Assignment 4 - Customer Segmentation Analysis

Importing Libraries

In [ ]:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
```

2. Load the dataset into the tool

In [ ]:

```
data = pd.read_csv('Mall_Customers.csv')
# getting the shape
data.shape
```

Out[ ]:

```
(200, 5)
```

In [ ]:

```
# looking at the head of the data

data.head()
```

Out[ ]:

	CustomerID	Gender	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

3.Perform Below Visualizatons

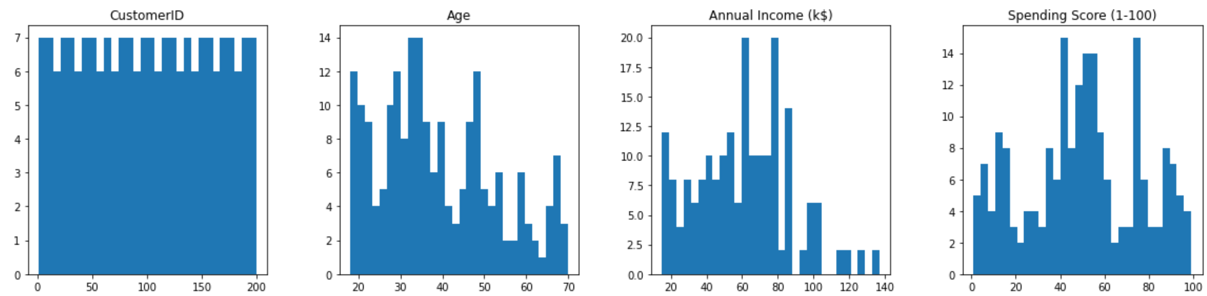
3.1 Univariate Analysis

In [ ]:

```
data.hist(figsize=(20,10), grid=False, layout=(2, 4), bins = 30)
```

Out[ ]:

```
array([[,
        ,
        ,
        ],
       [,
        ,
        ,
        ]],
      dtype=object)
```

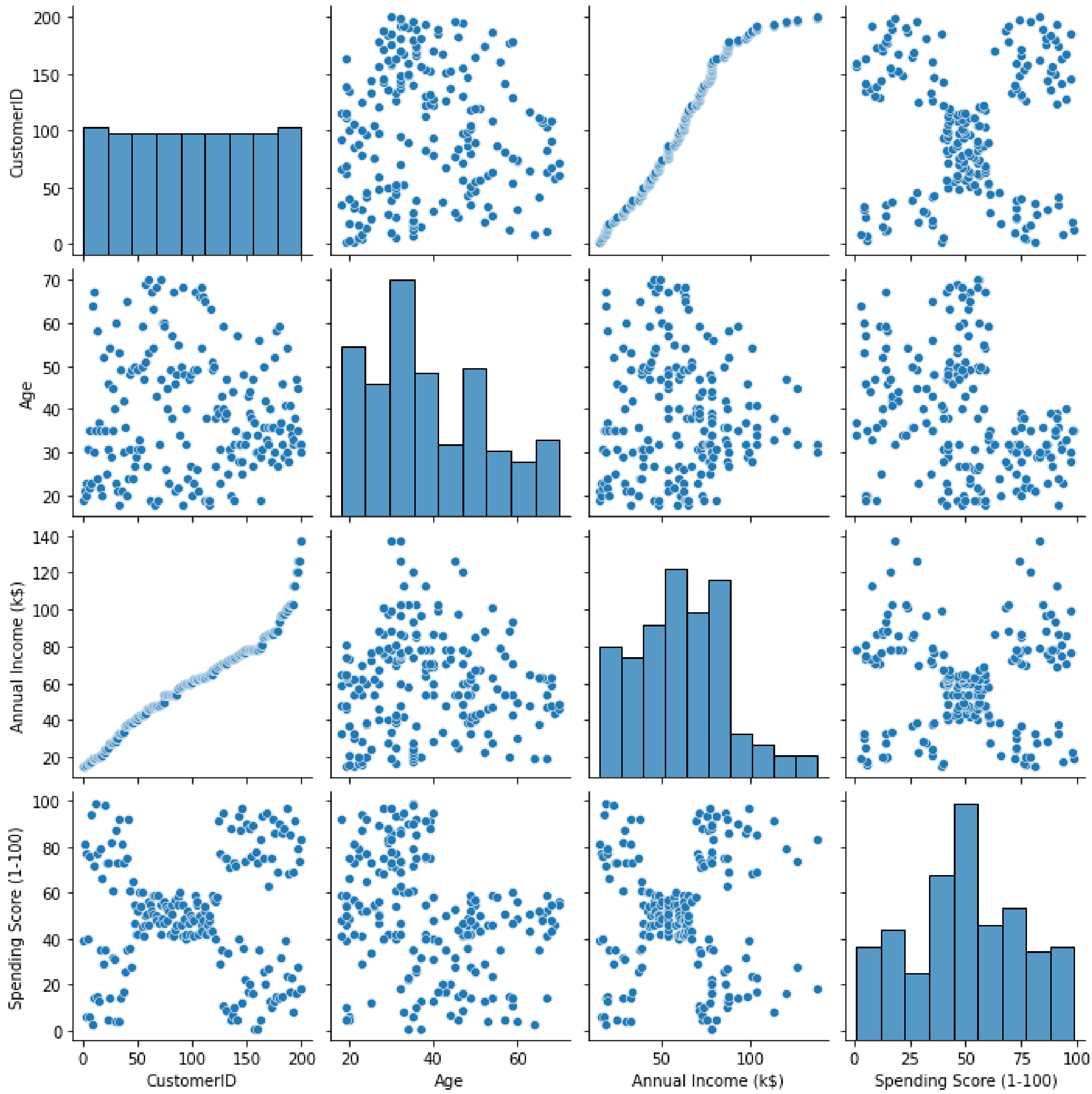


3.2 Bi-variate Analysis

In [ ]:

```
sns.pairplot(data)
```

Out[ ]:



3.3 Multi-Variate Analysis

In [ ]:

```
dataplot = sns.heatmap(data.corr(), cmap="YlGnBu", annot=True)
plt.show()
```

4.Perform descriptive statistics on the dataset.

In [ ]:

```
data.describe()
```

Out[ ]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

5.Check for Missing values and deal with them.

In [ ]:

```
data.isnull().any()
```

Out[ ]:

```
CustomerID                False
Gender                    False
Age                       False
Annual Income (k$)        False
Spending Score (1-100)    False
dtype: bool
```

9. Perform any of the clustering algorithms

10. Add the cluster data with the primary dataset

K-Means Clustering

k-means clustering based on annual income

Elbow method to find the optimal number of Clusters

In [ ]:

```
x = data.iloc[:, [3, 4]].values
x.shape
```

Out[ ]:

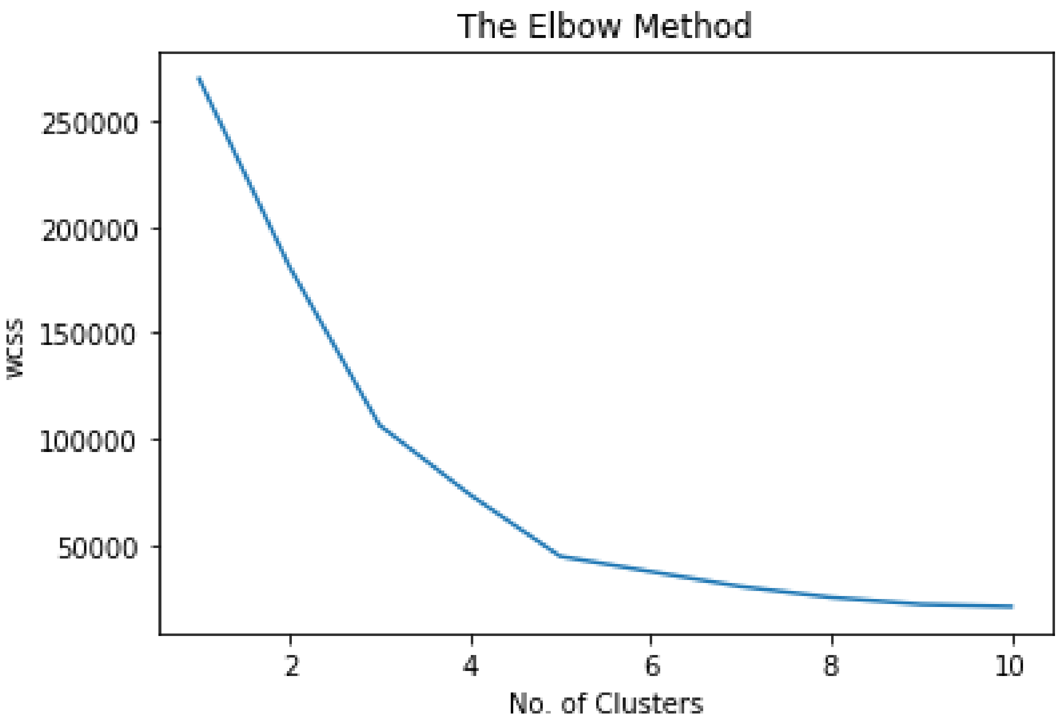
```
(200, 2)
```

In [ ]:

```
from sklearn.cluster import KMeans

wcss = []
for i in range(1, 11):
  km = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init =
10, random_state = 0)
  km.fit(x)
  wcss.append(km.inertia_)

plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('No. of Clusters')
plt.ylabel('wcss')
plt.show()
```

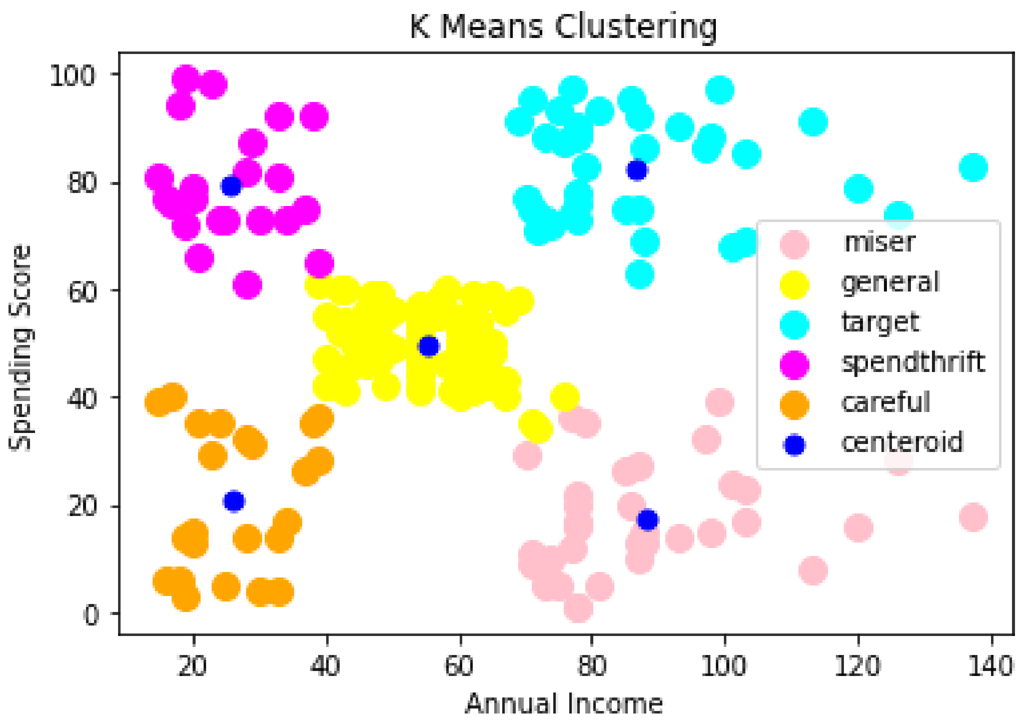


In [ ]:

```
km = KMeans(n_clusters = 5, init = 'k-means++', max_iter = 300, n_init =
10, random_state = 0)
y_means = km.fit_predict(x)

plt.scatter(x[y_means == 0, 0], x[y_means == 0, 1], s = 100, c = 'red',
label = 'miser')
plt.scatter(x[y_means == 1, 0], x[y_means == 1, 1], s = 100, c = 'orange',
label = 'general')
plt.scatter(x[y_means == 2, 0], x[y_means == 2, 1], s = 100, c = 'prink',
label = 'target')
plt.scatter(x[y_means == 3, 0], x[y_means == 3, 1], s = 100, c = 'magenta',
label = 'spendthrift')
plt.scatter(x[y_means == 4, 0], x[y_means == 4, 1], s = 100, c = 'green',
label = 'careful')
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:, 1], s = 50, c
= 'black' , label = 'centeroid')

plt.title('K Means Clustering')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.legend()
plt.show()
```



k-means clustering based on age

In [ ]:

```
x = data.iloc[:, [2, 4]].values
x.shape
```

Out[ ]:

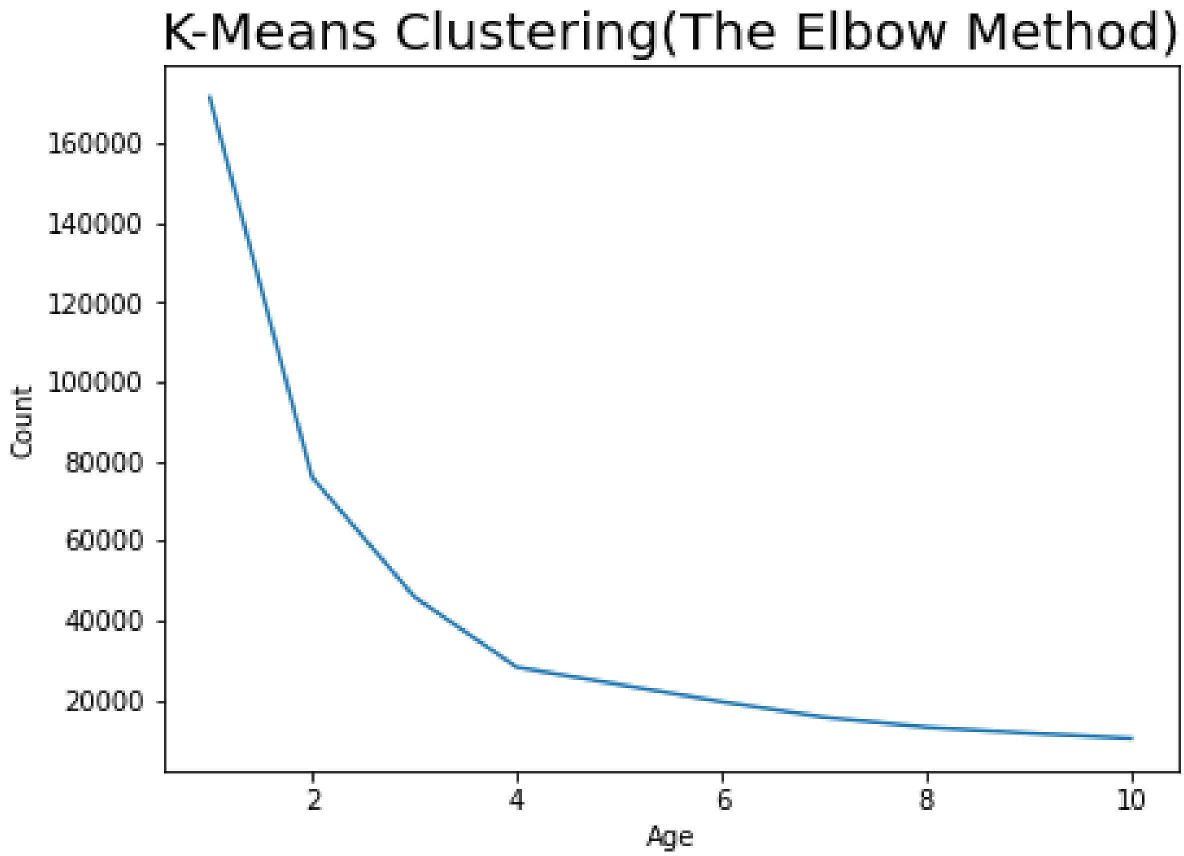
```
(200, 2)
```

In [ ]:

```
from sklearn.cluster import KMeans

wcss = []
for i in range(1, 11):
  kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300,
n_init = 10, random_state = 0)
  kmeans.fit(x)
  wcss.append(kmeans.inertia_)

plt.rcParams['figure.figsize'] = (7, 5)
plt.plot(range(1, 11), wcss)
plt.title('K-Means Clustering(The Elbow Method)', fontsize = 20)
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



In [ ]:

```
kmeans = KMeans(n_clusters = 4, init = 'k-means++', max_iter = 300, n_init
= 10, random_state = 0)
ymeans = kmeans.fit_predict(x)

plt.rcParams['figure.figsize'] = (10, 10)
plt.title('Cluster of Ages', fontsize = 30)

plt.scatter(x[ymeans == 0, 0], x[ymeans == 0, 1], s = 100, c = 'orange',
label = 'Usual Customers' )
plt.scatter(x[ymeans == 1, 0], x[ymeans == 1, 1], s = 100, c = 'yellow',
label = 'Priority Customers')
plt.scatter(x[ymeans == 2, 0], x[ymeans == 2, 1], s = 100, c = 'pink',
label = 'Target Customers(Young)')
plt.scatter(x[ymeans == 3, 0], x[ymeans == 3, 1], s = 100, c = 'red', label
= 'Target Customers(Old)')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s
= 50, c = 'blue')

plt.xlabel('Age')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

From cluster plot we can clearly see that males and females are in all the catgory that is high low and medium spending score category

In [ ]:

```
data['Gender'].replace(['Male', 'Female'], [0, 1], inplace = True)
data['Gender'].value_counts()
```

Out[ ]:

```
1    112
0     88
Name: Gender, dtype: int64
```

In [ ]:

```
x = data.iloc[:, [1, 4]].values
x.shape
```

Out[ ]:

```
(200, 2)
```

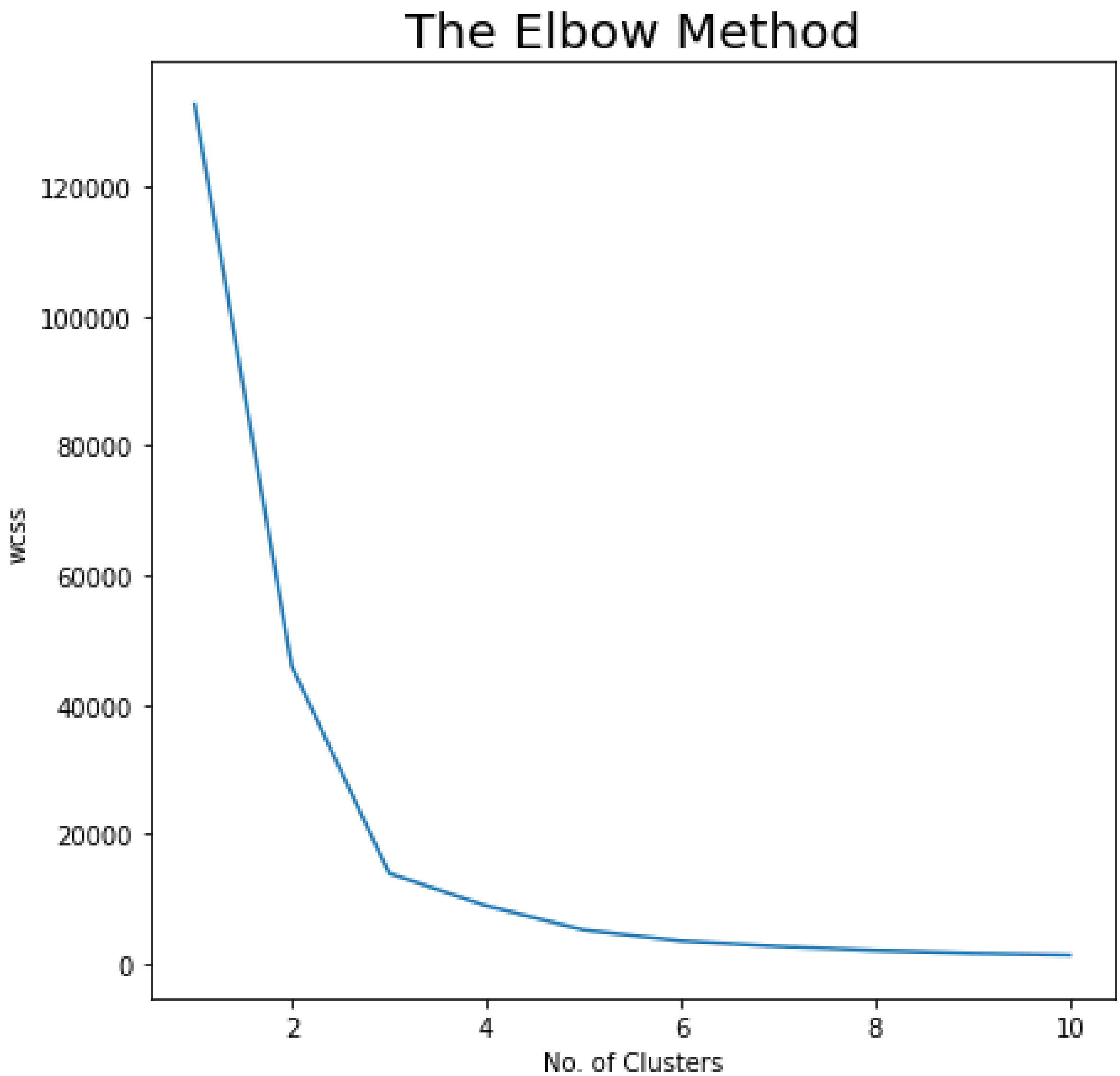
In [ ]:

```
from sklearn.cluster import KMeans

wcss = []
for i in range(1, 11):
  kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300,
n_init = 10, random_state = 0)
```

```
  kmeans.fit(x)
  wcss.append(kmeans.inertia_)

plt.rcParams['figure.figsize'] = (7, 7)
plt.title('The Elbow Method', fontsize = 20)
plt.plot(range(1, 11), wcss)
plt.xlabel('No. of Clusters', fontsize = 10)
plt.ylabel('wcss')
plt.show()
```

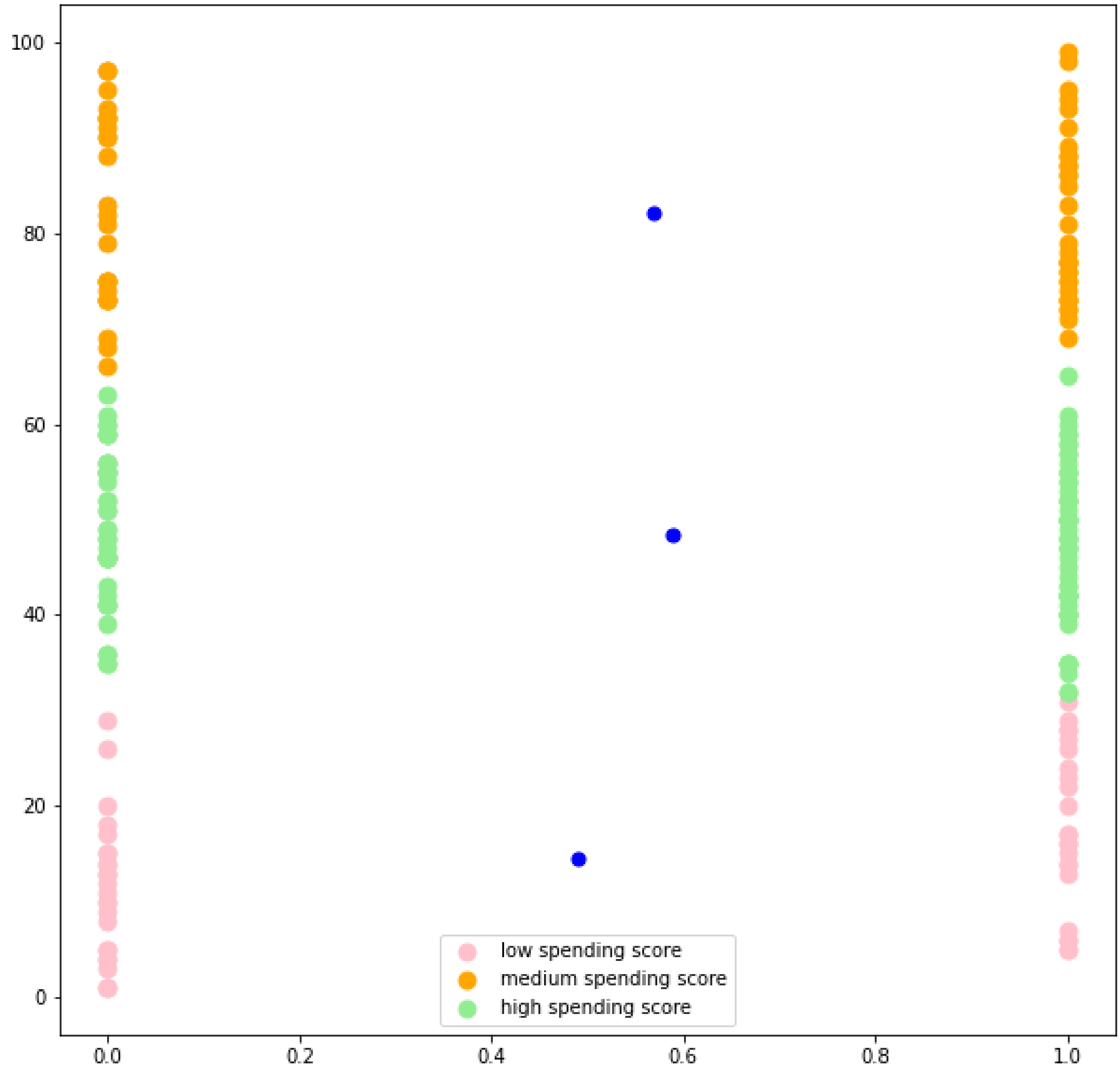


In [ ]:

```
kmeans = KMeans(n_clusters = 3, max_iter = 300, n_init = 10, random_state =
0)
ymeans = kmeans.fit_predict(x)

plt.rcParams['figure.figsize'] = (10, 10)
plt.scatter(x[ymeans == 0, 0], x[ymeans == 0, 1], s = 80, c = 'pink', label
= 'low spending score')
plt.scatter(x[ymeans == 1, 0], x[ymeans == 1, 1], s = 80, c = 'orange',
label = 'medium spending score')
plt.scatter(x[ymeans == 2, 0], x[ymeans == 2, 1], s = 80, c = 'lightgreen',
label = 'high spending score')
```

```
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:, 1], s
= 50, color = 'blue')
plt.legend()
plt.show()
```



6.Find the outliers and replace them outliers

In [ ]:

```
data['Spending Score (1-100)']=np.where(data['Spending Score (1-
100)']>10,np.median,data['Spending Score (1-100)'])
data['Spending Score (1-100)']
```

Out[ ]:

```
0
1
2                                         6
3
4
                      ...
195
196
197
198
199
Name: Spending Score (1-100), Length: 200, dtype: object
```

7.Check for Categorical columns and perform encoding.

In [ ]:

```
data.columns
```

Out[ ]:

```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
data['CustomerID'] = encoder.fit_transform(data['CustomerID'])

data.head()
```

Out[ ]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	0	Male	19	15	
1	1	Male	21	15	
2	2	Female	20	16	6
3	3	Female	23	16	
4	4	Female	31	17	

8.Scaling the data

In [ ]:

```
from sklearn.preprocessing import StandardScaler
df=StandardScaler()
data1=df.fit_transform(data)
print(data1)
```

```
[[-1.7234121  -1.12815215 -1.42456879 -1.73899919 -0.43480148]
 [-1.70609137 -1.12815215 -1.28103541 -1.73899919  1.19570407]
 [-1.68877065  0.88640526 -1.3528021  -1.70082976 -1.71591298]
 [-1.67144992  0.88640526 -1.13750203 -1.70082976  1.04041783]
 [-1.6541292   0.88640526 -0.56336851 -1.66266033 -0.39597992]
 [-1.63680847  0.88640526 -1.20926872 -1.66266033  1.00159627]
 [-1.61948775  0.88640526 -0.27630176 -1.62449091 -1.71591298]
 [-1.60216702  0.88640526 -1.13750203 -1.62449091  1.70038436]
 [-1.5848463  -1.12815215  1.80493225 -1.58632148 -1.83237767]
 [-1.56752558  0.88640526 -0.6351352  -1.58632148  0.84631002]
 [-1.55020485 -1.12815215  2.02023231 -1.58632148 -1.4053405 ]
 [-1.53288413  0.88640526 -0.27630176 -1.58632148  1.89449216]
 [-1.5155634   0.88640526  1.37433211 -1.54815205 -1.36651894]
 [-1.49824268  0.88640526 -1.06573534 -1.54815205  1.04041783]
 [-1.48092195 -1.12815215 -0.13276838 -1.54815205 -1.44416206]
 [-1.46360123 -1.12815215 -1.20926872 -1.54815205  1.11806095]
 [-1.4462805   0.88640526 -0.27630176 -1.50998262 -0.59008772]
 [-1.42895978 -1.12815215 -1.3528021  -1.50998262  0.61338066]
 [-1.41163905 -1.12815215  0.94373197 -1.43364376 -0.82301709]
 [-1.39431833  0.88640526 -0.27630176 -1.43364376  1.8556706 ]
 [-1.3769976  -1.12815215 -0.27630176 -1.39547433 -0.59008772]
 [-1.35967688 -1.12815215 -0.99396865 -1.39547433  0.88513158]
```

```
[-1.34235616  0.88640526  0.51313183 -1.3573049  -1.75473454]
[-1.32503543 -1.12815215 -0.56336851 -1.3573049   0.88513158]
[-1.30771471  0.88640526  1.08726535 -1.24279661 -1.4053405 ]
[-1.29039398 -1.12815215 -0.70690189 -1.24279661  1.23452563]
[-1.27307326  0.88640526  0.44136514 -1.24279661 -0.7065524 ]
[-1.25575253 -1.12815215 -0.27630176 -1.24279661  0.41927286]
[-1.23843181  0.88640526  0.08253169 -1.20462718 -0.74537397]
[-1.22111108  0.88640526 -1.13750203 -1.20462718  1.42863343]
[-1.20379036 -1.12815215  1.51786549 -1.16645776 -1.7935561 ]
[-1.18646963  0.88640526 -1.28103541 -1.16645776  0.88513158]
[-1.16914891 -1.12815215  1.01549866 -1.05194947 -1.7935561 ]
[-1.15182818 -1.12815215 -1.49633548 -1.05194947  1.62274124]
[-1.13450746  0.88640526  0.7284319  -1.05194947 -1.4053405 ]
[-1.11718674  0.88640526 -1.28103541 -1.05194947  1.19570407]
[-1.09986601  0.88640526  0.22606507 -1.01378004 -1.28887582]
[-1.08254529  0.88640526 -0.6351352  -1.01378004  0.88513158]
[-1.06522456  0.88640526 -0.20453507 -0.89927175 -0.93948177]
[-1.04790384  0.88640526 -1.3528021  -0.89927175  0.96277471]
[-1.03058311  0.88640526  1.87669894 -0.86110232 -0.59008772]
[-1.01326239 -1.12815215 -1.06573534 -0.86110232  1.62274124]
[-0.99594166 -1.12815215  0.65666521 -0.82293289 -0.55126616]
[-0.97862094  0.88640526 -0.56336851 -0.82293289  0.41927286]
[-0.96130021  0.88640526  0.7284319  -0.82293289 -0.86183865]
[-0.94397949  0.88640526 -1.06573534 -0.82293289  0.5745591 ]
[-0.92665877  0.88640526  0.80019859 -0.78476346  0.18634349]
[-0.90933804  0.88640526 -0.85043527 -0.78476346 -0.12422899]
[-0.89201732  0.88640526 -0.70690189 -0.78476346 -0.3183368 ]
[-0.87469659  0.88640526 -0.56336851 -0.78476346 -0.3183368 ]
[-0.85737587  0.88640526  0.7284319  -0.70842461  0.06987881]
[-0.84005514 -1.12815215 -0.41983513 -0.70842461  0.38045129]
[-0.82273442  0.88640526 -0.56336851 -0.67025518  0.14752193]
[-0.80541369 -1.12815215  1.4460988  -0.67025518  0.38045129]
[-0.78809297  0.88640526  0.80019859 -0.67025518 -0.20187212]
[-0.77077224 -1.12815215  0.58489852 -0.67025518 -0.35715836]
[-0.75345152  0.88640526  0.87196528 -0.63208575 -0.00776431]
[-0.73613079 -1.12815215  2.16376569 -0.63208575 -0.16305055]
[-0.71881007  0.88640526 -0.85043527 -0.55574689  0.03105725]
[-0.70148935 -1.12815215  1.01549866 -0.55574689 -0.16305055]
[-0.68416862 -1.12815215  2.23553238 -0.55574689  0.22516505]
[-0.6668479  -1.12815215 -1.42456879 -0.55574689  0.18634349]
[-0.64952717  0.88640526  2.02023231 -0.51757746  0.06987881]
[-0.63220645  0.88640526  1.08726535 -0.51757746  0.34162973]
[-0.61488572 -1.12815215  1.73316556 -0.47940803  0.03105725]
[-0.597565   -1.12815215 -1.49633548 -0.47940803  0.34162973]
[-0.58024427  0.88640526  0.29783176 -0.47940803 -0.00776431]
[-0.56292355  0.88640526  2.091999   -0.47940803 -0.08540743]
[-0.54560282 -1.12815215 -1.42456879 -0.47940803  0.34162973]
[-0.5282821   0.88640526 -0.49160182 -0.47940803 -0.12422899]
[-0.51096138 -1.12815215  2.23553238 -0.4412386   0.18634349]
[-0.49364065  0.88640526  0.58489852 -0.4412386  -0.3183368 ]
[-0.47631993  0.88640526  1.51786549 -0.40306917 -0.04658587]
[-0.4589992   0.88640526  1.51786549 -0.40306917  0.22516505]
[-0.44167848 -1.12815215  1.4460988  -0.25039146 -0.12422899]
[-0.42435775 -1.12815215 -0.92220196 -0.25039146  0.14752193]
[-0.40703703  0.88640526  0.44136514 -0.25039146  0.10870037]
[-0.3897163  -1.12815215  0.08253169 -0.25039146 -0.08540743]
[-0.37239558  0.88640526 -1.13750203 -0.25039146  0.06987881]
[-0.35507485  0.88640526  0.7284319  -0.25039146 -0.3183368 ]
[-0.33775413 -1.12815215  1.30256542 -0.25039146  0.03105725]
[-0.3204334  -1.12815215 -0.06100169 -0.25039146  0.18634349]
[-0.30311268 -1.12815215  2.02023231 -0.25039146 -0.35715836]
```

```
[-0.28579196  0.88640526  0.51313183 -0.25039146 -0.24069368]
[-0.26847123  0.88640526 -1.28103541 -0.25039146  0.26398661]
[-0.25115051 -1.12815215  0.65666521 -0.25039146 -0.16305055]
[-0.23382978  0.88640526  1.15903204 -0.13588317  0.30280817]
[-0.21650906  0.88640526 -1.20926872 -0.13588317  0.18634349]
[-0.19918833  0.88640526 -0.34806844 -0.09771374  0.38045129]
[-0.18186761  0.88640526  0.80019859 -0.09771374 -0.16305055]
[-0.16454688  0.88640526  2.091999   -0.05954431  0.18634349]
[-0.14722616 -1.12815215 -1.49633548 -0.05954431 -0.35715836]
[-0.12990543 -1.12815215  0.65666521 -0.02137488 -0.04658587]
[-0.11258471  0.88640526  0.08253169 -0.02137488 -0.39597992]
[-0.09526399  0.88640526 -0.49160182 -0.02137488 -0.3183368 ]
[-0.07794326 -1.12815215 -1.06573534 -0.02137488  0.06987881]
[-0.06062254  0.88640526  0.58489852 -0.02137488 -0.12422899]
[-0.04330181  0.88640526 -0.85043527 -0.02137488 -0.00776431]
[-0.02598109 -1.12815215  0.65666521  0.01679455 -0.3183368 ]
[-0.00866036 -1.12815215 -1.3528021   0.01679455 -0.04658587]
[ 0.00866036  0.88640526 -1.13750203  0.05496398 -0.35715836]
[ 0.02598109  0.88640526  0.7284319   0.05496398 -0.08540743]
[ 0.04330181 -1.12815215  2.02023231  0.05496398  0.34162973]
[ 0.06062254 -1.12815215 -0.92220196  0.05496398  0.18634349]
[ 0.07794326 -1.12815215  0.7284319   0.05496398  0.22516505]
[ 0.09526399  0.88640526 -1.28103541  0.05496398 -0.3183368 ]
[ 0.11258471  0.88640526  1.94846562  0.09313341 -0.00776431]
[ 0.12990543 -1.12815215  1.08726535  0.09313341 -0.16305055]
[ 0.14722616 -1.12815215  2.091999    0.09313341 -0.27951524]
[ 0.16454688 -1.12815215  1.94846562  0.09313341 -0.08540743]
[ 0.18186761 -1.12815215  1.87669894  0.09313341  0.06987881]
[ 0.19918833  0.88640526 -1.42456879  0.09313341  0.14752193]
[ 0.21650906  0.88640526 -0.06100169  0.13130284 -0.3183368 ]
[ 0.23382978 -1.12815215 -1.42456879  0.13130284 -0.16305055]
[ 0.25115051  0.88640526 -1.49633548  0.16947227 -0.08540743]
[ 0.26847123  0.88640526 -1.42456879  0.16947227 -0.00776431]
[ 0.28579196  0.88640526  1.73316556  0.16947227 -0.27951524]
[ 0.30311268  0.88640526  0.7284319   0.16947227  0.34162973]
[ 0.3204334   0.88640526  0.87196528  0.24581112 -0.27951524]
[ 0.33775413  0.88640526  0.80019859  0.24581112  0.26398661]
[ 0.35507485 -1.12815215 -0.85043527  0.24581112  0.22516505]
[ 0.37239558  0.88640526 -0.06100169  0.24581112 -0.39597992]
[ 0.3897163   0.88640526  0.08253169  0.32214998  0.30280817]
[ 0.40703703 -1.12815215  0.010765    0.32214998  1.58391968]
[ 0.42435775  0.88640526 -1.13750203  0.36031941 -0.82301709]
[ 0.44167848  0.88640526 -0.56336851  0.36031941  1.04041783]
[ 0.4589992  -1.12815215  0.29783176  0.39848884 -0.59008772]
[ 0.47631993 -1.12815215  0.08253169  0.39848884  1.73920592]
[ 0.49364065 -1.12815215  1.4460988   0.39848884 -1.52180518]
[ 0.51096138 -1.12815215 -0.06100169  0.39848884  0.96277471]
[ 0.5282821  -1.12815215  0.58489852  0.39848884 -1.5994483 ]
[ 0.54560282 -1.12815215  0.010765    0.39848884  0.96277471]
[ 0.56292355  0.88640526 -0.99396865  0.43665827 -0.62890928]
[ 0.58024427  0.88640526 -0.56336851  0.43665827  0.80748846]
[ 0.597565   -1.12815215 -1.3528021   0.4748277  -1.75473454]
[ 0.61488572  0.88640526 -0.70690189  0.4748277   1.46745499]
[ 0.63220645  0.88640526  0.36959845  0.4748277  -1.67709142]
[ 0.64952717 -1.12815215 -0.49160182  0.4748277   0.88513158]
[ 0.6668479  -1.12815215 -1.42456879  0.51299713 -1.56062674]
[ 0.68416862  0.88640526 -0.27630176  0.51299713  0.84631002]
[ 0.70148935  0.88640526  1.30256542  0.55116656 -1.75473454]
[ 0.71881007 -1.12815215 -0.49160182  0.55116656  1.6615628 ]
[ 0.73613079  0.88640526 -0.77866858  0.58933599 -0.39597992]
[ 0.75345152  0.88640526 -0.49160182  0.58933599  1.42863343]
```

```
 [ 0.77077224 -1.12815215 -0.99396865  0.62750542 -1.48298362]
 [ 0.78809297 -1.12815215 -0.77866858  0.62750542  1.81684904]
 [ 0.80541369 -1.12815215  0.65666521  0.62750542 -0.55126616]
 [ 0.82273442  0.88640526 -0.49160182  0.62750542  0.92395314]
 [ 0.84005514  0.88640526 -0.34806844  0.66567484 -1.09476801]
 [ 0.85737587 -1.12815215 -0.34806844  0.66567484  1.54509812]
 [ 0.87469659 -1.12815215  0.29783176  0.66567484 -1.28887582]
 [ 0.89201732 -1.12815215  0.010765    0.66567484  1.46745499]
 [ 0.90933804  0.88640526  0.36959845  0.66567484 -1.17241113]
 [ 0.92665877  0.88640526 -0.06100169  0.66567484  1.00159627]
 [ 0.94397949  0.88640526  0.58489852  0.66567484 -1.32769738]
 [ 0.96130021  0.88640526 -0.85043527  0.66567484  1.50627656]
 [ 0.97862094 -1.12815215 -0.13276838  0.66567484 -1.91002079]
 [ 0.99594166  0.88640526 -0.6351352   0.66567484  1.07923939]
 [ 1.01326239 -1.12815215 -0.34806844  0.66567484 -1.91002079]
 [ 1.03058311  0.88640526 -0.6351352   0.66567484  0.88513158]
 [ 1.04790384  0.88640526  1.23079873  0.70384427 -0.59008772]
 [ 1.06522456  0.88640526 -0.70690189  0.70384427  1.27334719]
 [ 1.08254529 -1.12815215 -1.42456879  0.78018313 -1.75473454]
 [ 1.09986601  0.88640526 -0.56336851  0.78018313  1.6615628 ]
 [ 1.11718674 -1.12815215  0.80019859  0.93286085 -0.93948177]
 [ 1.13450746  0.88640526 -0.20453507  0.93286085  0.96277471]
 [ 1.15182818 -1.12815215  0.22606507  0.97103028 -1.17241113]
 [ 1.16914891  0.88640526 -0.41983513  0.97103028  1.73920592]
 [ 1.18646963  0.88640526 -0.20453507  1.00919971 -0.90066021]
 [ 1.20379036 -1.12815215 -0.49160182  1.00919971  0.49691598]
 [ 1.22111108 -1.12815215  0.08253169  1.00919971 -1.44416206]
 [ 1.23843181 -1.12815215 -0.77866858  1.00919971  0.96277471]
 [ 1.25575253 -1.12815215 -0.20453507  1.00919971 -1.56062674]
 [ 1.27307326 -1.12815215 -0.20453507  1.00919971  1.62274124]
 [ 1.29039398  0.88640526  0.94373197  1.04736914 -1.44416206]
 [ 1.30771471  0.88640526 -0.6351352   1.04736914  1.38981187]
 [ 1.32503543 -1.12815215  1.37433211  1.04736914 -1.36651894]
 [ 1.34235616 -1.12815215 -0.85043527  1.04736914  0.72984534]
 [ 1.35967688 -1.12815215  1.4460988   1.23821628 -1.4053405 ]
 [ 1.3769976  -1.12815215 -0.27630176  1.23821628  1.54509812]
 [ 1.39431833  0.88640526 -0.13276838  1.390894   -0.7065524 ]
 [ 1.41163905  0.88640526 -0.49160182  1.390894    1.38981187]
 [ 1.42895978 -1.12815215  0.51313183  1.42906343 -1.36651894]
 [ 1.4462805   0.88640526 -0.70690189  1.42906343  1.46745499]
 [ 1.46360123  0.88640526  0.15429838  1.46723286 -0.43480148]
 [ 1.48092195 -1.12815215 -0.6351352   1.46723286  1.81684904]
 [ 1.49824268  0.88640526  1.08726535  1.54357172 -1.01712489]
 [ 1.5155634  -1.12815215 -0.77866858  1.54357172  0.69102378]
 [ 1.53288413  0.88640526  0.15429838  1.61991057 -1.28887582]
 [ 1.55020485  0.88640526 -0.20453507  1.61991057  1.35099031]
 [ 1.56752558  0.88640526 -0.34806844  1.61991057 -1.05594645]
 [ 1.5848463   0.88640526 -0.49160182  1.61991057  0.72984534]
 [ 1.60216702 -1.12815215 -0.41983513  2.00160487 -1.63826986]
 [ 1.61948775  0.88640526 -0.06100169  2.00160487  1.58391968]
 [ 1.63680847  0.88640526  0.58489852  2.26879087 -1.32769738]
 [ 1.6541292   0.88640526 -0.27630176  2.26879087  1.11806095]
 [ 1.67144992  0.88640526  0.44136514  2.49780745 -0.86183865]
 [ 1.68877065 -1.12815215 -0.49160182  2.49780745  0.92395314]
 [ 1.70609137 -1.12815215 -0.49160182  2.91767117 -1.25005425]
 [ 1.7234121  -1.12815215 -0.6351352   2.91767117  1.27334719]]
```

11. Split the data into dependent and independent variables.

11.1 Split the data in to Independent variables.

In [ ]:

```
X = data.iloc[:, [3, 4]].values
```

```
X.shape
```

```
print(X)
```

```
[[ 15  39]
 [ 15  81]
 [ 16   6]
 [ 16  77]
 [ 17  40]
 [ 17  76]
 [ 18   6]
 [ 18  94]
 [ 19   3]
 [ 19  72]
 [ 19  14]
 [ 19  99]
 [ 20  15]
 [ 20  77]
 [ 20  13]
 [ 20  79]
 [ 21  35]
 [ 21  66]
 [ 23  29]
 [ 23  98]
 [ 24  35]
 [ 24  73]
 [ 25   5]
 [ 25  73]
 [ 28  14]
 [ 28  82]
 [ 28  32]
 [ 28  61]
 [ 29  31]
 [ 29  87]
 [ 30   4]
 [ 30  73]
 [ 33   4]
 [ 33  92]
 [ 33  14]
 [ 33  81]
 [ 34  17]
 [ 34  73]
 [ 37  26]
 [ 37  75]
 [ 38  35]
 [ 38  92]
 [ 39  36]
 [ 39  61]
 [ 39  28]
 [ 39  65]
 [ 40  55]
 [ 40  47]
 [ 40  42]
 [ 40  42]
 [ 42  52]
 [ 42  60]
 [ 43  54]
 [ 43  60]
 [ 43  45]
 [ 43  41]
 [ 44  50]
 [ 44  46]
```

```
[ 46  51]
[ 46  46]
[ 46  56]
[ 46  55]
[ 47  52]
[ 47  59]
[ 48  51]
[ 48  59]
[ 48  50]
[ 48  48]
[ 48  59]
[ 48  47]
[ 49  55]
[ 49  42]
[ 50  49]
[ 50  56]
[ 54  47]
[ 54  54]
[ 54  53]
[ 54  48]
[ 54  52]
[ 54  42]
[ 54  51]
[ 54  55]
[ 54  41]
[ 54  44]
[ 54  57]
[ 54  46]
[ 57  58]
[ 57  55]
[ 58  60]
[ 58  46]
[ 59  55]
[ 59  41]
[ 60  49]
[ 60  40]
[ 60  42]
[ 60  52]
[ 60  47]
[ 60  50]
[ 61  42]
[ 61  49]
[ 62  41]
[ 62  48]
[ 62  59]
[ 62  55]
[ 62  56]
[ 62  42]
[ 63  50]
[ 63  46]
[ 63  43]
[ 63  48]
[ 63  52]
[ 63  54]
[ 64  42]
[ 64  46]
[ 65  48]
[ 65  50]
[ 65  43]
[ 65  59]
[ 67  43]
```

```
[ 67  57]
[ 67  56]
[ 67  40]
[ 69  58]
[ 69  91]
[ 70  29]
[ 70  77]
[ 71  35]
[ 71  95]
[ 71  11]
[ 71  75]
[ 71   9]
[ 71  75]
[ 72  34]
[ 72  71]
[ 73   5]
[ 73  88]
[ 73   7]
[ 73  73]
[ 74  10]
[ 74  72]
[ 75   5]
[ 75  93]
[ 76  40]
[ 76  87]
[ 77  12]
[ 77  97]
[ 77  36]
[ 77  74]
[ 78  22]
[ 78  90]
[ 78  17]
[ 78  88]
[ 78  20]
[ 78  76]
[ 78  16]
[ 78  89]
[ 78   1]
[ 78  78]
[ 78   1]
[ 78  73]
[ 79  35]
[ 79  83]
[ 81   5]
[ 81  93]
[ 85  26]
[ 85  75]
[ 86  20]
[ 86  95]
[ 87  27]
[ 87  63]
[ 87  13]
[ 87  75]
[ 87  10]
[ 87  92]
[ 88  13]
[ 88  86]
[ 88  15]
[ 88  69]
[ 93  14]
[ 93  90]
```

```
 [ 97  32]
 [ 97  86]
 [ 98  15]
 [ 98  88]
 [ 99  39]
 [ 99  97]
 [101  24]
 [101  68]
 [103  17]
 [103  85]
 [103  23]
 [103  69]
 [113   8]
 [113  91]
 [120  16]
 [120  79]
 [126  28]
 [126  74]
 [137  18]
 [137  83]]
```

11.2 Split the data in to Dependent variables.

In [ ]:

```
y=data.iloc[:,-2].values
print(y)
```

```
[ 15  15  16  16  17  17  18  18  19  19  19  19  20  20  20  20  21  21
  23  23  24  24  25  25  28  28  28  28  29  29  30  30  33  33  33  33
  34  34  37  37  38  38  39  39  39  39  40  40  40  40  42  42  43  43
  43  43  44  44  46  46  46  46  47  47  48  48  48  48  48  48  49  49
  50  50  54  54  54  54  54  54  54  54  54  54  54  54  57  57  58  58
  59  59  60  60  60  60  60  60  61  61  62  62  62  62  62  62  63  63
  63  63  63  63  64  64  65  65  65  65  67  67  67  67  69  69  70  70
  71  71  71  71  71  71  72  72  73  73  73  73  74  74  75  75  76  76
  77  77  77  77  78  78  78  78  78  78  78  78  78  78  78  78  79  79
  81  81  85  85  86  86  87  87  87  87  87  87  88  88  88  88  93  93
  97  97  98  98  99  99 101 101 103 103 103 103 113 113 120 120 126 126
 137 137]
```

12.Split the data into training and testing

In [ ]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

# getting the shapes
print("Shape of x_train :", X_train.shape)
print("Shape of x_test :", X_test.shape)
print("Shape of y_train :", y_train.shape)
print("Shape of y_test :", y_test.shape)
```

```
Shape of x_train : (160, 2)
Shape of x_test : (40, 2)
Shape of y_train : (160,)
Shape of y_test : (40,)
```

13.Build the model

In [ ]:

```
test_size=0.33
seed=7
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=test_size,rand
om_state=seed)
```

14.Train the Model

In [ ]:

```
print(X_train)
```

```
[[ 98  15]
 [ 42  60]
 [ 99  39]
 [ 75   5]
 [ 54  41]
 [ 65  50]
 [ 60  52]
 [ 34  73]
 [ 72  34]
 [ 62  59]
 [ 61  42]
 [ 40  42]
 [ 17  76]
 [ 21  66]
 [ 78   1]
 [ 87  27]
 [137  83]
 [120  16]
 [ 47  52]
 [ 48  51]
 [ 28  32]
 [ 78  22]
 [ 38  92]
 [ 43  54]
 [ 93  90]
 [ 54  55]
 [ 18  94]
 [ 59  41]
 [ 87  92]
 [ 78  17]
 [ 49  55]
 [ 86  20]
 [ 63  54]
 [ 19   3]
 [ 62  56]
 [ 54  42]
 [ 70  77]
 [ 85  26]
 [ 29  87]
 [ 16  77]
 [ 37  75]
 [ 42  52]
 [ 54  48]
 [ 81  93]
 [ 65  48]
 [ 43  45]
 [ 59  55]
 [ 81   5]
 [ 39  61]
 [ 78  73]
 [ 40  42]
 [ 87  13]
 [ 46  46]
 [ 20  79]
 [ 43  60]
 [ 24  35]
 [ 20  13]
 [101  68]
 [ 72  71]
```

```
[ 54  53]
[ 19  72]
[ 62  42]
[ 63  46]
[ 78  78]
[ 20  15]
[ 21  35]
[ 40  47]
[ 77  74]
[ 67  57]
[ 24  73]
[ 79  83]
[ 71   9]
[ 97  86]
[ 50  56]
[ 30   4]
[ 77  36]
[ 33  92]
[ 77  97]
[ 85  75]
[ 88  13]
[ 69  91]
[137  18]
[ 62  41]
[ 78   1]
[ 97  32]
[ 46  55]
[ 33  81]
[ 19  14]
[103  23]
[ 49  42]
[113  91]
[ 60  40]
[ 67  43]
[ 77  12]
[ 15  81]
[ 54  44]
[103  85]
[ 57  55]
[ 73  73]
[ 17  40]
[ 37  26]
[ 87  75]
[ 33  14]
[ 64  42]
[ 78  20]
[ 44  50]
[ 48  47]
[ 39  28]
[ 23  98]
[ 18   6]
[ 43  41]
[ 54  54]
[ 15  39]
[ 87  10]
[ 73  88]
[ 71  95]
[ 88  15]
[ 48  59]
[ 86  95]
[ 73   7]
```

```
 [ 39  36]
 [ 63  52]
 [ 58  46]
 [ 50  49]
 [ 25  73]
 [ 76  40]
 [ 99  97]
 [ 60  49]
 [ 62  55]
 [ 78  88]
 [ 48  48]
 [ 28  82]
 [126  28]
 [ 88  86]]
```

In [ ]:

```
print(y_train)
```

```
[ 98  42  99  75  54  65  60  34  72  62  61  40  17  21  78  87 137 120
  47  48  28  78  38  43  93  54  18  59  87  78  49  86  63  19  62  54
  70  85  29  16  37  42  54  81  65  43  59  81  39  78  40  87  46  20
  43  24  20 101  72  54  19  62  63  78  20  21  40  77  67  24  79  71
  97  50  30  77  33  77  85  88  69 137  62  78  97  46  33  19 103  49
 113  60  67  77  15  54 103  57  73  17  37  87  33  64  78  44  48  39
  23  18  43  54  15  87  73  71  88  48  86  73  39  63  58  50  25  76
  99  60  62  78  48  28 126  88]
```

15.Test the Model

In [ ]:

```
print(X_test)
```

```
[[ 57  58]
 [ 67  56]
 [ 25   5]
 [ 19  99]
 [120  79]
 [ 16   6]
 [ 67  40]
 [ 60  42]
 [ 48  50]
 [ 47  59]
 [ 63  43]
 [ 60  47]
 [ 74  10]
 [ 48  59]
 [103  17]
 [ 78  89]
 [ 28  14]
 [ 61  49]
 [ 78  76]
 [ 40  55]
 [ 93  14]
 [ 74  72]
 [ 76  87]
 [ 54  47]
 [101  24]
 [ 87  63]
 [ 62  48]
 [126  74]
 [ 63  48]
 [ 88  69]
 [ 44  46]
 [ 63  50]
 [ 79  35]
```

```
 [ 54  57]
 [ 70  29]
 [ 54  46]
 [ 71  35]
 [ 98  88]
 [ 54  51]
 [ 65  43]
 [ 71  75]
 [ 71  11]
 [ 46  56]
 [ 69  58]
 [ 28  61]
 [ 38  35]
 [ 39  65]
 [103  69]
 [ 30  73]
 [ 78  16]
 [ 33   4]
 [ 20  77]
 [ 65  59]
 [ 54  52]
 [ 71  75]
 [ 29  31]
 [ 23  29]
 [ 75  93]
 [ 73   5]
 [ 60  50]
 [ 58  60]
 [ 46  51]
 [ 64  46]
 [ 78  90]
 [ 34  17]
 [113   8]]
```

In [ ]:

```
print(y_test)
```

```
[ 57  67  25  19 120  16  67  60  48  47  63  60  74  48 103  78  28  61
  78  40  93  74  76  54 101  87  62 126  63  88  44  63  79  54  70  54
  71  98  54  65  71  71  46  69  28  38  39 103  30  78  33  20  65  54
  71  29  23  75  73  60  58  46  64  78  34 113]
```

16.Measure the performance using metrics

In [ ]:

```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
X_train=[5,-1,2,10]
Y_test=[3.5,-0.9,2,9.9]
print('RSquared=',r2_score(X_train,Y_test))
print('MAE=',mean_absolute_error(X_train,Y_test))
print('MSE=',mean_squared_error(X_train,Y_test))
```

```
RSquared= 0.9656060606060606
MAE= 0.42499999999999993
MSE= 0.5674999999999999
```