```
#Example 2
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
data = pd.read_csv('Data\\kmeans_data2.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data

K = range(1,11); Sum_of_squared_distances = []
for num_clusters in K :
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(data)
    Sum_of_squared_distances.append(kmeans.inertia_)
plt.figure();
plt.plot(K,Sum_of_squared_distances,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Sum of squared distances/Inertia')
plt.title('Elbow Method For Optimal k')
```

Example 2 - Question

- The sum of squares are shown as below.
- There is sharp drop. There are two clusters.
- In the data, the two clusters are well-separated.



Example 3 - Question

- Cluster the Iris dataset with K-means clustering algorithm.
- There are 3 groups with 150 samples and 4 attributes.
- The three groups are Iris-setosa, Iris-versicolor and Iris-virginica.
- Suppose we do not know that the dataset has 3 groups.

Example 3 - Question

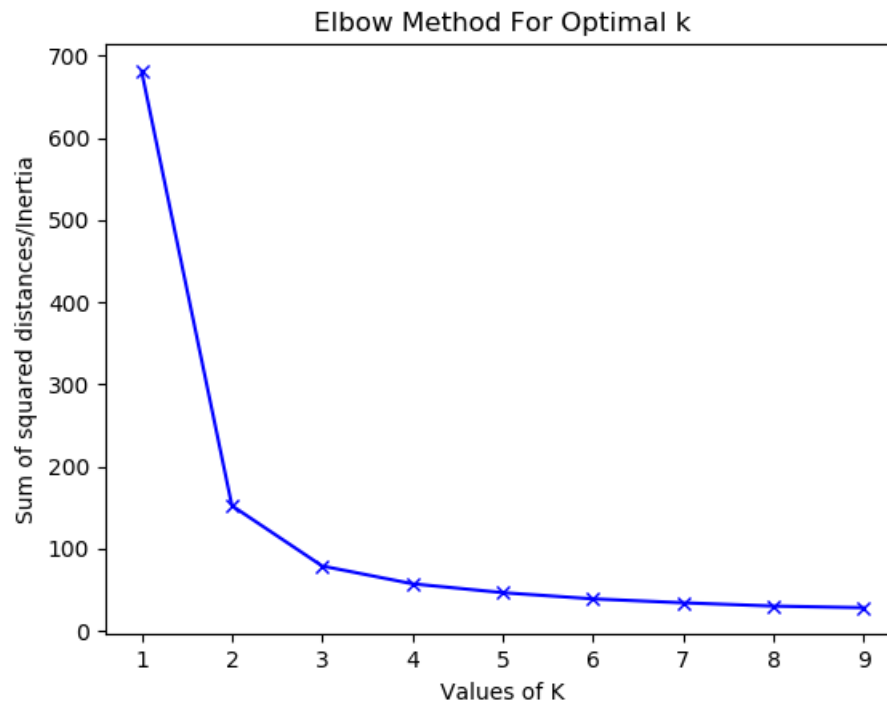
#Example 3

```
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
data = pd.read_csv('Data\\iris.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
```

```
K = range(1,11); Sum_of_squared_distances = []
for num_clusters in K :
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(data.iloc[:,-1])
    Sum_of_squared_distances.append(kmeans.inertia_)
plt.figure();
plt.plot(K,Sum_of_squared_distances,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Sum of squared distances/Inertia')
plt.title('Elbow Method For Optimal k')
```

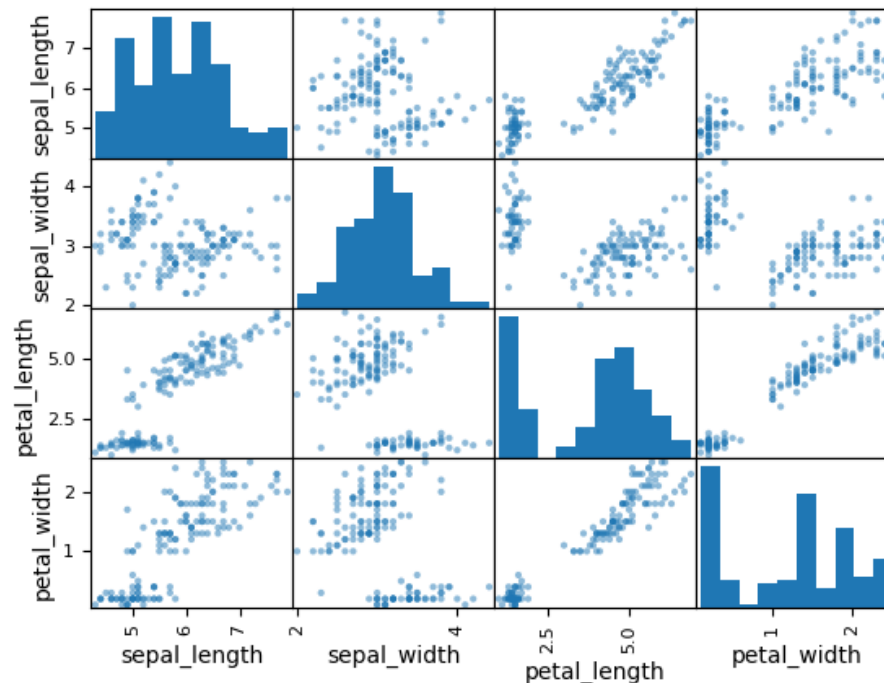
Example 3 - Question

- The scores using elbow method are shown as below.
- There is a sharp drop at $K = 2$. It means there are totally two clusters in the dataset.



Example 3 - Question

- If we look at the data, there are two groups of clusters.
- The reason why the method cannot detect there are three clusters is that two of the groups are closed to each other.



Remark

- The elbow curve method can also be used to evaluate the quality of clusters.
- For K-means, the clustering results are depended on the initial cluster centers. We can apply K-means clustering (with fixed K) with different initial cluster centers. The one with the smallest inertia is the best clustering results.

AIC, BIC

Akaike Information Criterion (AIC)

- An AIC value is calculated for each model of interest. The model with the lowest AIC is the best model.

$$AIC = -2 \ln L \{M_i | Y\} + 2p_i$$

$-\ln L \{M_i | Y\}$ is the negative log-likelihood

M_i is model i

Y is the data

p_i is the number of model parameters

- It's usually used with probabilistic clustering.
- AIC is also widely used in model selection in regression analysis, time series analysis, etc.
- Small value of AIC represents a good model.

Bayes Information Criterion, BIC

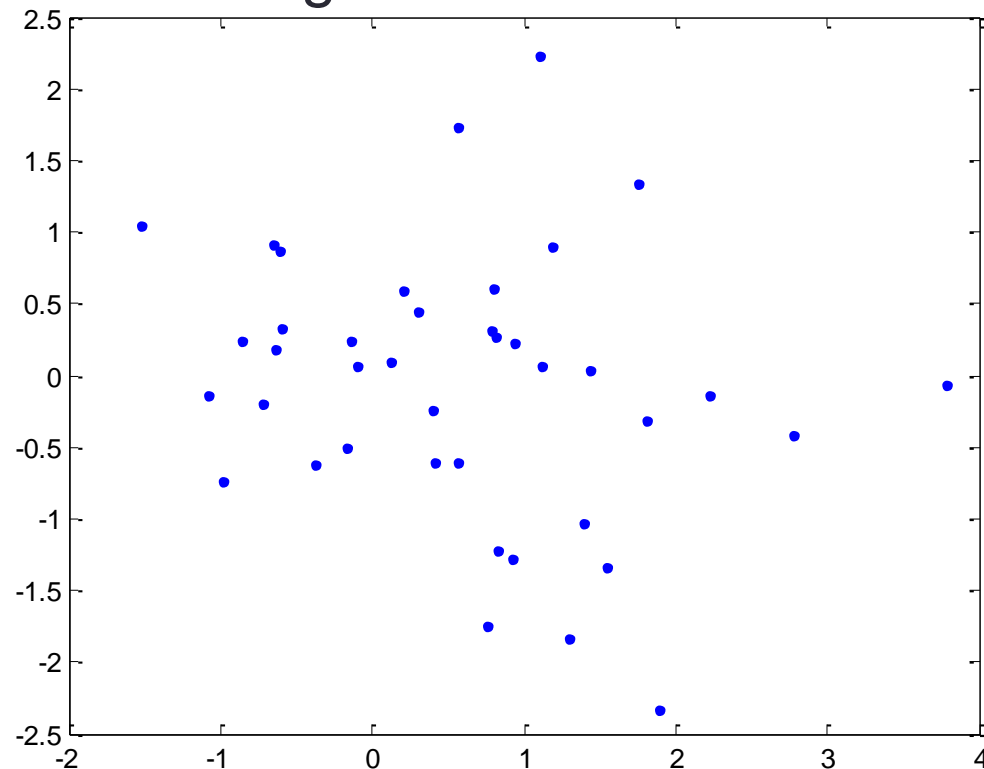
- Bayes Information Criterion, BIC

$$\text{BIC} = -2 \ln L \{M_i|Y\} + p_i \ln(n)$$

- It's usually used with probabilistic clustering.
- BIC is also widely used in model selection in regression analysis, time series analysis, etc.
- Small value of BIC represents a good model.
- AIC and BIC are used together.

Example 1 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data1.csv).
- Apply GMM clustering algorithm to partition the data into different clusters using AIC and BIC methods.



Example 1 - Question

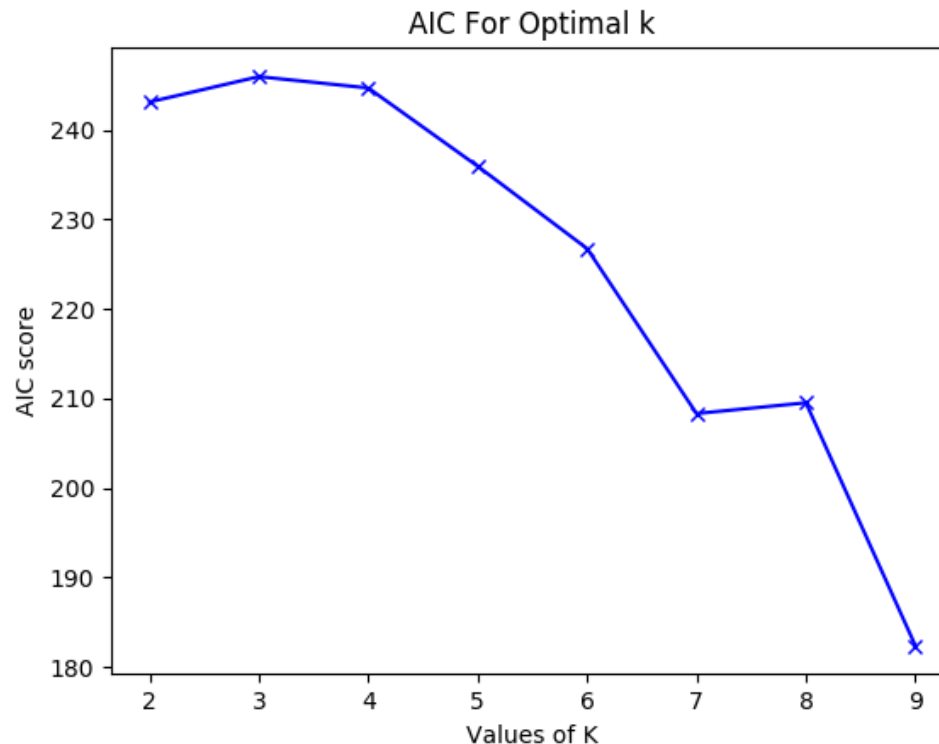
```
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.mixture import GaussianMixture #Import GMM module
data = pd.read_csv('Data\\kmeans_data1.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
K = range(2,11); aic,bic = [],[]
for num_clusters in K :
    gmm = GaussianMixture(n_components=num_clusters).fit(data) #perform EM clustering with number of clusters = 2
    aic.append(gmm.aic(data))
    bic.append(gmm.bic(data))

plt.figure();
plt.plot(K,aic,'bx-')
plt.xlabel('Values of K')
plt.ylabel('AIC score')
plt.title('AIC For Optimal k')

plt.figure();
plt.plot(K,bic,'bx-')
plt.xlabel('Values of K')
plt.ylabel('BIC score')
plt.title('BIC For Optimal k')
```

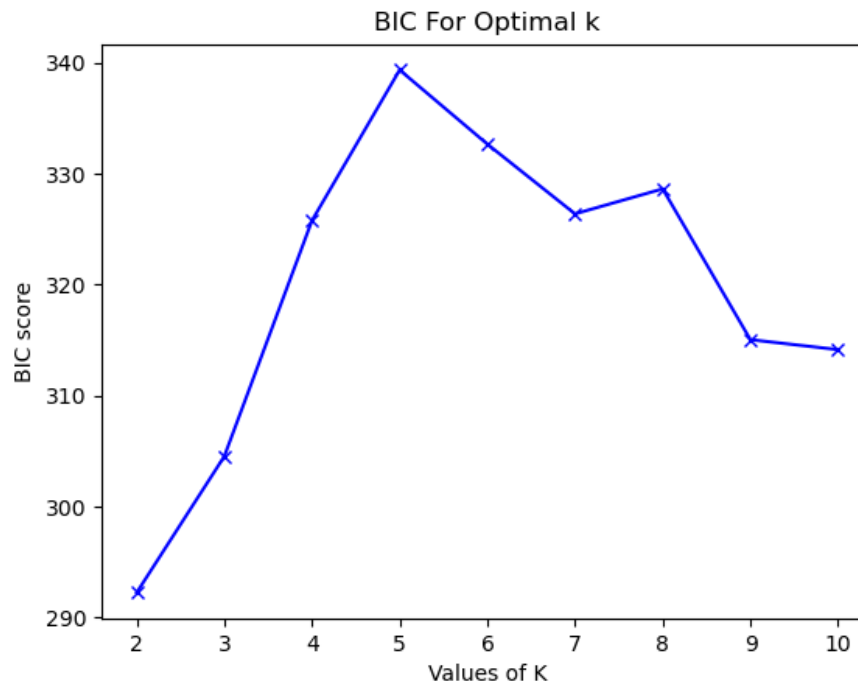
Example 1 - Question

- The AIC scores are shown below.
- However, the scores reach minimum at $K = 9$. This is because the clusters are not well-separated.



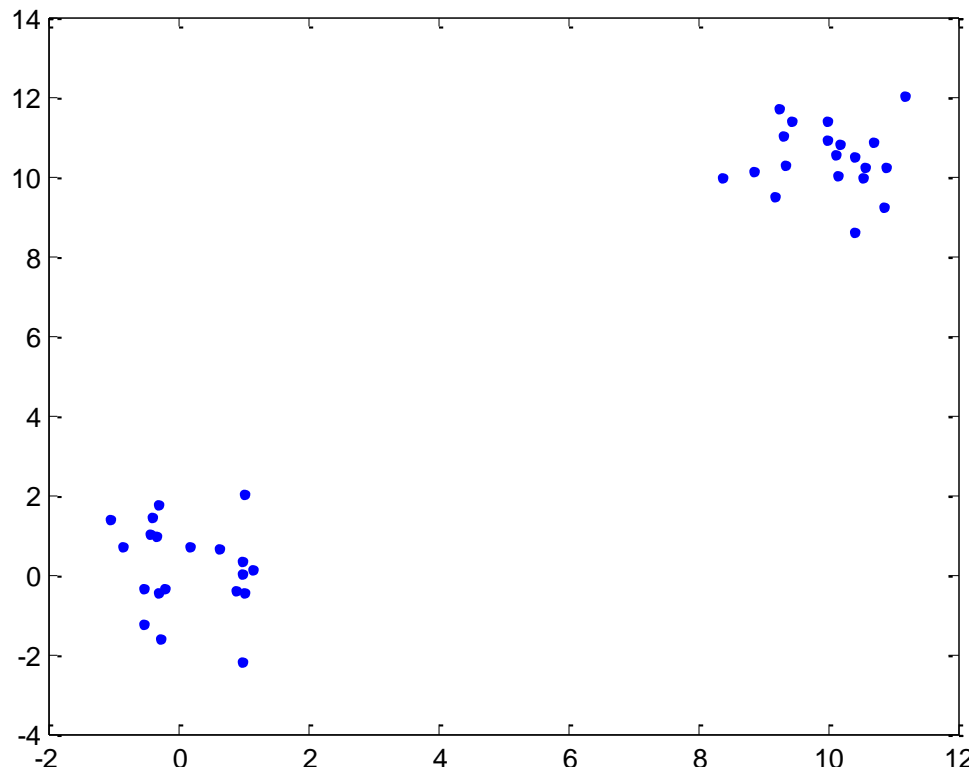
Example 1 - Question

- The minimum of BIC scores reach at $K = 2$.
- Again, this is because the clusters are not well-separated.
- AIC and BIC have disagreement. So, we cannot draw any conclusion. But in this case, BIC is better.



Example 2 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data2.csv).
- Apply GMM clustering algorithm to partition the data into different clusters using AIC and BIC methods.



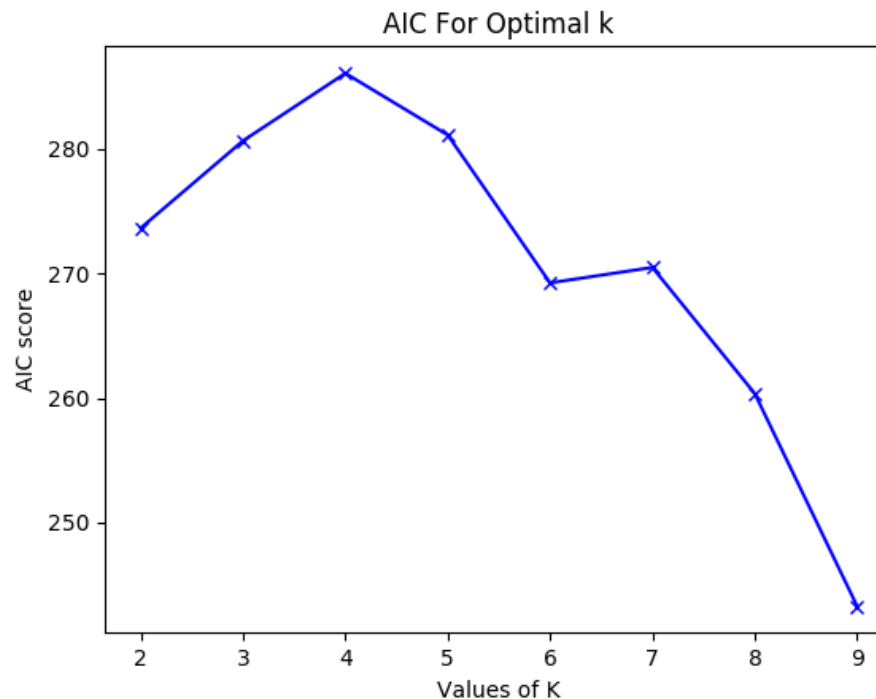
#Example 2

```
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.mixture import GaussianMixture #Import GMM module
data = pd.read_csv('Data\\kmeans_data2.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
K = range(2,11); aic,bic = [],[]
for num_clusters in K :
    gmm = GaussianMixture(n_components=num_clusters).fit(data) #perform EM
    clustering with number of clusters = 2
    aic.append(gmm.aic(data))
    bic.append(gmm.bic(data))
plt.figure();
plt.plot(K,aic,'bx-')
plt.xlabel('Values of K')
plt.ylabel('AIC score')
plt.title('AIC For Optimal k')

plt.figure();
plt.plot(K,bic,'bx-')
plt.xlabel('Values of K')
plt.ylabel('BIC score')
plt.title('BIC For Optimal k')
```

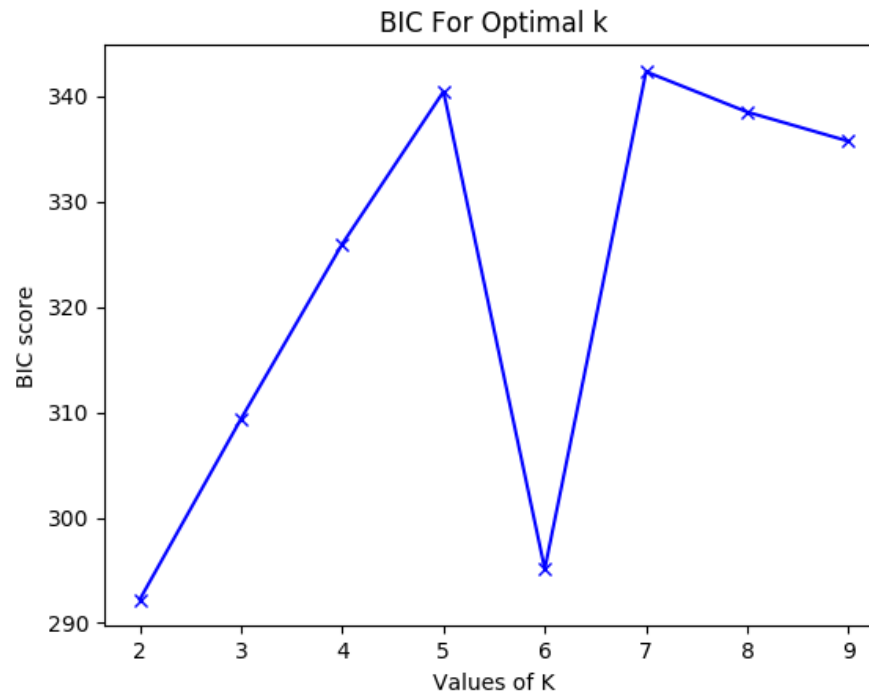
Example 2 - Question

- The AIC scores are shown as below.
- The minimum is reached at $K = 9$.
- The method is not very accurate even for well-separated clusters.



Example 2 - Question

- The BIC scores are shown as below.
- The minimum is reached at $K = 2$. There are two clusters.
- In the data, the two clusters are well-separated.
- Again, AIC and BIC have disagreement. But BIC is better.



Example 3 - Question

- Cluster the Iris dataset with GMM clustering algorithm to partition the data into different clusters using AIC and BIC methods.
- There are 3 groups with 150 samples and 4 attributes.
- The three groups are Iris-setosa, Iris-versicolor and Iris-virginica.
- Suppose we do not know that the dataset has 3 groups.

Example 3 - Question

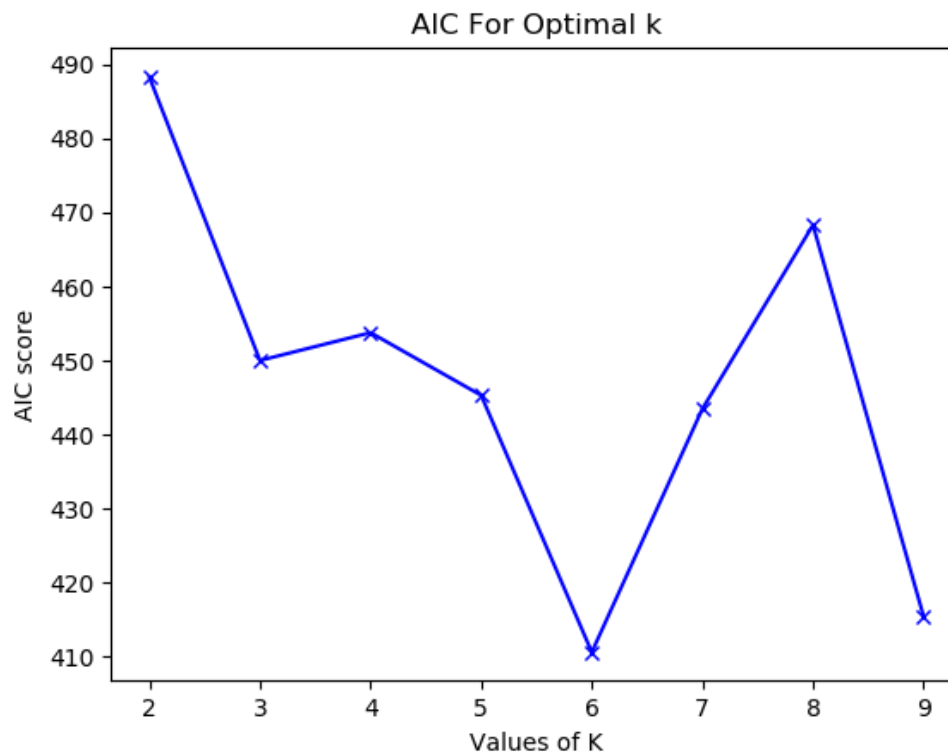
```
#Example 3
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.mixture import GaussianMixture #Import GMM module
data = pd.read_csv('Data\\iris.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
K = range(2,11); aic,bic = [],[]
for num_clusters in K :
    gmm = GaussianMixture(n_components=num_clusters).fit(data.iloc[:, :-1]) #perform EM clustering with number of clusters = 2
    aic.append(gmm.aic(data.iloc[:, :-1]))
    bic.append(gmm.bic(data.iloc[:, :-1]))

plt.figure();
plt.plot(K,aic,'bx-')
plt.xlabel('Values of K')
plt.ylabel('AIC score')
plt.title('AIC For Optimal k')

plt.figure();
plt.plot(K,bic,'bx-')
plt.xlabel('Values of K')
plt.ylabel('BIC score')
plt.title('BIC For Optimal k')
```

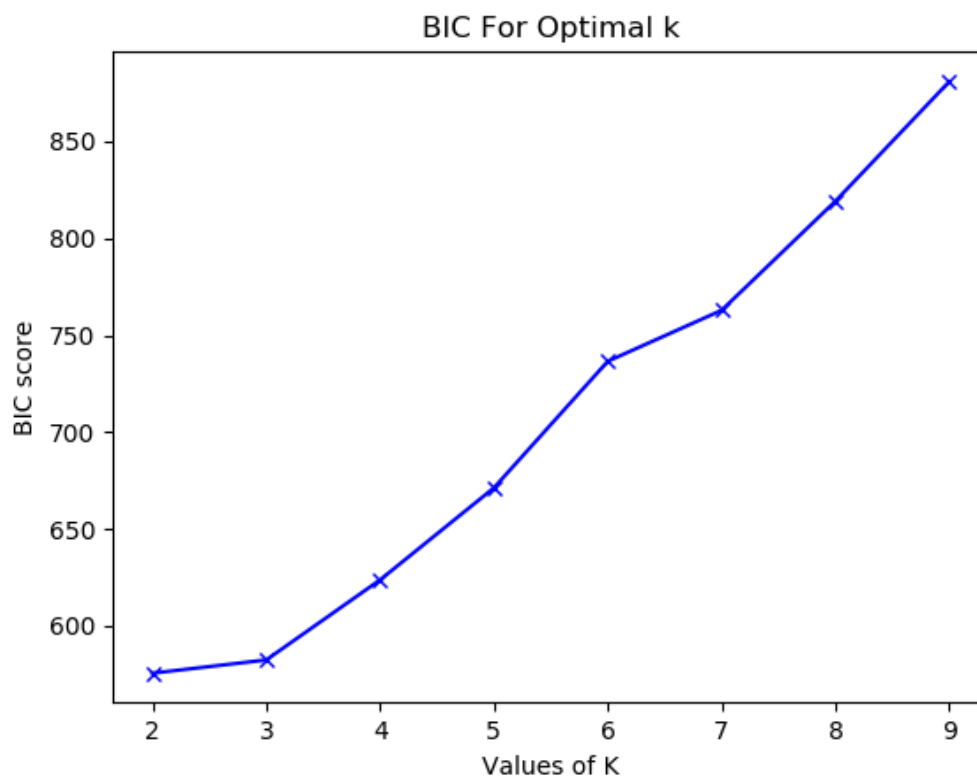
Example 3 - Question

- AIC shows that there are totally six clusters.



Example 3 - Question

- The BIC shows that there are two clusters.
- AIC and BIC have disagreement. BIC has a better result.



AIC and BIC

- In the above three examples, AIC and BIC have disagreements. This is usually the case in real-world data.
- If we really want to draw a conclusion using AIC and BIC, we may use the following rules:
- If the sample size is large, we consider the result of AIC.
- If the sample size is small, we consider the result of BIC.
- Actually, in the above three examples, the sample sizes are relatively small. The BIC performs better.

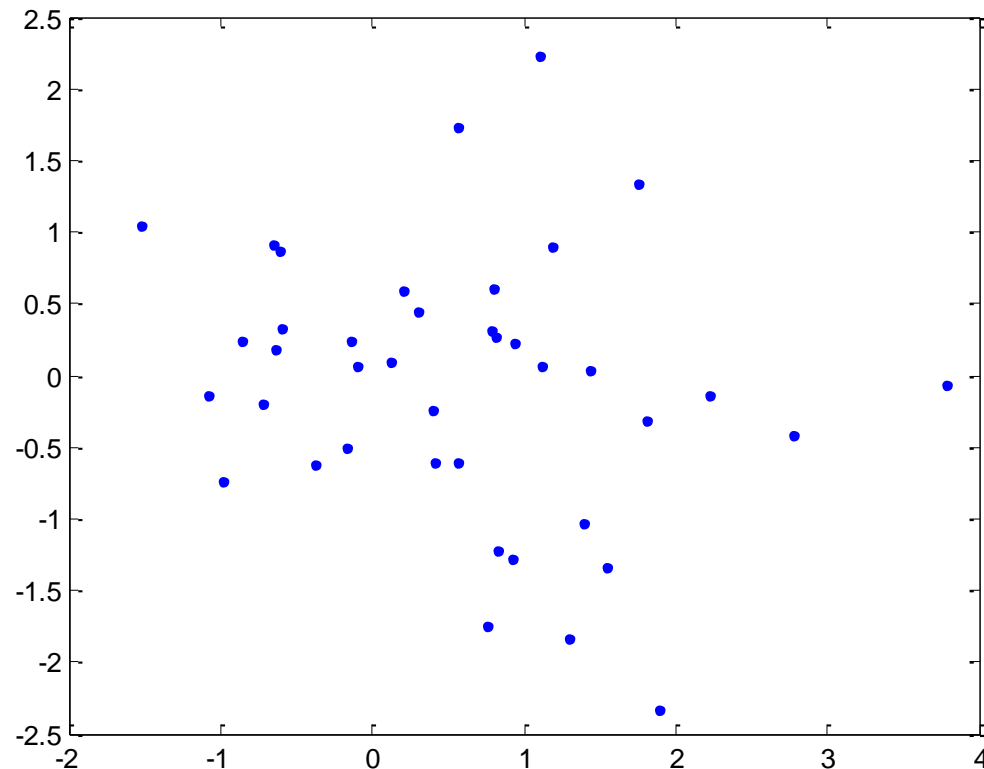
HIERARCHICAL CLUSTERING

Hierarchical clustering

- After having hierarchy of the data, we can find the distances among different clusters. The largest distance is usually treated as the “threshold”. This can help determine the number of clusters.

Example 1 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data1.csv).
- Apply hierarchical clustering algorithm to partition the data into different clusters.



Example 1 - Question

65

```
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
import scipy.cluster.hierarchy as shc
data = pd.read_csv('Data\\kmeans_data1.csv') #Load the data file
plt.figure(); #Plot the figure
plt.plot(data.iloc[:,0],data.iloc[:,1],'.');
```

##Visualize the data

```
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
data = pd.read_csv('Data\\kmeans_data1.csv') #Load the data file
plt.figure(); #Plot the figure
plt.plot(data.iloc[:,0],data.iloc[:,1],'.');
```

#Example 1 - Single link

```
plt.figure(); #Plot the figure
dend = shc.dendrogram(shc.linkage(data, method='single'))
plt.title('Single Link')
```

#Example 1 - Complete link

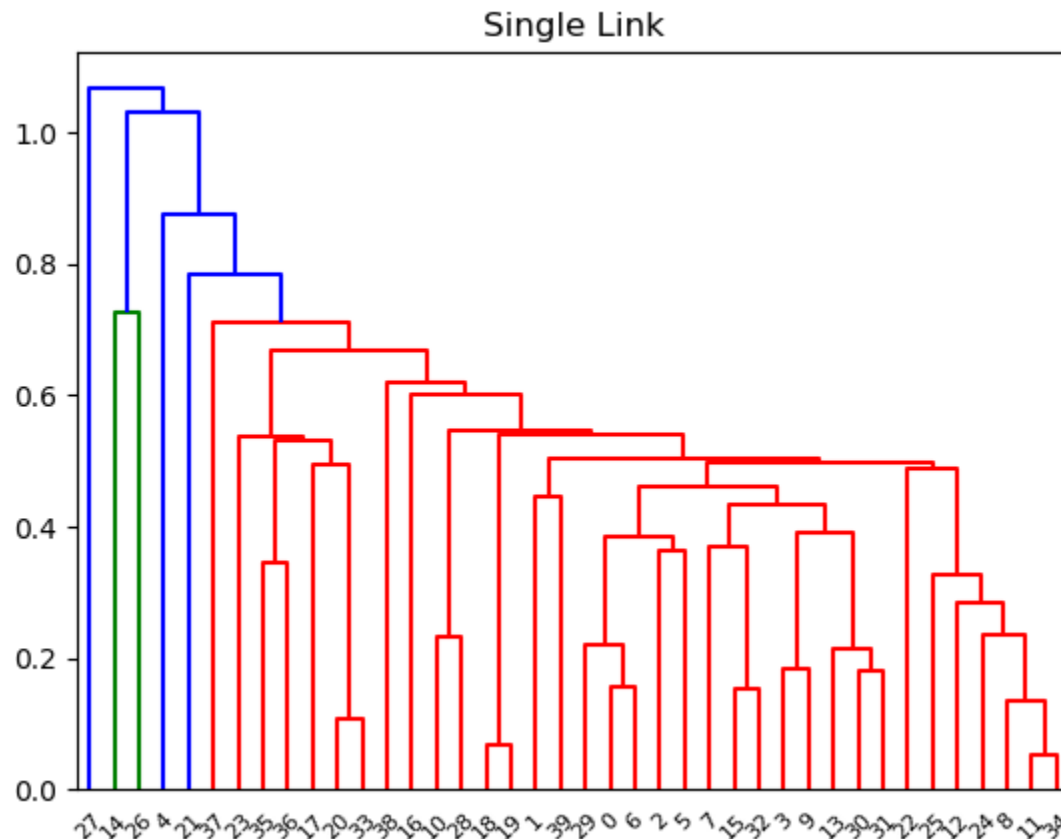
```
plt.figure(); #Plot the figure
dend = shc.dendrogram(shc.linkage(data, method='complete'))
plt.title('Complete Link')
```

#Example 1 - Average link

```
plt.figure(); #Plot the figure
dend = shc.dendrogram(shc.linkage(data, method='average'))
plt.title('Average Link')
```

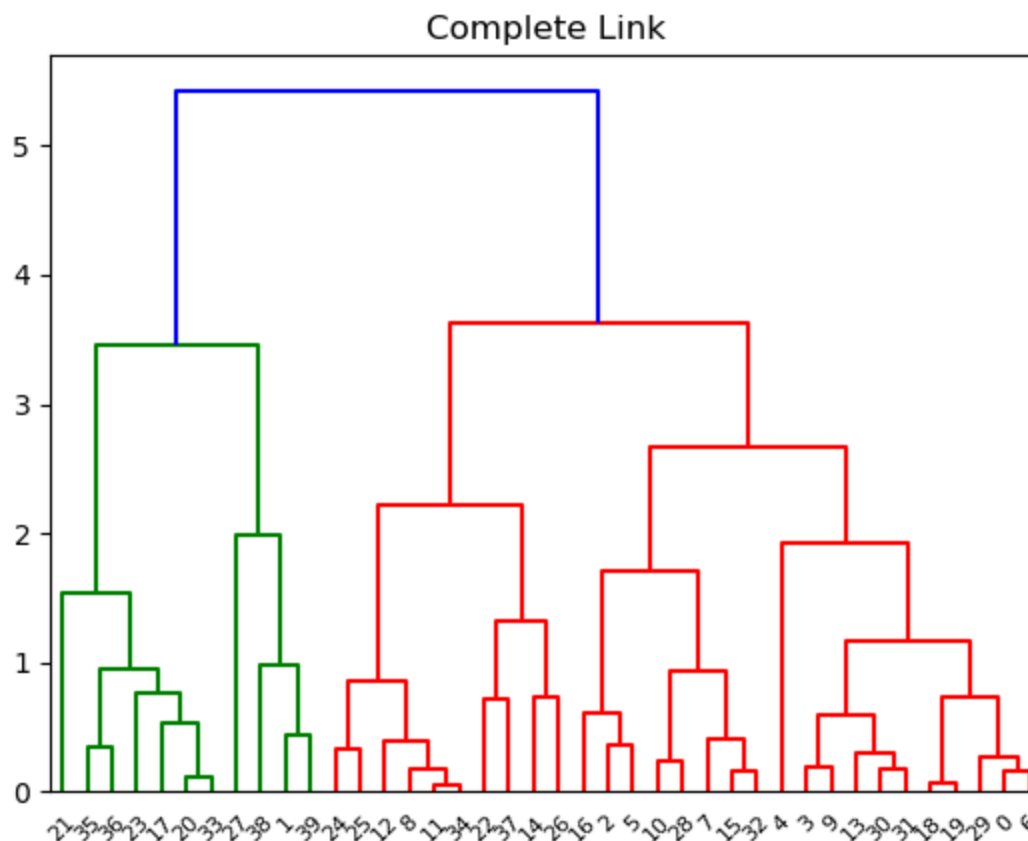
Example 1 - Question

- Results of single link. Hard to determine the number of clusters.



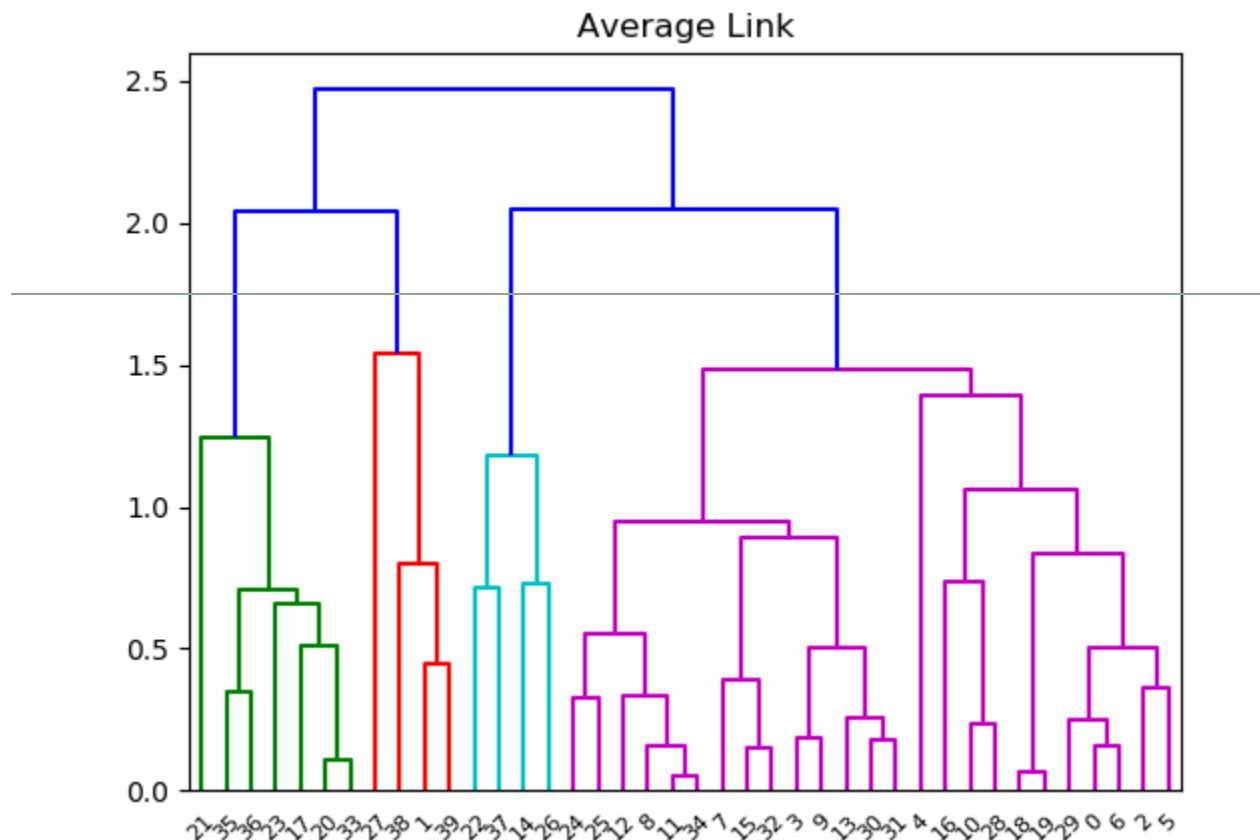
Example 1 - Question

- Results of complete link. There are two clusters.



Example 1 - Question

- Results of average link. There should be four clusters.



Example 2 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data2.csv).
- Apply hierarchical clustering algorithm to partition the data into two different clusters.

```
#Example 2 - Single link
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
import scipy.cluster.hierarchy as shc
data = pd.read_csv('Data\\kmeans_data2.csv') #Load the data file
plt.figure(); #Plot the figure
plt.plot(data.iloc[:,0],data.iloc[:,1],'.');
```

```
##Visualize the data
plt.figure(); #Plot the figure
dend = shc.dendrogram(shc.linkage(data, method='single'))
plt.title('Single Link')
```

```
#Example 2 - Complete link
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
data = pd.read_csv('Data\\kmeans_data2.csv') #Load the data file
plt.figure(); #Plot the figure
plt.plot(data.iloc[:,0],data.iloc[:,1],'.');
```

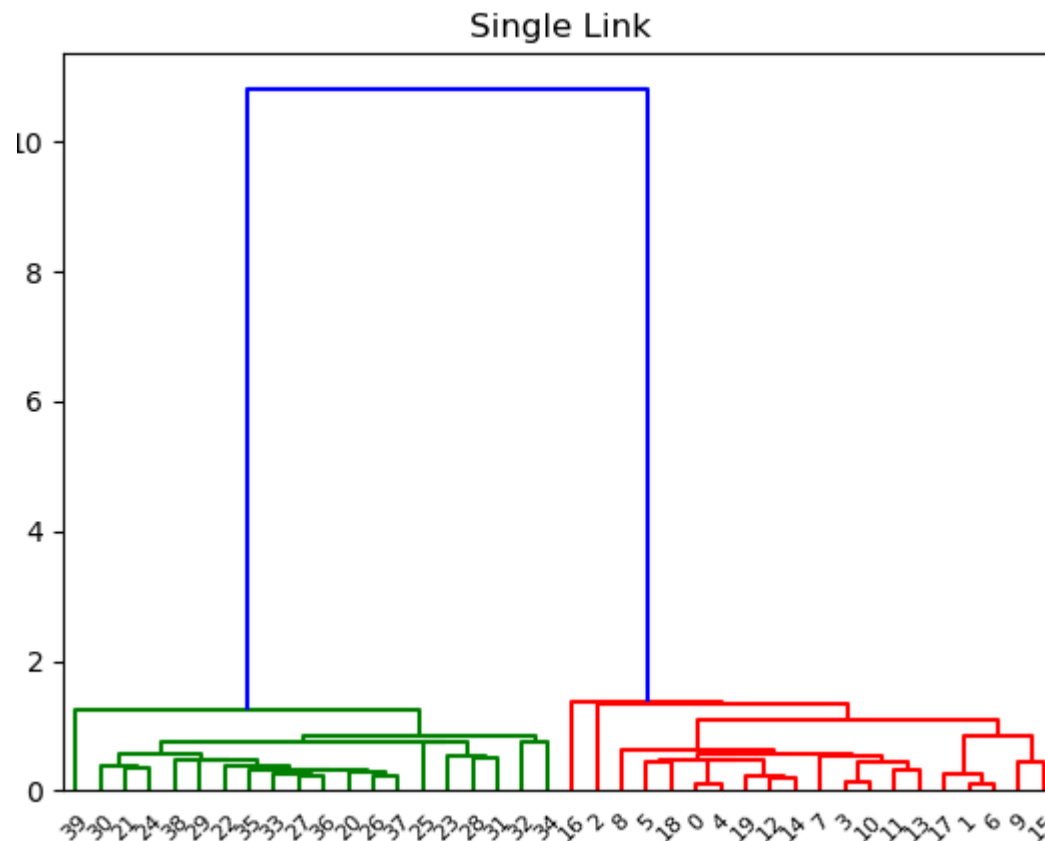
```
##Visualize the data
plt.figure(); #Plot the figure
dend = shc.dendrogram(shc.linkage(data, method='complete'))
plt.title('Complete Link')
```

```
#Example 2 - Average link
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
data = pd.read_csv('Data\\kmeans_data2.csv') #Load the data file
plt.figure(); #Plot the figure
plt.plot(data.iloc[:,0],data.iloc[:,1],'.');
```

```
##Visualize the data
plt.figure(); #Plot the figure
dend = shc.dendrogram(shc.linkage(data, method='average'))
plt.title('Average Link')
```

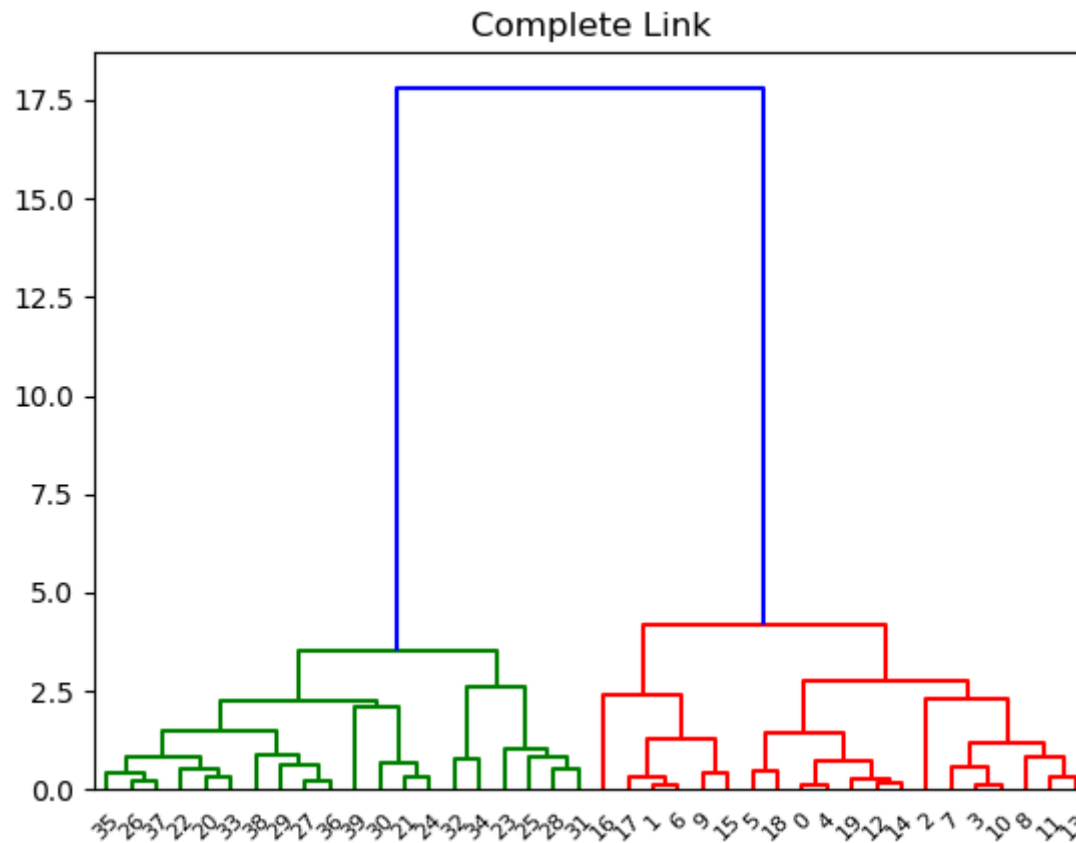
Example 2 - Question

- Results of single link. There are two clusters.



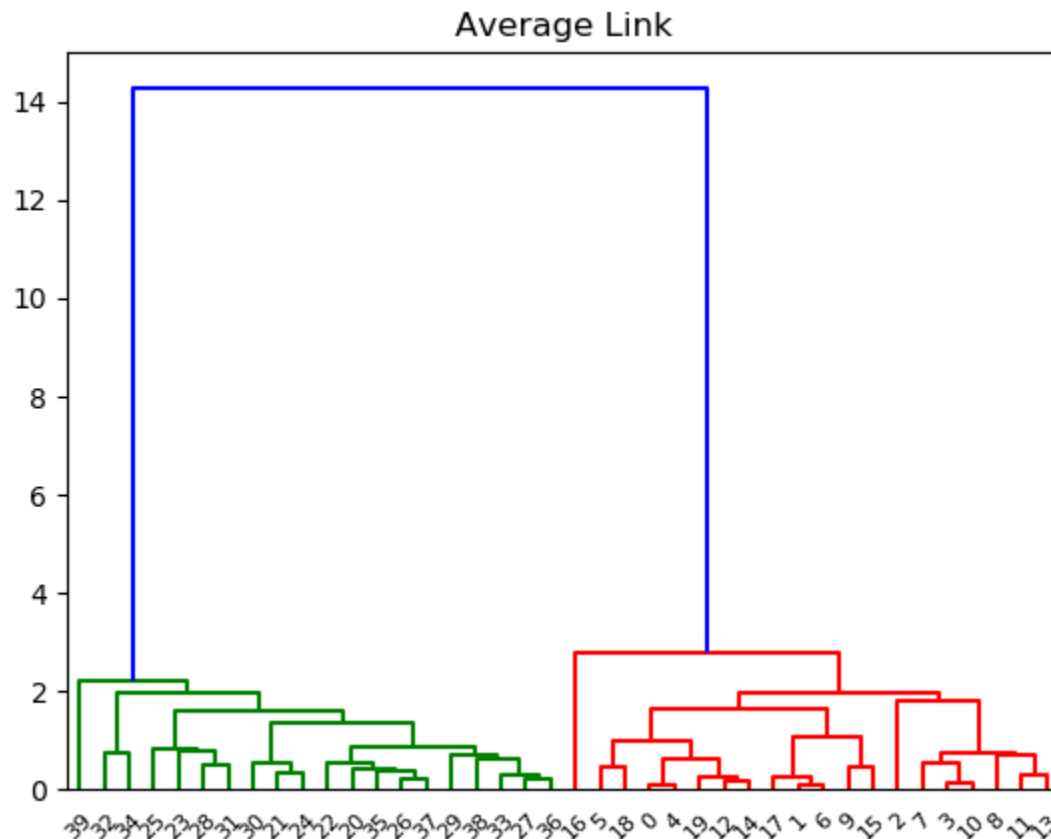
Example 2 - Question

- Results of complete link. There are two clusters.



Example 2 - Question

- Results of average link. There are two clusters.



Example 2 - Question

- Hierarchical clustering can successfully identify two clusters. The clustering results are good as well.
- This is because the two clusters are well-separated.

Example 3 - Question

- Cluster the Iris dataset with hierarchical clustering algorithm.
- There are 3 groups with 150 samples and 4 attributes.
- The three groups are Iris-setosa, Iris-versicolor and Iris-virginica.
- Suppose we do not know that the dataset has 3 groups.

Example 3 - Question

76

```
#Example 3 - Single link
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
import scipy.cluster.hierarchy as shc
data = pd.read_csv('Data\iris.csv') #Load the data file
```

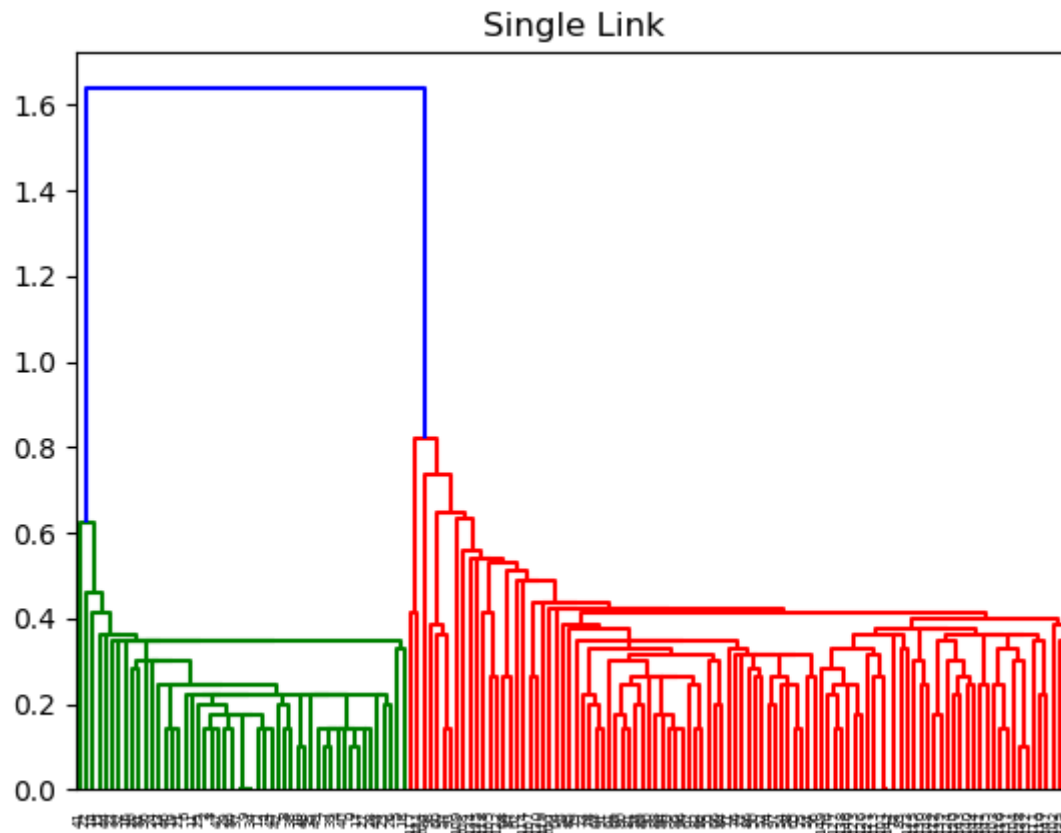
```
#Example 3 - Single link
plt.figure(); #Plot the figure
dend = shc.dendrogram(shc.linkage(data.iloc[:, :-1], method='single'))
plt.title('Single Link')
```

```
#Example 3 - Complete link
plt.figure(); #Plot the figure
dend = shc.dendrogram(shc.linkage(data.iloc[:, :-1], method='complete'))
plt.title('Complete Link')
```

```
#Example 3 - Average link
plt.figure(); #Plot the figure
dend = shc.dendrogram(shc.linkage(data.iloc[:, :-1], method='average'))
plt.title('Average Link')
```

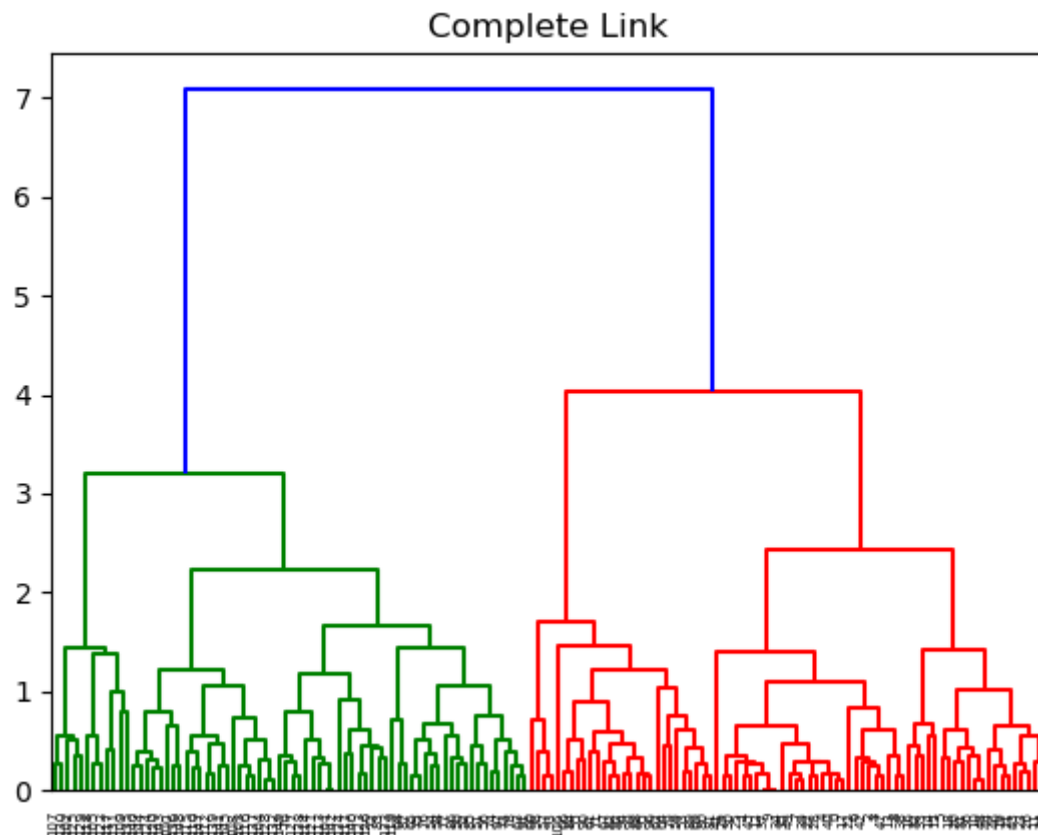
Example 3 - Question

- Results of single link. There are two clusters.



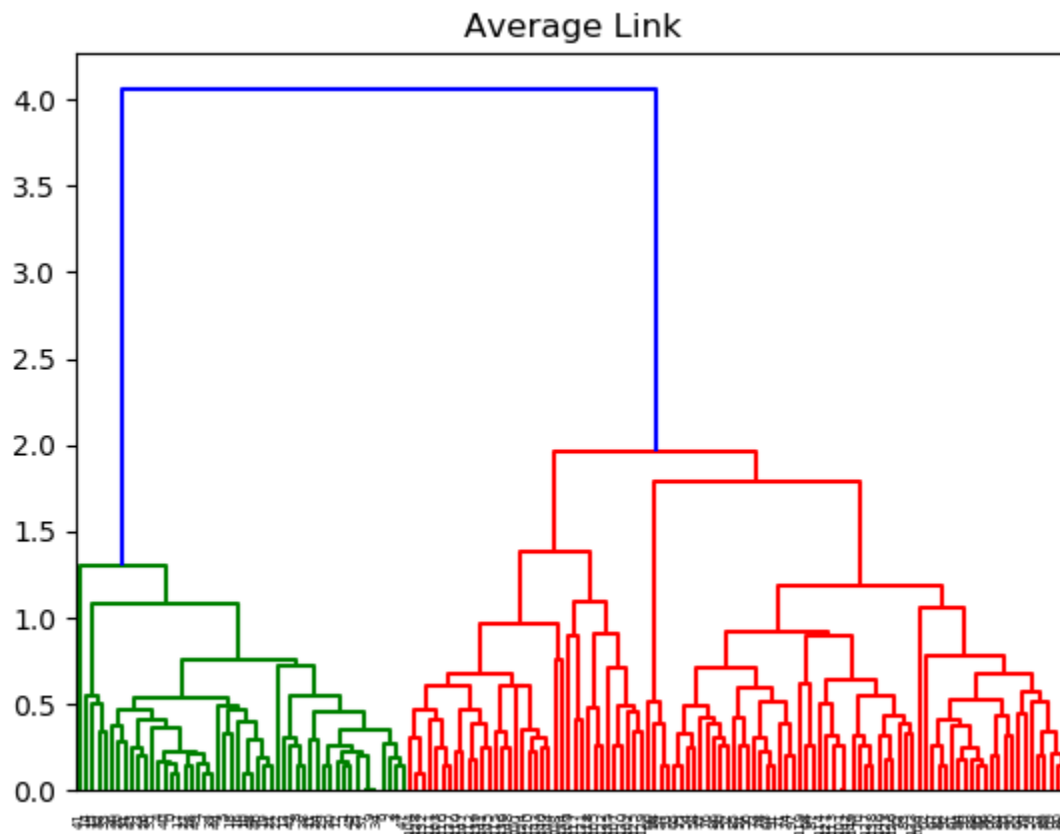
Example 3 - Question

- Results of complete link. There are two clusters.



Example 3 - Question

- Results of average link. There are two clusters.



Example 3 - Question

- Again, for iris dataset, although there are three groups, only two clusters are identified.
- This is because two groups of the datasets are closed to each other.

Remark

- Determine the number of clusters is a challenging problem.
- Although different methods have been proposed, there is no single method that can determine the number of clusters well in all cases.
- To determine the number of clusters, we have to understand the data well. Sometimes, we need extensive labour work (have to visualize the data) and determine the number clusters.