Department of Computer Science and Engineering

Islamic University of Technology (IUT)

A subsidiary organ of OIC

Quiz 04

CSE 4739: Data Mining

Name:Mazen Asag
Student ID:180041257
Section:B
Semester:7th
Academic Year:2022

Contents :

1 Introduction :

Clustering is the belong to unsupervised learning and this type of learning is consider as type of machine learning algorithm that aims to understand relationships within datasets, or classify data and make predictions without any training data or any human interactive , there are different algorithms of unsupervised learning also models used to do the task of clustering, and one of these algorithms is K-means .

k-means like the other clustering models that try to divide a population or set of data points into groups such that data points in the same group are more similar to data points in other groups than that data points in other groups are similar to one another.

K-Means :

statistical data analysis used to solve the problem of clustering in data mining and pattern recognition, and involves grouping several objects into a number of groups or clusters and finding similarities between them when we don't know their classes .

K-Means Algorithm Strategy:

Step 1: define the Number of Clusters K in entire data.

K is the clusters we will use to classify our data and start categorize and assign our data points to these clusters .

Step 2: Randomly choose Centroid Points .

Centroid is center point in each cluster the will be selected and used to measure the distance between it and the all points of data the distance will be computed using euclidean formula.

Step 3: Assigning of points to each k Clusters

we start by assigning the points by measuring the distance from each data point to each centroid in all clusters we defined .

if A-point is close to 3-centroid of the cluster number 3 then this point will belong to this cluster .

Step 4: choose a New Centroid of Each Cluster : we look for the new centroids for each cluster by computing the mean of x points and the mean the y points .

Step 5: Assign all points to the new clusters centroid : we repeat the same steps with the new centroid and re assign the points with minimum euclidean distance to each centroid.

Step 6 : Repeat 5 , 6 steps :

we repeat doing steps 5 and 6 until there will be no change .

Define the best K Cluster:

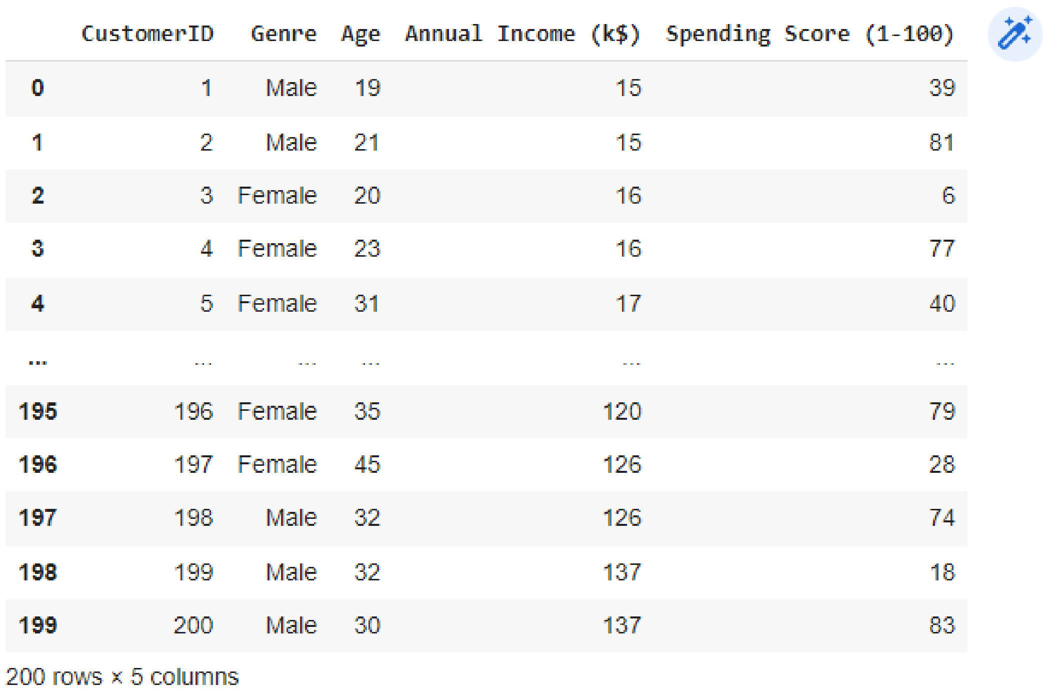
WCSS is method use to find the optimal number of clusters we can form for our data by compute the sum of squared distance between each point and the centroid in a cluster.

Dataset:

An interesting dataset containing hypothetical customer data that show information about customers with different features ,and the aim is to divide all this features of the customes into different categories , the dataset consist of 200 record with five different features .

Features of the Mall customers :

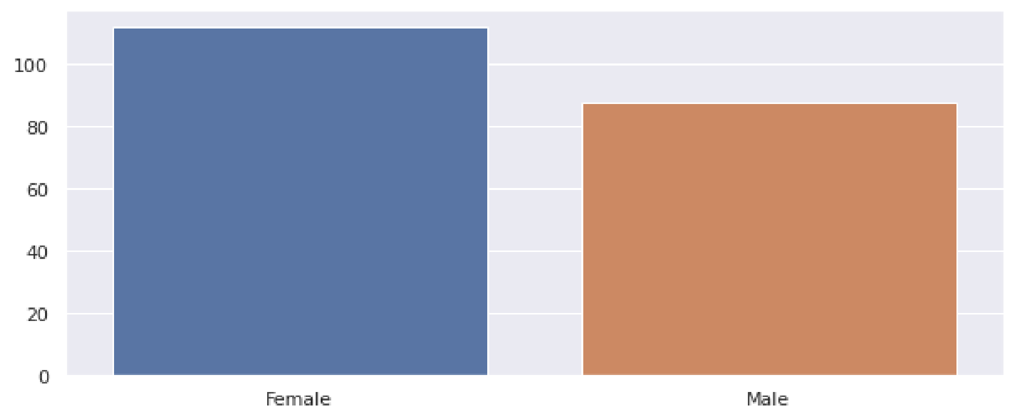
1. Customer ID
2. Customer Gender
3. Customer Age
4. Annual Income of the customer (in Thousand Dollars)
5. Spending score of the customer (based on customer behaviour and spending nature)



	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

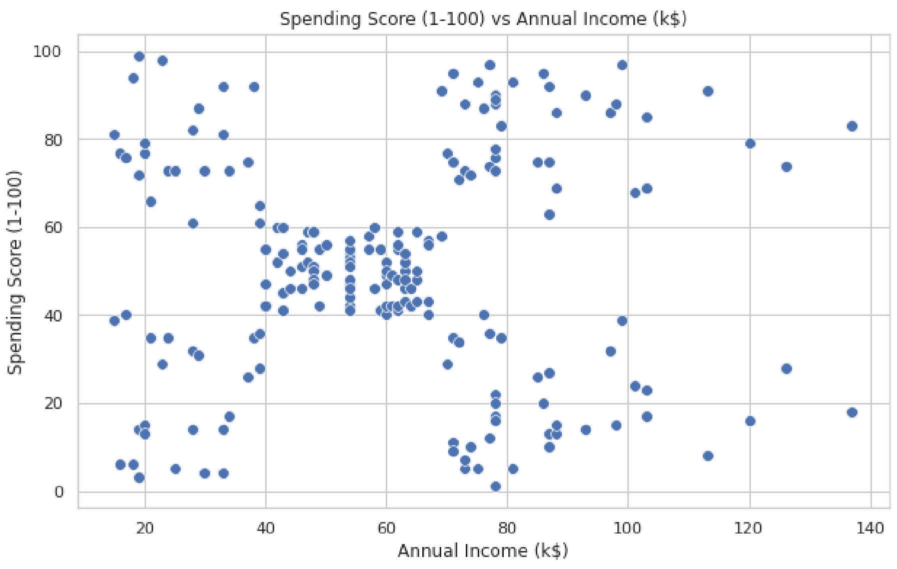
200 rows × 5 columns

For gender analysis it shows the there is more female customers than male in this dataset .

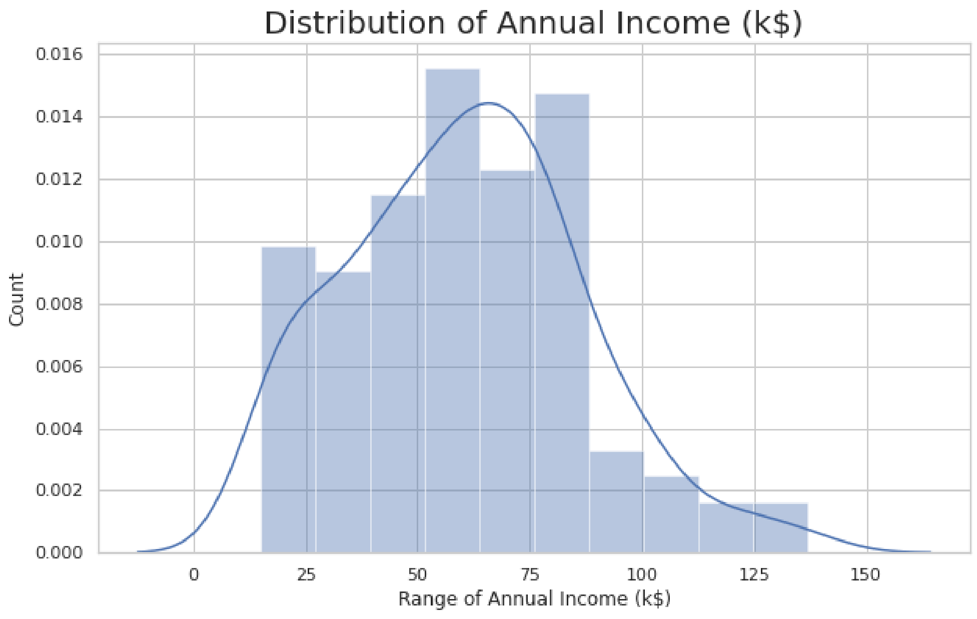


The spending score of the

Input data .

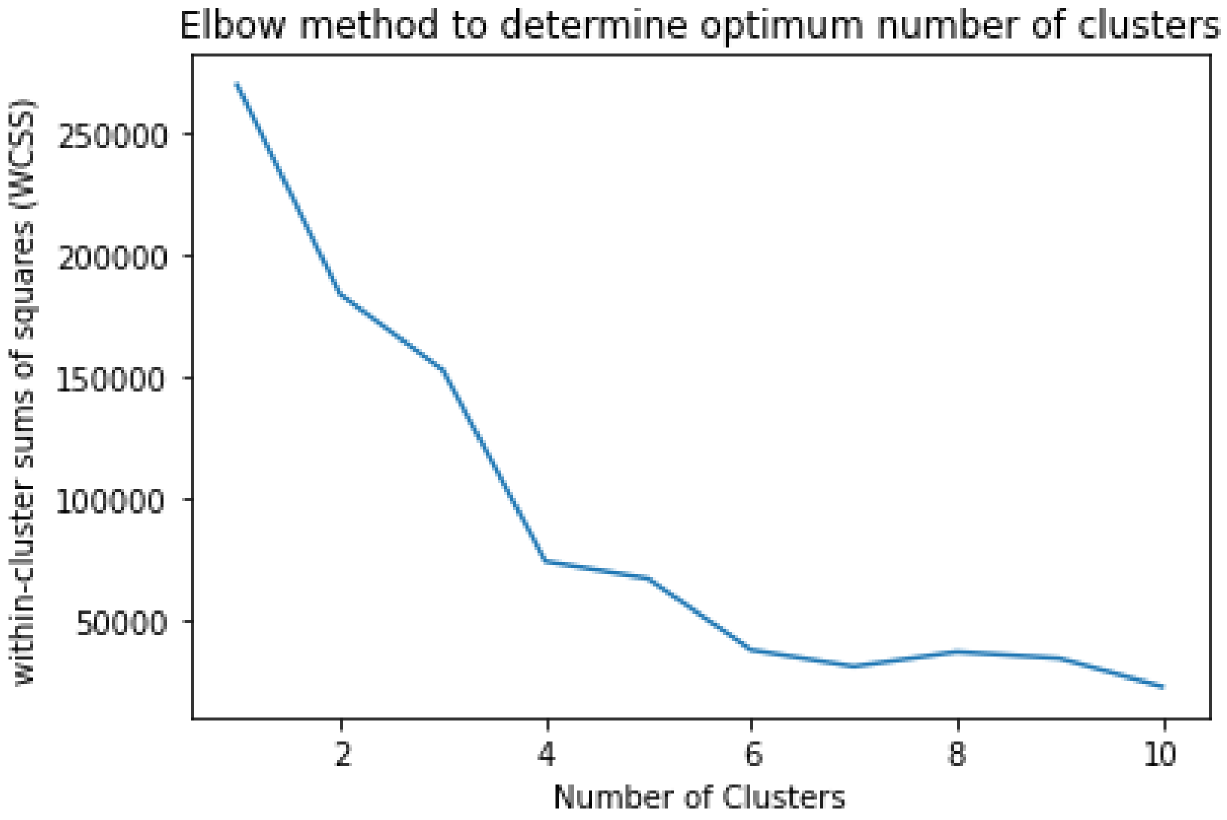


For the annual income It shows that the minmum annual income is 14k anf the maximum is 140k and it falls between 50K to 85K.

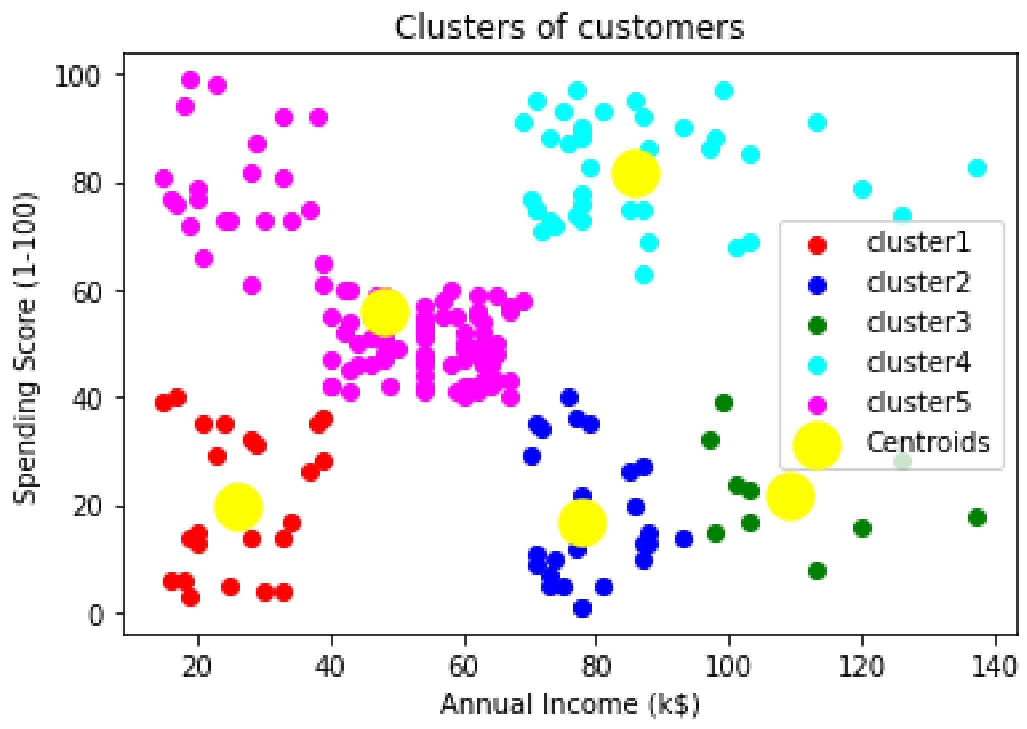


Algorithm Evaluation :

After computing the Elbow we fined that the optimal number of clusters can be generated is 5 .



We can notice a huge reduction in variation until 5 but after that, the variation doesn't go down as quickly.



The Result is 5 different clusters that created from the data. and we can see different details that tells us a lot of information about the data , for example the red cluster is representing the customers with the least income and least spending score in which the score vary from 10 - 40 and the annual income is between 10k-40k, similarly, the green cluster that shows customers with the most income and spending score.

Code:

```
#import libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random as rd
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
Mall_dataset=pd.read_csv('/content/Mall_Customers.csv')

#Distribution of Annnual Income
plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(Mall_dataset['Annual Income (k$)'])
plt.title('Distribution of Annual Income (k$)', fontsize = 20)
plt.xlabel('Range of Annual Income (k$)')
plt.ylabel('Count')

#Distribution of age
plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(data['Age'])
plt.title('Distribution of Age', fontsize = 20)
plt.xlabel('Range of Age')
plt.ylabel('Count')

#We take just the Annual Income and Spending score
data_v=Mall_dataset[["CustomerID","Genre","Age","Annual Income (k$)","Spen
ding Score (1-100)"]]
X_1=data_v[["Annual Income (k$)","Spending Score (1-100)"]]


#Scatterplot of the input data
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-
100)',  data = X_1  ,s = 60 )
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.show()


genders = Mall_dataset.Genre.value_counts()
sns.set_style("darkgrid")
plt.figure(figsize=(10,4))
sns.barplot(x=genders.index, y=genders.values)
plt.show()


class Kmeans:
    def __init__(self,X,K):
        self.X=X
        self.Output={}
        self.Centroids=np.array([]).reshape(self.X.shape[1],0)
        self.K=K
        self.m=self.X.shape[0]

    def kmeanspp(self,X,K):
        i=rd.randint(0,X.shape[0])
        Centroid_temp=np.array([X[i]])
        for k in range(1,K):
            D=np.array([])
            for x in X:
                D=np.append(D,np.min(np.sum((x-Centroid_temp)**2)))
            prob=D/np.sum(D)
            cummuprob=np.cumsum(prob)
            r=rd.random()
            i=0
            for j,p in enumerate(cummuprob):
                if r<p:
                    i=j
                    break
            Centroid_temp=np.append(Centroid_temp,[X[i]],axis=0)
        return Centroid_temp.T

    def rand(self,n_iter):
        self.Centroids=self.kmeanspp(self.X,self.K)
        for n in range(n_iter):
            Eucl_Dis=np.array([]).reshape(self.m,0)
            for k in range(self.K):
                tempDist=np.sum((self.X-self.Centroids[:,k])**2,axis=1)
                Eucl_Dis=np.c_[Eucl_Dis,tempDist]
            C=np.argmin(Eucl_Dis,axis=1)+1
            Y={}
            for k_v in range(self.K):
                Y[k_v+1]=np.array([]).reshape(2,0)
            for a in range(self.m):
                Y[C[a]]=np.c_[Y[C[a]],self.X[a]]

            for k_v in range(self.K):
                Y[k_v+1]=Y[k_v+1].T
            for k_v in range(self.K):
                self.Centroids[:,k_v]=np.mean(Y[k_v+1],axis=0)

            self.Output=Y
    def predict(self):
        return self.Output,self.Centroids.T

    def WCSS_calc(self):
        wcss=0
        for k_v in range(self.K):
            wcss+=np.sum((self.Output[k_vk+1]-self.Centroids[:,k_v])**2)
        return WCSS_calc


WS_arr=np.array([])
for K in range(1,11):
    kmeans=Kmeans(X,K)
    kmeans.rand(n_iter)
    Output,Centroids=kmeans.predict()
    wcss=0
    for k_v in range(K):
        wcss+=np.sum((Output[k_v+1]-Centroids[k_v,:])**2)
    WS_arr=np.append(WS_arr,wcss)
```