Silesian
University
of Technology

CLASSIFIERS LABORATORY REPORT

Data Clustering I

Authors:

Jean Remy UWACU
Maurice NGABONZIZA
Ankit RATHI

Introduction

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

Broadly speaking, clustering can be divided into two subgroups:

- **Hard Clustering:** In hard clustering, each data point either belongs to a cluster completely or not.
- **Soft Clustering**: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned.

K-means is a simple unsupervised machine learning algorithm that groups a dataset into a user-specified number (*k*) of clusters. The algorithm is somewhat naive--it clusters the data into *k* clusters, even if *k* is not the right number of clusters to use. Therefore, when using k-means clustering, users need some way to determine whether they are using the right number of clusters.

One method to validate the number of clusters is the **elbow method**. The idea of the elbow method is to run k-means clustering on the dataset for a range of values of *k* (say, *k* from 1 to 10 in the examples above), and for each value of *k* calculate the sum of squared errors (**SSE**).

Clustering and K-means method

As we have seen, Clustering is a technique that involves the grouping of data points. It is a process which partitions a given data set into homogeneous groups based on given features such that similar objects are kept in a group whereas dissimilar objects are in different groups. Given a set of data points, we can use a clustering algorithm to classify each data point into a specific group. In theory, data points that are in the same group should have similar properties and/or features, while data points in different groups should have highly dissimilar properties and/or features. Clustering is a method of unsupervised learning and is a common technique for statistical data analysis used in many fields.

We can use clustering analysis to gain some valuable insights from our data by seeing what groups the data points fall into when we apply a clustering algorithm.

K-Means Clustering

K-Means is probably the most well know clustering algorithm. It's taught in a lot of introductory data science and machine learning classes.

1.To begin, we first select a number of classes/groups to use and randomly initialize their respective center points. To figure out the number of classes to use, it's good to take a quick look at the data and try to identify any distinct groupings.

2.Each data point is classified by computing the distance between that point and each group center, and then classifying the point to be in the group whose center is closest to it.

3.Based on these classified points, we recompute the group center by taking the mean of all the vectors in the group.

4. Repeat these steps for a set number of iterations or until the group centers don't change much between iterations. We can also opt to randomly initialize the group centers a few times, and then select the run that looks like it provided the best results.

K-Means has the advantage that it is pretty fast, as all we are really doing is computing the distances between points and group centers; very few computations! It thus has a linear complexity $O(n)$.

On the other hand, K-Means has a couple of disadvantages. Firstly, you have to select how many groups/classes there are. This is not always trivial and ideally with a clustering algorithm we had want it to figure those out for us because the point of it is to gain some insight from the data. K-means also starts with a random choice of cluster centers and therefore it may yield different clustering results on different runs of the algorithm. Thus, the results may not be repeatable and lack consistency.

How we will proceed:

- We are going to generate synthetic datasets and we'll use the k-means method on them.
- We'll implement the elbow method to determine k of k-means.
- Elbow method: to find the optimal number of clusters to split the data, at the percentage of variance explained as a function of the number of clusters (one should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data).

Using this calculation:

$$W(k) = \sum_{i=1}^{k} \sum_{j=1}^{m} (X_{ij} - z_i)^2$$

Using the Elbow method

The basic idea of k-means clustering, is to define clusters such that the total intra-cluster variation [or total within-cluster sum of square (WSS)] is minimized. The total WSS measures the compactness of the clustering and we want it to be as small as possible.

The Elbow method looks at the total WSS as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't improve much better the total WSS.

The optimal number of clusters can be defined as follow:

1. Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, by varying k from 1 to 10 clusters.
2. For each k, calculate the total within-cluster sum of square (wss).
3. Plot the curve of wss according to the number of clusters k.
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

1. Plotting the Elbow Point for synthetic dataset

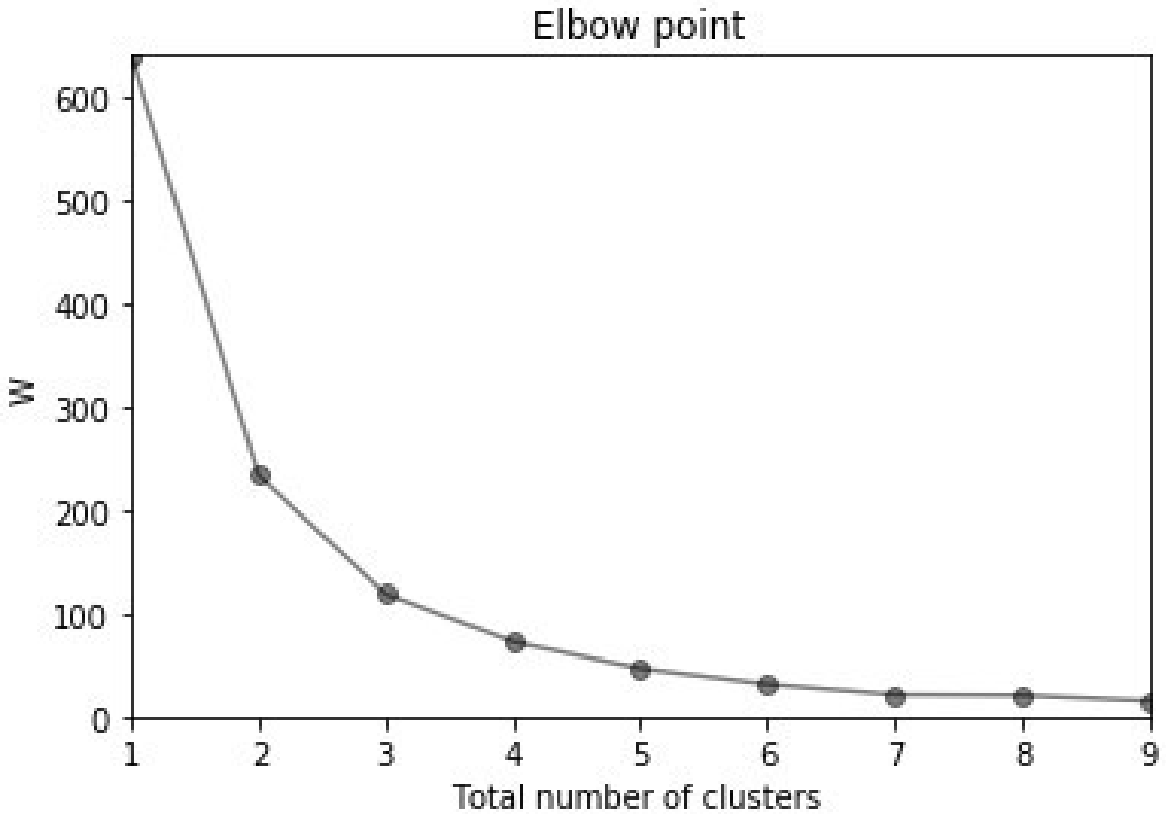

Fig. 1 The Plot of the Elbow point for Artificial dataset

2. Plotting the Elbow Point for real data

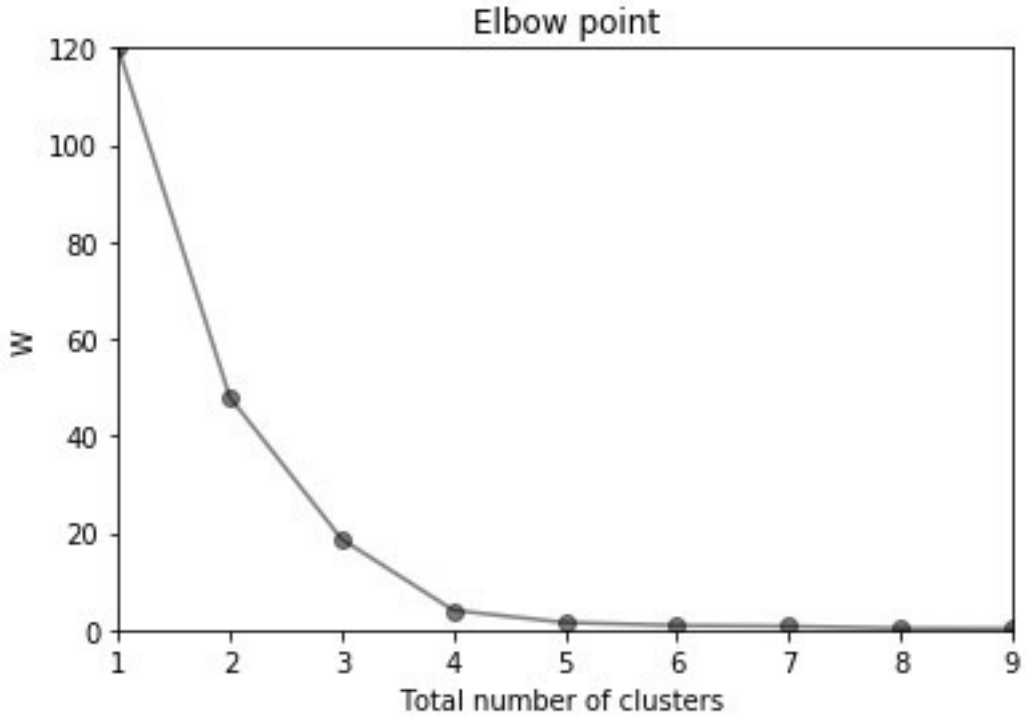

Fig. 2 The plot of the elbow point for X0, X1

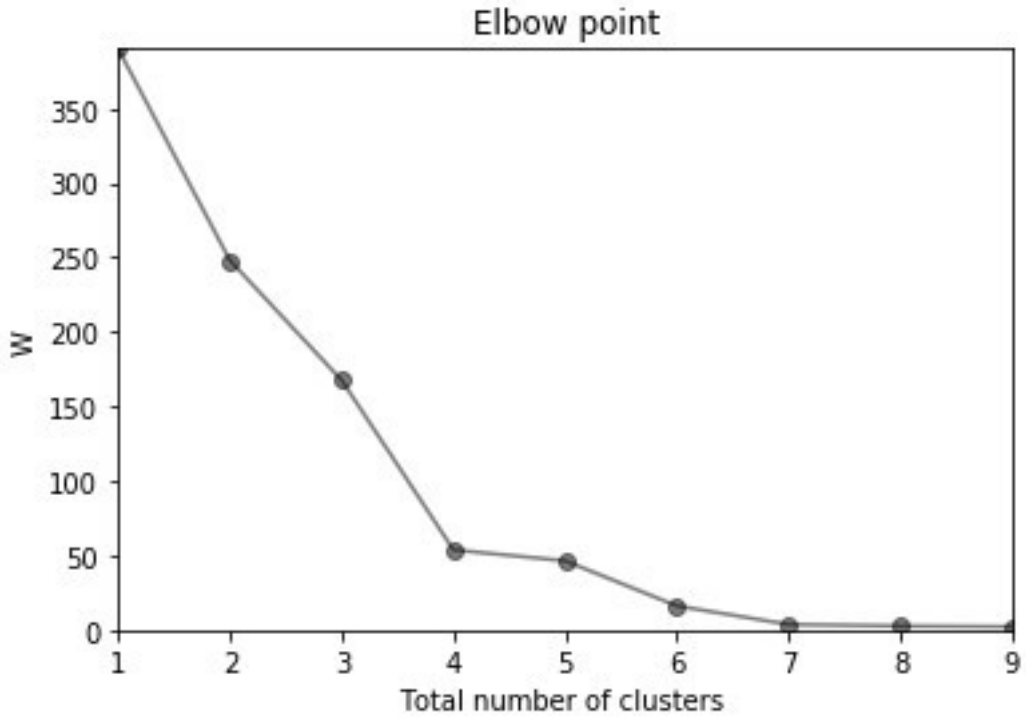

Fig. 3 The plot of the elbow point for X1, X2

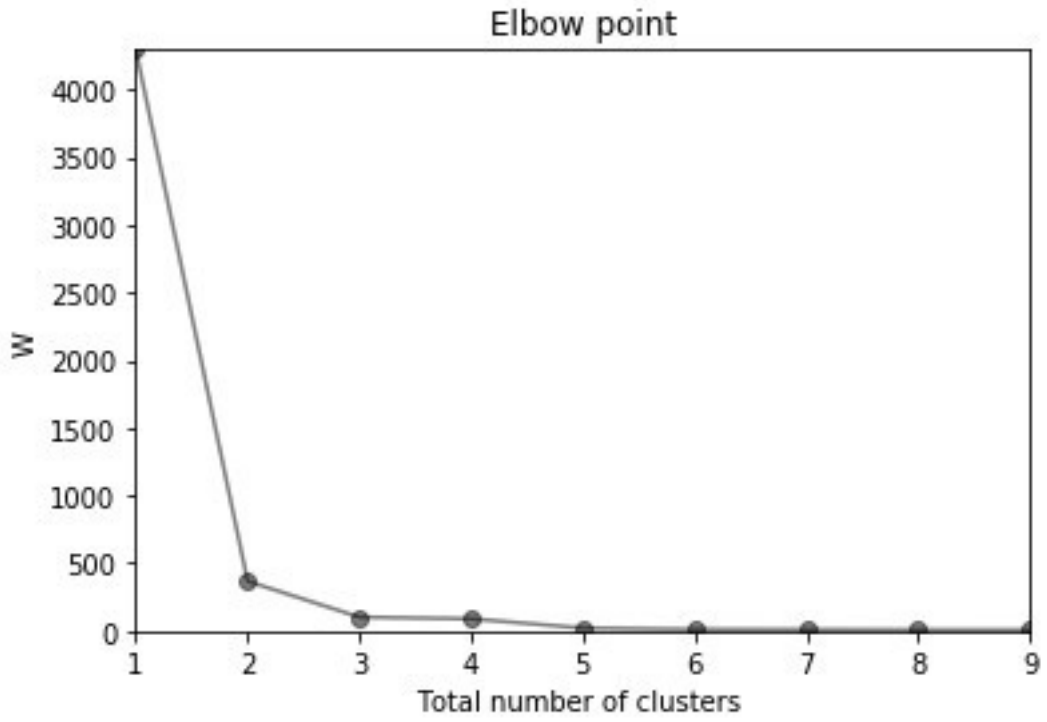

Fig. 4 The plot of the elbow point for X2, X3

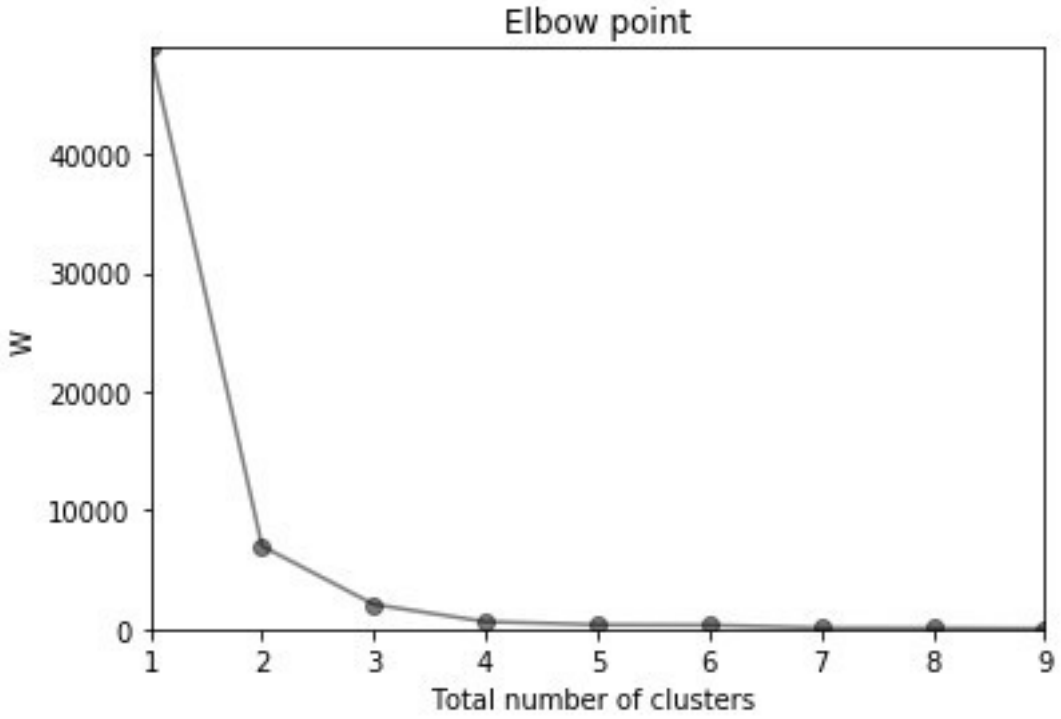

Fig. 5 The plot of the elbow point for X3, X4

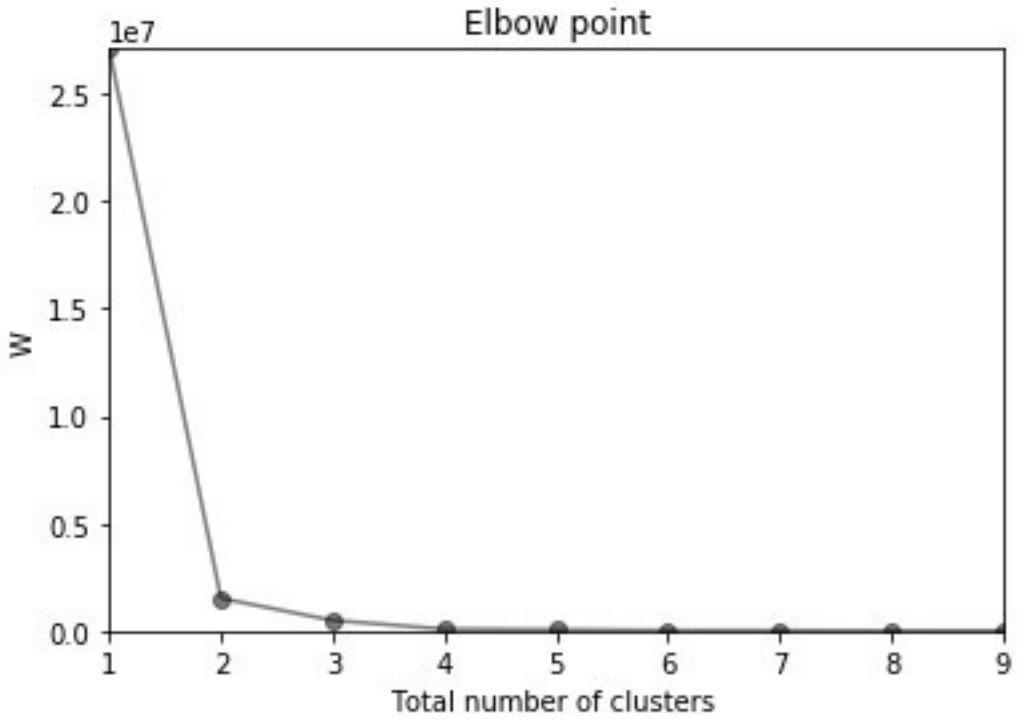

Fig. 6 The plot of the elbow point for X4, X5

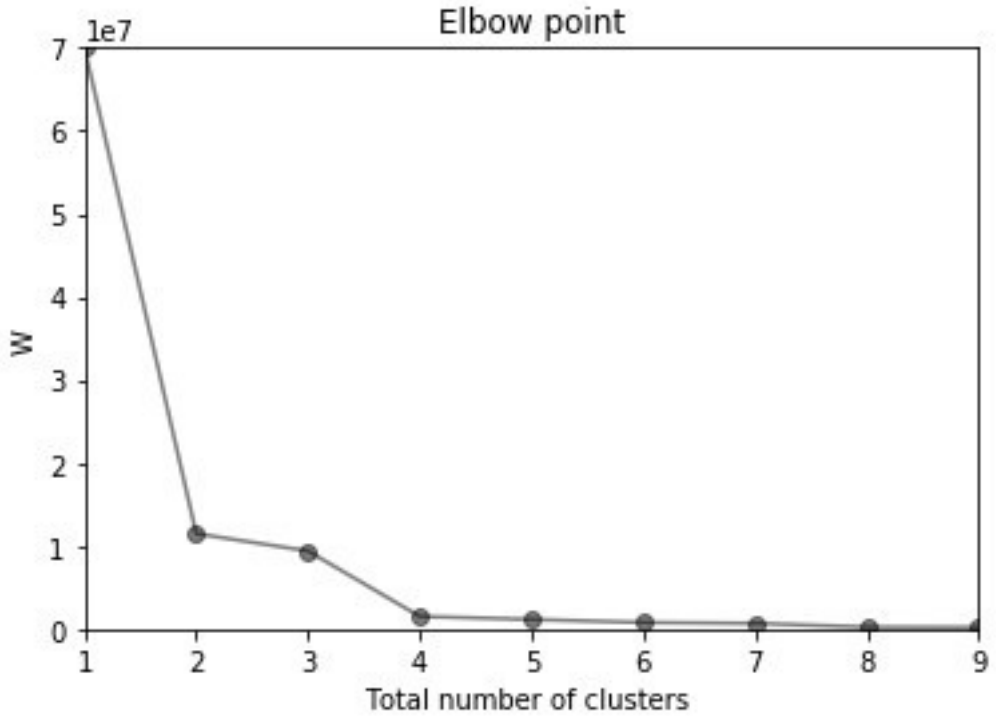

Fig. 7 The plot of the elbow point for X6, X7

Conclusion:

To resume we used K-means method following these 4 steps:

1. Specify the desired number of clusters K
2. Randomly assign each data point to a cluster
3. Compute cluster centroids
4. Re-assign each point to the closest cluster centroid

The Elbow method helped us to find appropriate number of cluster for our dataset. To find appropriate number of clusters we used following coefficient:

$$W(k) = \sum_{i=1}^{k} \sum_{j=1}^{m} (X_{ij} - z_i)^2$$

Basically this method is based on to find the smallest distance between vectors of cluster which is represent by k and centroids of these clusters. We calculate these distances over k time to check which number of cluster would be the most properly. Elbow point representing the appropriate number of clusters in a dataset.

Overview of the Code in Python:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import matplotlib.ticker as plticker
import csv
import math

### Artificial Data
###A Sample of dataset
mean_class_1 = [-10, 10]
covariance_class_1 = [[4, 0], [0, 4]]  # diagonal covariance-class 1

mean_class_2 = [5,20]
covariance_class_2 = [[4, 0], [0,4]]  # diagonal covariance-class 2

mean_class_3 = [50,50]
covariance_class_3 = [[4, 0], [0,4]]  # diagonal covariance-class 2

class_1=np.random.multivariate_normal(mean_class_1, covariance_class_1, 100)
class_2=np.random.multivariate_normal(mean_class_2, covariance_class_2, 100)
class_3=np.random.multivariate_normal(mean_class_3, covariance_class_3, 100)
dataset_class_one=pd.DataFrame({'X1':class_1[:,0],'X2':class_1[:,1]})
dataset_class_two=pd.DataFrame({'X1':class_2[:,0],'X2':class_2[:,1]})
#dataset_class_three=pd.DataFrame({'X1':class_3[:,0],'X2':class_3[:,1]})
dataset=dataset_class_one.append(dataset_class_two)
#dataset = dataset.append(dataset_class_three)

dataset = dataset.iloc[:, :].values
#####

def calculate_elbow(k, matrix_X, centers):
    sum_clusters=0
    vector_distance =0;
    for k in range(k):
        centroid = centers[k]
        for i in range(len(matrix_X[k])):
            for j in range(2):
                vector_distance += (matrix_X[k][i][j] -
            centroid[j])**2
            sum_clusters+=(vector_distance**1/2)**2
            vector_distance = 0
    print(sum_clusters)
    return sum_clusters
```

```
def calculate_coordinates(n, data):
    x = []
    y = []

    for number_of_cluster in range(1, n, 1):
        Kmean =
        KMeans(n_clusters=number_of_cluster)
        Kmean.fit(data)

        centers = Kmean.cluster_centers_
        cluster_number = Kmean.labels_

        matrix_X = np.array([[]]*number_of_cluster)
        matrix_X = matrix_X.tolist()
        for i in range(len(data)):
            matrix_X[cluster_number[i]].append(data[i])

        x.append(number_of_cluster)
        y.append(calculate_elbow(number_of_cluster, matrix_X, centers))
#       y[0] = 50000

    return x, y, y[0]

def draw_graph_of_elbow(data):
#    plt.clf()

    x_axis, y_axis, limit = calculate_coordinates(10, data)
    plt.ylim(0, math.ceil(limit))
    plt.xlim(1, )
    plt.plot(x_axis, y_axis, '-ok', color='black', alpha=0.5)
    plt.xticks(np.arange(min(x_axis), max(x_axis)+1,
    1.0)) plt.title("Elbow point")
    plt.xlabel("Total number of clusters ")
    plt.ylabel("W")
    plt.show()

### Artificial Data
draw_graph_of_elbow(dataset)

### Real Data
real_data = pd.read_csv('yeast.csv')
real_data = real_data.iloc[1:, :].values

list_of_real_data = []

def split_data():
    for index in range(real_data.shape[1]-1):
```

```
        dataset_real_data=pd.DataFrame({'X1':real_data[:, index],'X2':real_data[:,
        index+1]}) list_of_real_data.append(dataset_real_data)
        draw_graph_of_elbow(dataset_real_data.iloc[:, :].values)

split_data()
```