In [8]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# to remove warnings do the following
import warnings
warnings.filterwarnings('ignore')
```

In [9]:
```
pwd
```

Out[9]: 'C:\\Users\\yokes\\Downloads'

In [10]:
```
import os
os.chdir("C:\\Users\\yokes\\Downloads")
```

# 1. Download the dataset: Dataset

# 2. Load the dataset into the tool.

In [11]:
```
df = pd.read_csv('Mall_Customers.csv')
```

In [12]:
```
df.head()
```

Out[12]:

| | 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 |

In [13]:
```
df = df.rename(columns = {'Annual Income (k$)': 'Annual_Income','Spending Score (1
df.head()
```

Out[13]:

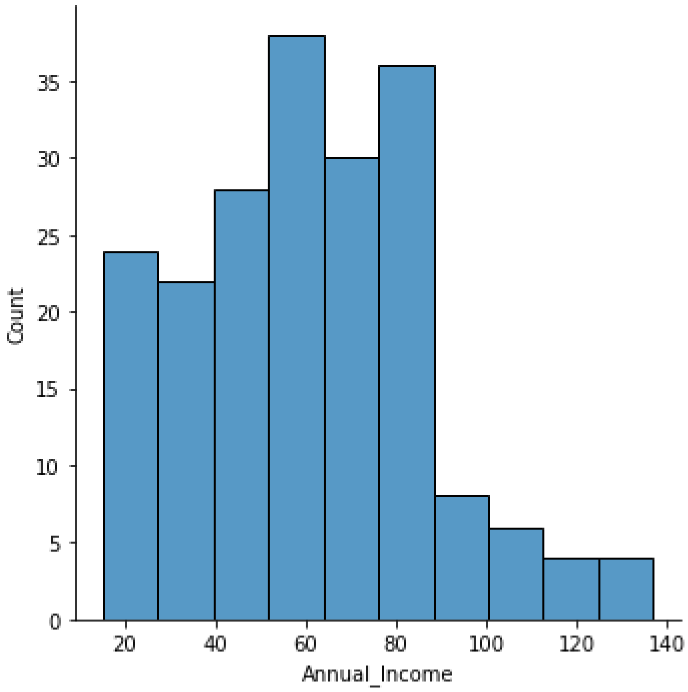
| | CustomerID | Gender | Age | Annual_Income | Spending_Score |
|---|---|---|---|---|---|
| **0** | 1 | Male | 19 | 15 | 39 |
| **1** | 2 | Male | 21 | 15 | 81 |
| **2** | 3 | Female | 20 | 16 | 6 |
| **3** | 4 | Female | 23 | 16 | 77 |
| **4** | 5 | Female | 31 | 17 | 40 |

# 3. Perform Below Visualizations.

# Univariate Analysis

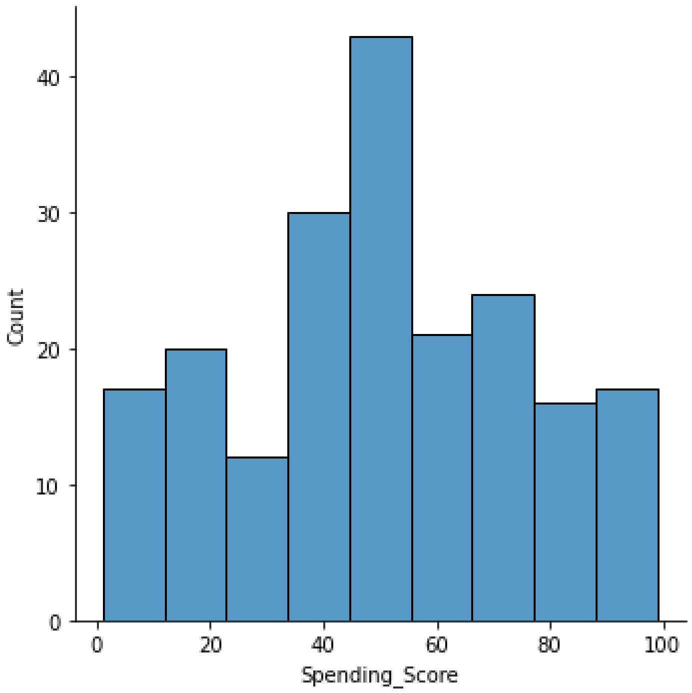
In [14]:
```
sns.displot(df.Annual_Income)
```

Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x2b1cf3cbc70>
```



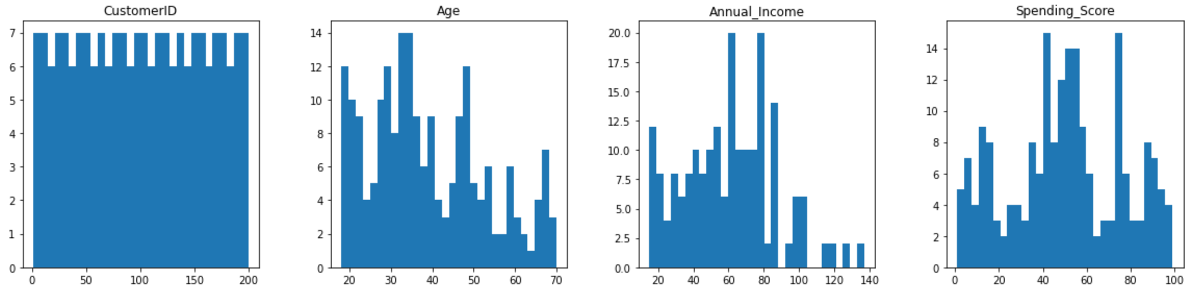

In [15]:
```
sns.displot(df.Spending_Score)
```

Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x2b1cfb87c40>
```




In [16]:
```
df.hist(figsize=(20,10),grid=False,layout=(2,4),bins=30)
```

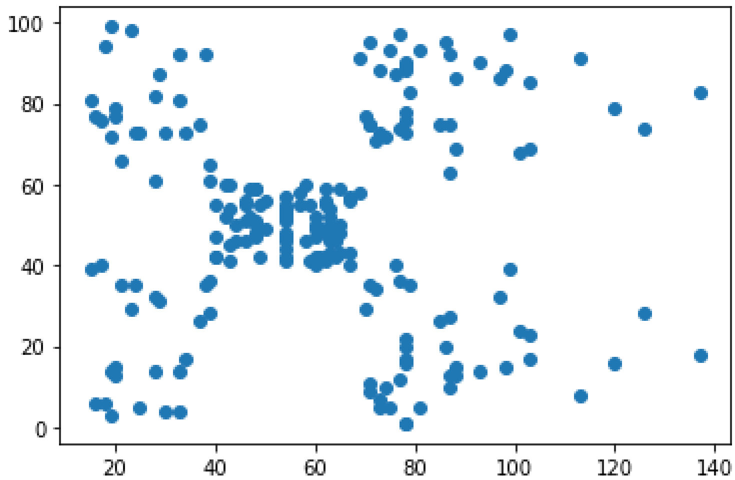
Out[16]:
```
array([[<AxesSubplot:title={'center':'CustomerID'}>,
        <AxesSubplot:title={'center':'Age'}>,
        <AxesSubplot:title={'center':'Annual_Income'}>,
        <AxesSubplot:title={'center':'Spending_Score'}>],
       [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>]],
      dtype=object)
```




# Bi- Variate Analysis

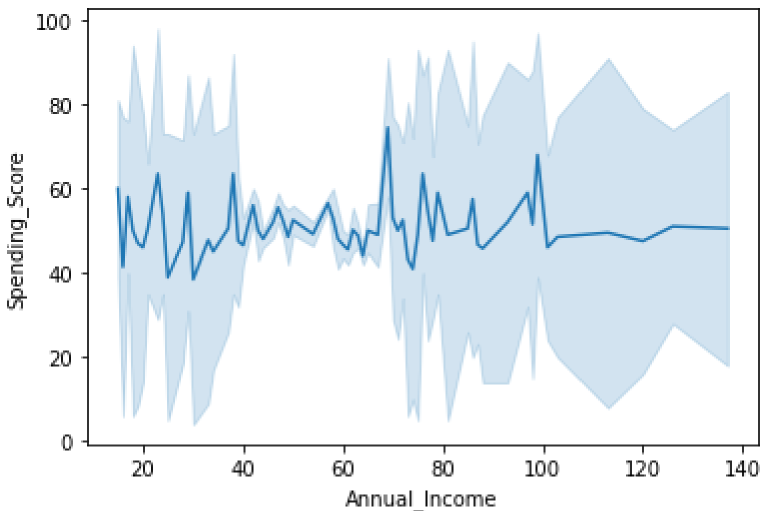
In [17]:
```
plt.scatter(df.Annual_Income,df.Spending_Score)
```

Out[17]: <matplotlib.collections.PathCollection at 0x2b1d04d5550>



In [18]:
```
sns.lineplot(df.Annual_Income,df.Spending_Score)
```

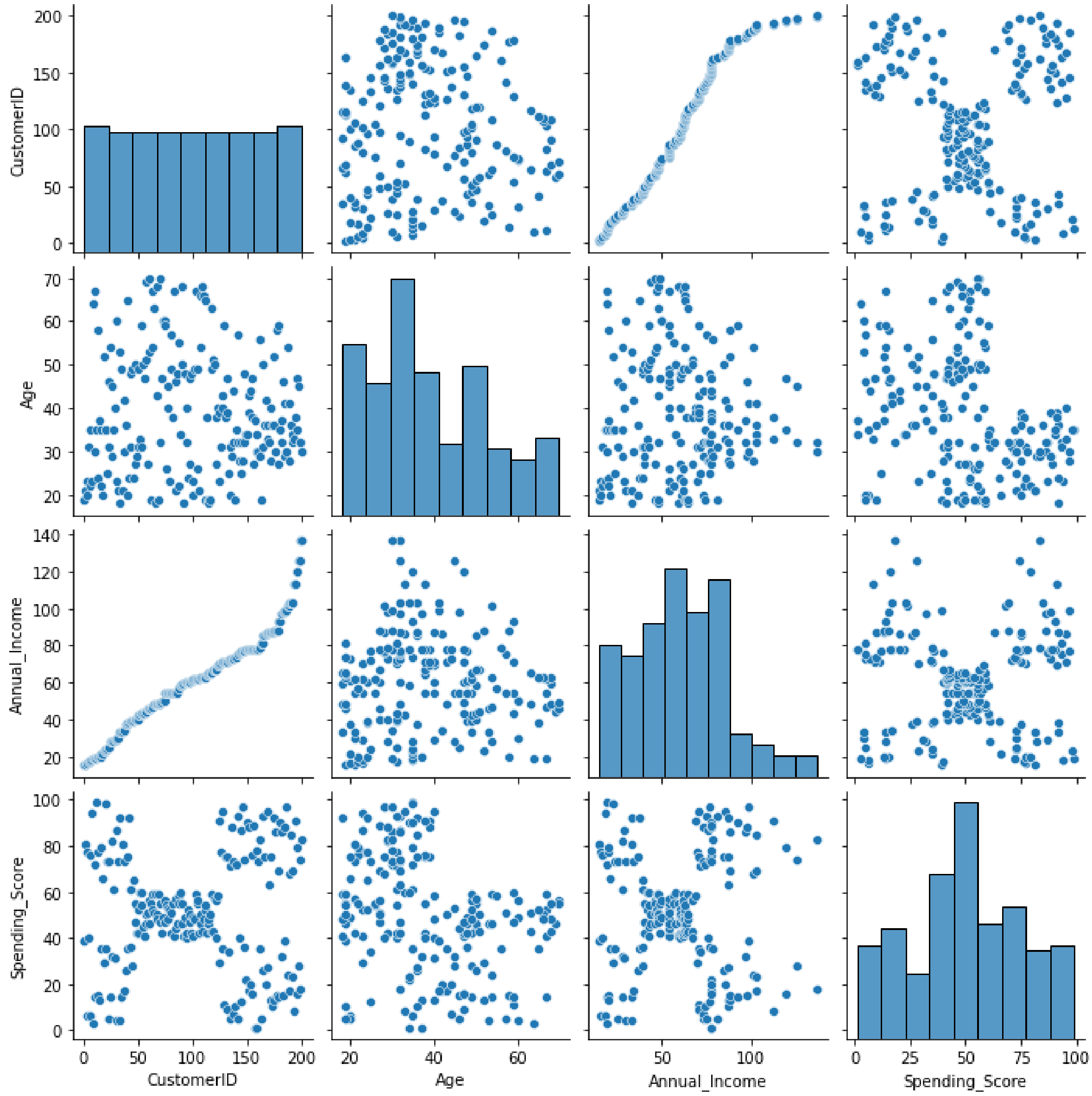
Out[18]: <AxesSubplot:xlabel='Annual_Income', ylabel='Spending_Score'>




# Multi-Variate Analysis

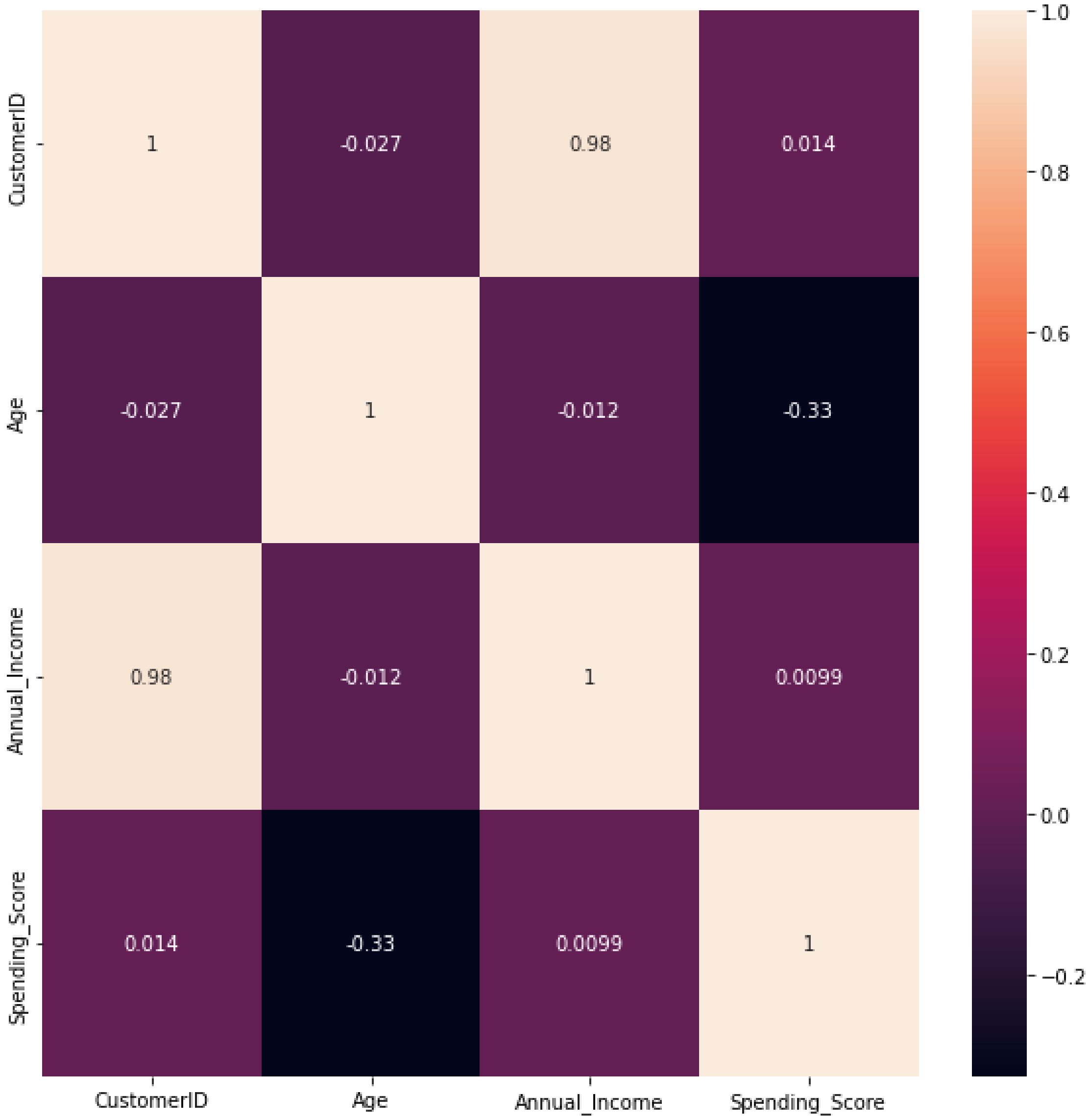
In [19]:
```
sns.pairplot(df)
```

Out[19]: <seaborn.axisgrid.PairGrid at 0x2b1cfb92be0>




In [20]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
```

Out[20]: <AxesSubplot:>




# 4. Perform descriptive statistics on the dataset.

In [21]:
```
df.describe()
```

Out[21]:

| | CustomerID | Age | Annual_Income | Spending_Score |
|---|---|---|---|---|
| **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 |

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   CustomerID      200 non-null    int64 
 1   Gender          200 non-null    object
 2   Age             200 non-null    int64 
 3   Annual_Income   200 non-null    int64 
 4   Spending_Score  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

# 5. Check for Missing values and deal with them.

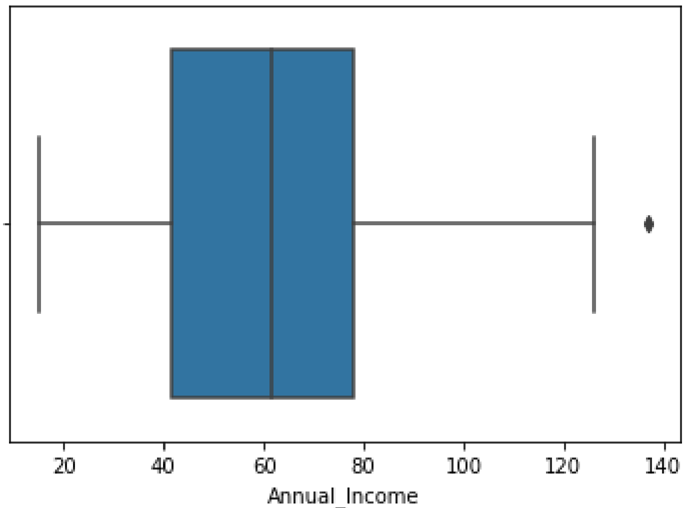
In [23]:
```
df.isnull().sum()
```

Out[23]:
```
CustomerID        0
Gender            0
Age               0
Annual_Income     0
Spending_Score    0
dtype: int64
```

# 6. Find the outliers and replace them outliers

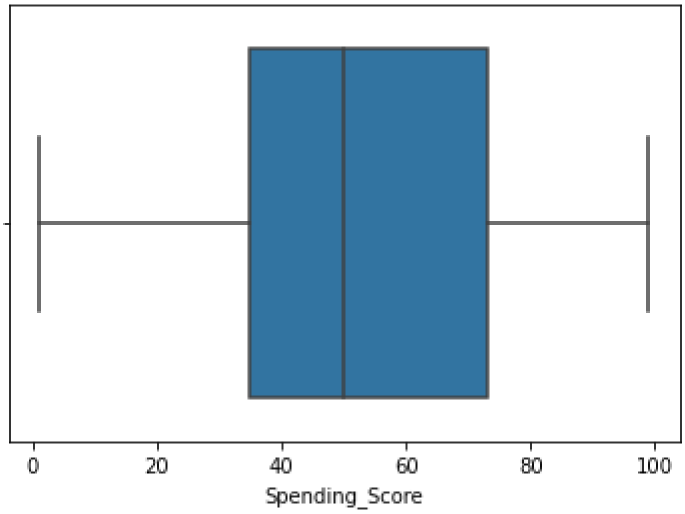
In [24]:
```
sns.boxplot(df.Annual_Income)
```

Out[24]:
```
<AxesSubplot:xlabel='Annual_Income'>
```




In [25]:
```
sns.boxplot(df.Spending_Score)
```

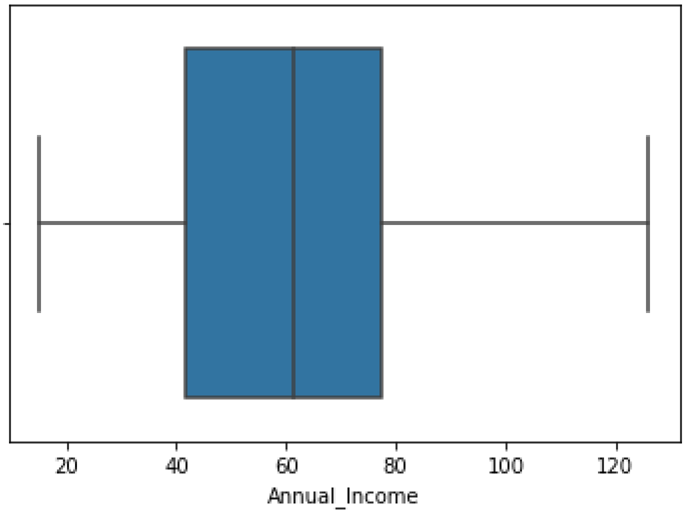
Out[25]:
```
<AxesSubplot:xlabel='Spending_Score'>
```




In [26]:
```
#replacing the outliers
median=df['Annual_Income'].median()
print(median)
df['Annual_Income']=df['Annual_Income'].mask(df['Annual_Income']>130,61.5)
sns.boxplot(df['Annual_Income'])
```

```
61.5
```

Out[26]:
```
<AxesSubplot:xlabel='Annual_Income'>
```




# 7. Check for Categorical columns and perform encoding.

In [27]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df.Gender=le.fit_transform(df.Gender)
df.head()
```

Out[27]:

| | CustomerID | Gender | Age | Annual_Income | Spending_Score |
|---|---|---|---|---|---|
| **0** | 1 | 1 | 19 | 15.0 | 39 |
| **1** | 2 | 1 | 21 | 15.0 | 81 |
| **2** | 3 | 0 | 20 | 16.0 | 6 |
| **3** | 4 | 0 | 23 | 16.0 | 77 |
| **4** | 5 | 0 | 31 | 17.0 | 40 |

# 8. Scaling the data

In [28]:

```
data=df.drop(['CustomerID'],axis=1)
data.head()
```

Out[28]:

| | Gender | Age | Annual_Income | Spending_Score |
|---|---|---|---|---|
| **0** | 1 | 19 | 15.0 | 39 |
| **1** | 1 | 21 | 15.0 | 81 |
| **2** | 0 | 20 | 16.0 | 6 |
| **3** | 0 | 23 | 16.0 | 77 |
| **4** | 0 | 31 | 17.0 | 40 |

In [29]:

```
#scalling data
from sklearn.preprocessing import MinMaxScaler

scale=MinMaxScaler()
#scalingData=scale.fit_transform(data.iloc[:,:3])
scalingData=scale.fit_transform(data)
scalingData
```

```
Out[29]: array([[1.        , 0.01923077, 0.        , 0.3877551 ],
       [1.        , 0.05769231, 0.        , 0.81632653],
       [0.        , 0.03846154, 0.00900901, 0.05102041],
       [0.        , 0.09615385, 0.00900901, 0.7755102 ],
       [0.        , 0.25      , 0.01801802, 0.39795918],
       [0.        , 0.07692308, 0.01801802, 0.76530612],
       [0.        , 0.32692308, 0.02702703, 0.05102041],
       [0.        , 0.09615385, 0.02702703, 0.94897959],
       [1.        , 0.88461538, 0.03603604, 0.02040816],
       [0.        , 0.23076923, 0.03603604, 0.7244898 ],
       [1.        , 0.94230769, 0.03603604, 0.13265306],
       [0.        , 0.32692308, 0.03603604, 1.        ],
       [0.        , 0.76923077, 0.04504505, 0.14285714],
       [0.        , 0.11538462, 0.04504505, 0.7755102 ],
       [1.        , 0.36538462, 0.04504505, 0.12244898],
       [1.        , 0.07692308, 0.04504505, 0.79591837],
       [0.        , 0.32692308, 0.05405405, 0.34693878],
       [1.        , 0.03846154, 0.05405405, 0.66326531],
       [1.        , 0.65384615, 0.07207207, 0.28571429],
       [0.        , 0.32692308, 0.07207207, 0.98979592],
       [1.        , 0.32692308, 0.08108108, 0.34693878],
       [1.        , 0.13461538, 0.08108108, 0.73469388],
       [0.        , 0.53846154, 0.09009009, 0.04081633],
       [1.        , 0.25      , 0.09009009, 0.73469388],
       [0.        , 0.69230769, 0.11711712, 0.13265306],
       [1.        , 0.21153846, 0.11711712, 0.82653061],
       [0.        , 0.51923077, 0.11711712, 0.31632653],
       [1.        , 0.32692308, 0.11711712, 0.6122449 ],
       [0.        , 0.42307692, 0.12612613, 0.30612245],
       [0.        , 0.09615385, 0.12612613, 0.87755102],
       [1.        , 0.80769231, 0.13513514, 0.03061224],
       [0.        , 0.05769231, 0.13513514, 0.73469388],
       [1.        , 0.67307692, 0.16216216, 0.03061224],
       [1.        , 0.        , 0.16216216, 0.92857143],
       [0.        , 0.59615385, 0.16216216, 0.13265306],
       [0.        , 0.05769231, 0.16216216, 0.81632653],
       [0.        , 0.46153846, 0.17117117, 0.16326531],
       [0.        , 0.23076923, 0.17117117, 0.73469388],
       [0.        , 0.34615385, 0.1981982 , 0.25510204],
       [0.        , 0.03846154, 0.1981982 , 0.75510204],
       [0.        , 0.90384615, 0.20720721, 0.34693878],
       [1.        , 0.11538462, 0.20720721, 0.92857143],
       [1.        , 0.57692308, 0.21621622, 0.35714286],
       [0.        , 0.25      , 0.21621622, 0.6122449 ],
       [0.        , 0.59615385, 0.21621622, 0.2755102 ],
       [0.        , 0.11538462, 0.21621622, 0.65306122],
       [0.        , 0.61538462, 0.22522523, 0.55102041],
       [0.        , 0.17307692, 0.22522523, 0.46938776],
       [0.        , 0.21153846, 0.22522523, 0.41836735],
       [0.        , 0.25      , 0.22522523, 0.41836735],
       [0.        , 0.59615385, 0.24324324, 0.52040816],
       [1.        , 0.28846154, 0.24324324, 0.60204082],
       [0.        , 0.25      , 0.25225225, 0.54081633],
       [1.        , 0.78846154, 0.25225225, 0.60204082],
       [0.        , 0.61538462, 0.25225225, 0.44897959],
       [1.        , 0.55769231, 0.25225225, 0.40816327],
       [0.        , 0.63461538, 0.26126126, 0.5       ],
       [1.        , 0.98076923, 0.26126126, 0.45918367],
       [0.        , 0.17307692, 0.27927928, 0.51020408],
       [1.        , 0.67307692, 0.27927928, 0.45918367],
       [1.        , 1.        , 0.27927928, 0.56122449],
       [1.        , 0.01923077, 0.27927928, 0.55102041],
       [0.        , 0.94230769, 0.28828829, 0.52040816],
       [0.        , 0.69230769, 0.28828829, 0.59183673],
```

```
       [1.        , 0.86538462, 0.2972973 , 0.51020408],
       [1.        , 0.        , 0.2972973 , 0.59183673],
       [0.        , 0.48076923, 0.2972973 , 0.5       ],
       [0.        , 0.96153846, 0.2972973 , 0.47959184],
       [1.        , 0.01923077, 0.2972973 , 0.59183673],
       [0.        , 0.26923077, 0.2972973 , 0.46938776],
       [1.        , 1.        , 0.30630631, 0.55102041],
       [0.        , 0.55769231, 0.30630631, 0.41836735],
       [0.        , 0.80769231, 0.31531532, 0.48979592],
       [0.        , 0.80769231, 0.31531532, 0.56122449],
       [1.        , 0.78846154, 0.35135135, 0.46938776],
       [1.        , 0.15384615, 0.35135135, 0.54081633],
       [0.        , 0.51923077, 0.35135135, 0.53061224],
       [1.        , 0.42307692, 0.35135135, 0.47959184],
       [0.        , 0.09615385, 0.35135135, 0.52040816],
       [0.        , 0.59615385, 0.35135135, 0.41836735],
       [1.        , 0.75      , 0.35135135, 0.51020408],
       [1.        , 0.38461538, 0.35135135, 0.55102041],
       [1.        , 0.94230769, 0.35135135, 0.40816327],
       [0.        , 0.53846154, 0.35135135, 0.43877551],
       [0.        , 0.05769231, 0.35135135, 0.57142857],
       [1.        , 0.57692308, 0.35135135, 0.45918367],
       [0.        , 0.71153846, 0.37837838, 0.58163265],
       [0.        , 0.07692308, 0.37837838, 0.55102041],
       [0.        , 0.30769231, 0.38738739, 0.60204082],
       [0.        , 0.61538462, 0.38738739, 0.45918367],
       [0.        , 0.96153846, 0.3963964 , 0.55102041],
       [1.        , 0.        , 0.3963964 , 0.40816327],
       [1.        , 0.57692308, 0.40540541, 0.48979592],
       [0.        , 0.42307692, 0.40540541, 0.39795918],
       [0.        , 0.26923077, 0.40540541, 0.41836735],
       [1.        , 0.11538462, 0.40540541, 0.52040816],
       [0.        , 0.55769231, 0.40540541, 0.46938776],
       [0.        , 0.17307692, 0.40540541, 0.5       ],
       [1.        , 0.57692308, 0.41441441, 0.41836735],
       [1.        , 0.03846154, 0.41441441, 0.48979592],
       [0.        , 0.09615385, 0.42342342, 0.40816327],
       [0.        , 0.59615385, 0.42342342, 0.47959184],
       [1.        , 0.94230769, 0.42342342, 0.59183673],
       [1.        , 0.15384615, 0.42342342, 0.55102041],
       [1.        , 0.59615385, 0.42342342, 0.56122449],
       [0.        , 0.05769231, 0.42342342, 0.41836735],
       [0.        , 0.92307692, 0.43243243, 0.5       ],
       [1.        , 0.69230769, 0.43243243, 0.45918367],
       [1.        , 0.96153846, 0.43243243, 0.42857143],
       [1.        , 0.92307692, 0.43243243, 0.47959184],
       [1.        , 0.90384615, 0.43243243, 0.52040816],
       [0.        , 0.01923077, 0.43243243, 0.54081633],
       [0.        , 0.38461538, 0.44144144, 0.41836735],
       [1.        , 0.01923077, 0.44144144, 0.45918367],
       [0.        , 0.        , 0.45045045, 0.47959184],
       [0.        , 0.01923077, 0.45045045, 0.5       ],
       [0.        , 0.86538462, 0.45045045, 0.42857143],
       [0.        , 0.59615385, 0.45045045, 0.59183673],
       [0.        , 0.63461538, 0.46846847, 0.42857143],
       [0.        , 0.61538462, 0.46846847, 0.57142857],
       [1.        , 0.17307692, 0.46846847, 0.56122449],
       [0.        , 0.38461538, 0.46846847, 0.39795918],
       [0.        , 0.42307692, 0.48648649, 0.58163265],
       [1.        , 0.40384615, 0.48648649, 0.91836735],
       [0.        , 0.09615385, 0.4954955 , 0.28571429],
       [0.        , 0.25      , 0.4954955 , 0.7755102 ],
       [1.        , 0.48076923, 0.5045045 , 0.34693878],
       [1.        , 0.42307692, 0.5045045 , 0.95918367],
```

```
       [1.        , 0.78846154, 0.5045045 , 0.10204082],
       [1.        , 0.38461538, 0.5045045 , 0.75510204],
       [1.        , 0.55769231, 0.5045045 , 0.08163265],
       [1.        , 0.40384615, 0.5045045 , 0.75510204],
       [0.        , 0.13461538, 0.51351351, 0.33673469],
       [0.        , 0.25      , 0.51351351, 0.71428571],
       [1.        , 0.03846154, 0.52252252, 0.04081633],
       [0.        , 0.21153846, 0.52252252, 0.8877551 ],
       [0.        , 0.5       , 0.52252252, 0.06122449],
       [1.        , 0.26923077, 0.52252252, 0.73469388],
       [1.        , 0.01923077, 0.53153153, 0.09183673],
       [0.        , 0.32692308, 0.53153153, 0.7244898 ],
       [0.        , 0.75      , 0.54054054, 0.04081633],
       [1.        , 0.26923077, 0.54054054, 0.93877551],
       [0.        , 0.19230769, 0.54954955, 0.39795918],
       [0.        , 0.26923077, 0.54954955, 0.87755102],
       [1.        , 0.13461538, 0.55855856, 0.1122449 ],
       [1.        , 0.19230769, 0.55855856, 0.97959184],
       [1.        , 0.57692308, 0.55855856, 0.35714286],
       [0.        , 0.26923077, 0.55855856, 0.74489796],
       [0.        , 0.30769231, 0.56756757, 0.21428571],
       [1.        , 0.30769231, 0.56756757, 0.90816327],
       [1.        , 0.48076923, 0.56756757, 0.16326531],
       [1.        , 0.40384615, 0.56756757, 0.8877551 ],
       [0.        , 0.5       , 0.56756757, 0.19387755],
       [0.        , 0.38461538, 0.56756757, 0.76530612],
       [0.        , 0.55769231, 0.56756757, 0.15306122],
       [0.        , 0.17307692, 0.56756757, 0.89795918],
       [1.        , 0.36538462, 0.56756757, 0.        ],
       [0.        , 0.23076923, 0.56756757, 0.78571429],
       [1.        , 0.30769231, 0.56756757, 0.        ],
       [0.        , 0.23076923, 0.56756757, 0.73469388],
       [0.        , 0.73076923, 0.57657658, 0.34693878],
       [0.        , 0.21153846, 0.57657658, 0.83673469],
       [1.        , 0.01923077, 0.59459459, 0.04081633],
       [0.        , 0.25      , 0.59459459, 0.93877551],
       [1.        , 0.61538462, 0.63063063, 0.25510204],
       [0.        , 0.34615385, 0.63063063, 0.75510204],
       [1.        , 0.46153846, 0.63963964, 0.19387755],
       [0.        , 0.28846154, 0.63963964, 0.95918367],
       [0.        , 0.34615385, 0.64864865, 0.26530612],
       [1.        , 0.26923077, 0.64864865, 0.63265306],
       [1.        , 0.42307692, 0.64864865, 0.12244898],
       [1.        , 0.19230769, 0.64864865, 0.75510204],
       [1.        , 0.34615385, 0.64864865, 0.09183673],
       [1.        , 0.34615385, 0.64864865, 0.92857143],
       [0.        , 0.65384615, 0.65765766, 0.12244898],
       [0.        , 0.23076923, 0.65765766, 0.86734694],
       [1.        , 0.76923077, 0.65765766, 0.14285714],
       [1.        , 0.17307692, 0.65765766, 0.69387755],
       [1.        , 0.78846154, 0.7027027 , 0.13265306],
       [1.        , 0.32692308, 0.7027027 , 0.90816327],
       [0.        , 0.36538462, 0.73873874, 0.31632653],
       [0.        , 0.26923077, 0.73873874, 0.86734694],
       [1.        , 0.53846154, 0.74774775, 0.14285714],
       [0.        , 0.21153846, 0.74774775, 0.8877551 ],
       [0.        , 0.44230769, 0.75675676, 0.3877551 ],
       [1.        , 0.23076923, 0.75675676, 0.97959184],
       [0.        , 0.69230769, 0.77477477, 0.23469388],
       [1.        , 0.19230769, 0.77477477, 0.68367347],
       [0.        , 0.44230769, 0.79279279, 0.16326531],
       [0.        , 0.34615385, 0.79279279, 0.85714286],
       [0.        , 0.30769231, 0.79279279, 0.2244898 ],
       [0.        , 0.26923077, 0.79279279, 0.69387755],
```

```
[1.        , 0.28846154, 0.88288288, 0.07142857],
[0.        , 0.38461538, 0.88288288, 0.91836735],
[0.        , 0.55769231, 0.94594595, 0.15306122],
[0.        , 0.32692308, 0.94594595, 0.79591837],
[0.        , 0.51923077, 1.        , 0.2755102 ],
[1.        , 0.26923077, 1.        , 0.74489796],
[1.        , 0.26923077, 0.41891892, 0.17346939],
[1.        , 0.23076923, 0.41891892, 0.83673469]])
```

# 9. Perform any of the clustering algorithms

In [30]:
```
from sklearn import cluster
```

In [31]:
```
error =[]
for i in range(1,11):
    kmeans=cluster.KMeans(n_clusters=i,init='k-means++',random_state=0)
    kmeans.fit(df)
    error.append(kmeans.inertia_)
```

In [32]:
```
error
```

Out[32]:
```
[963713.6749999999,
 381436.10486048606,
 267939.10708367854,
 191458.22767094016,
 153440.36158172583,
 119069.64011047289,
 101207.28033910532,
 85587.68268953268,
 76477.19921512422,
 68494.14545083063]
```

In [33]:
```
from sklearn.cluster import KMeans
TWSS=[]
k=list(range(2,12))

for i in k:
  kmeans=KMeans(n_clusters=i,init='k-means++')
  kmeans.fit(data)
  TWSS.append(kmeans.inertia_)

TWSS
```

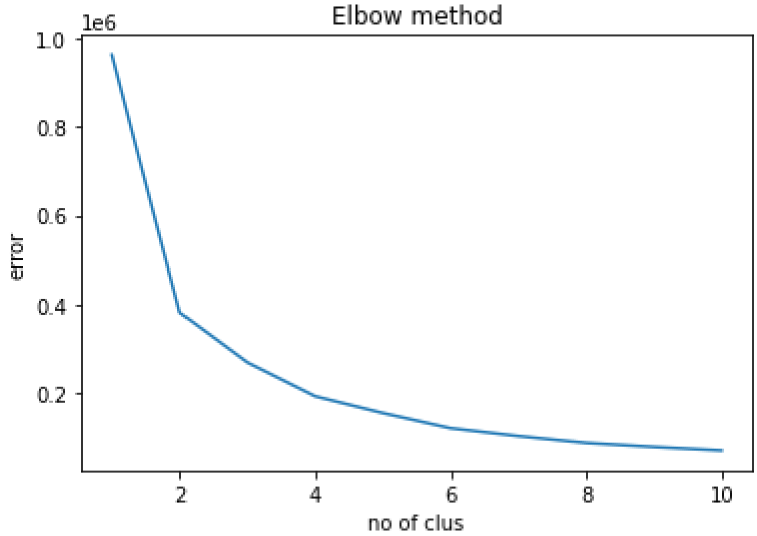
Out[33]:
```
[201152.1081841432,
 139326.23321730687,
 100349.31619915174,
 71419.31019600156,
 54455.93879921248,
 48692.47907464585,
 43313.71899120992,
 39664.19972596675,
 36439.15065511218,
 33228.41729951047]
```

In [34]:
```
#selecting  3 clusters
model=KMeans(n_clusters=3)
model.fit(data)
```

Out[34]:
```
KMeans(n_clusters=3)
```

# Visualization

In [35]:
```
import matplotlib.pyplot as plt
plt.plot(range(1,11),error)
plt.title('Elbow method')
plt.xlabel('no of clus')
plt.ylabel('error')
plt.show()
```




# 10. Add the cluster data with the primary dataset

In [36]:
```
#get the labels
model.labels_
```

Out[36]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 0, 2, 0, 2, 0, 2, 0,
       2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0,
       2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0,
       2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0,
       2, 0])
```

In [37]:
```
#converting in series
mb=pd.Series(model.labels_)
```

In [38]:
```
df['cluster'] =mb #kmeans.fit_predict(df)
```

In [39]:
```
df
```

Out[39]:

| | CustomerID | Gender | Age | Annual_Income | Spending_Score | cluster |
|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 19 | 15.0 | 39 | 1 |
| **1** | 2 | 1 | 21 | 15.0 | 81 | 1 |
| **2** | 3 | 0 | 20 | 16.0 | 6 | 1 |
| **3** | 4 | 0 | 23 | 16.0 | 77 | 1 |
| **4** | 5 | 0 | 31 | 17.0 | 40 | 1 |
| **...** | ... | ... | ... | ... | ... | ... |
| **195** | 196 | 0 | 35 | 120.0 | 79 | 0 |
| **196** | 197 | 0 | 45 | 126.0 | 28 | 2 |
| **197** | 198 | 1 | 32 | 126.0 | 74 | 0 |
| **198** | 199 | 1 | 32 | 61.5 | 18 | 2 |
| **199** | 200 | 1 | 30 | 61.5 | 83 | 0 |

200 rows × 6 columns

# 11. Split the data into dependent and independent variables.

# Independent variable

In [40]:

```
idv=df.iloc[:,:-1]#independent variables
idv
```

Out[40]:

| | CustomerID | Gender | Age | Annual_Income | Spending_Score |
|---|---|---|---|---|---|
| **0** | 1 | 1 | 19 | 15.0 | 39 |
| **1** | 2 | 1 | 21 | 15.0 | 81 |
| **2** | 3 | 0 | 20 | 16.0 | 6 |
| **3** | 4 | 0 | 23 | 16.0 | 77 |
| **4** | 5 | 0 | 31 | 17.0 | 40 |
| **...** | ... | ... | ... | ... | ... |
| **195** | 196 | 0 | 35 | 120.0 | 79 |
| **196** | 197 | 0 | 45 | 126.0 | 28 |
| **197** | 198 | 1 | 32 | 126.0 | 74 |
| **198** | 199 | 1 | 32 | 61.5 | 18 |
| **199** | 200 | 1 | 30 | 61.5 | 83 |

200 rows × 5 columns

# Dependent variable

In [41]:
```
dv=df.iloc[:,-1]#dependent variables
dv
```

Out[41]:
```
0      1
1      1
2      1
3      1
4      1
      ..
195    0
196    2
197    0
198    2
199    0
Name: cluster, Length: 200, dtype: int32
```

# 12. Split the data into training and testing

In [42]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(idv,dv,test_size=0.3,random_state=7
X_train,X_test
```

Out[42]:

```
(     CustomerID  Gender  Age  Annual_Income  Spending_Score
 88           89       0   34           58.0              60
 58           59       0   27           46.0              51
 113         114       1   19           64.0              46
 149         150       1   34           78.0              90
 36           37       0   42           34.0              17
 ..          ...     ...  ...            ...             ...
 151         152       1   39           78.0              88
 67           68       0   68           48.0              48
 25           26       1   29           28.0              82
 196         197       0   45          126.0              28
 175         176       0   30           88.0              86

 [140 rows x 5 columns],
      CustomerID  Gender  Age  Annual_Income  Spending_Score
 86           87       0   55           57.0              58
 120         121       1   27           67.0              56
 22           23       0   46           25.0               5
 11           12       0   35           19.0              99
 195         196       0   35          120.0              79
 2             3       0   20           16.0               6
 121         122       0   38           67.0              40
 94           95       0   32           60.0              42
 66           67       0   43           48.0              50
 63           64       0   54           47.0              59
 108         109       1   68           63.0              43
 96           97       0   47           60.0              47
 138         139       1   19           74.0              10
 65           66       1   18           48.0              59
 188         189       0   41          103.0              17
 155         156       0   27           78.0              89
 24           25       0   54           28.0              14
 99          100       1   20           61.0              49
 153         154       0   38           78.0              76
 46           47       0   50           40.0              55
 178         179       1   59           93.0              14
 139         140       0   35           74.0              72
 143         144       0   32           76.0              87
 74           75       1   59           54.0              47
 186         187       0   54          101.0              24
 169         170       1   32           87.0              63
 101         102       0   49           62.0              48
 197         198       1   32          126.0              74
 109         110       1   66           63.0              48
 177         178       1   27           88.0              69
 57           58       1   69           44.0              46
 106         107       0   66           63.0              50
 160         161       0   56           79.0              35
 84           85       0   21           54.0              57
 124         125       0   23           70.0              29
 85           86       1   48           54.0              46
 126         127       1   43           71.0              35
 183         184       0   29           98.0              88
 80           81       1   57           54.0              51
 116         117       0   63           65.0              43
 129         130       1   38           71.0              75
 128         129       1   59           71.0              11
 60           61       1   70           46.0              56
 122         123       0   40           69.0              58
 27           28       1   35           28.0              61
 40           41       0   65           38.0              35
 45           46       0   24           39.0              65
 191         192       0   32          103.0              69
 31           32       0   21           30.0              73
```

```
154          155      0   47          78.0             16
32            33      1   53          33.0              4
13            14      0   24          20.0             77
117          118      0   49          65.0             59
78            79      0   23          54.0             52
131          132      1   39          71.0             75
28            29      0   40          29.0             31
18            19      1   52          23.0             29
141          142      1   32          75.0             93
134          135      1   20          73.0              5
97            98      0   27          60.0             50)
```

In [43]:
```
print(y_train,y_test.shape)
```

```
88     1
58     1
113    1
149    0
36     1
      ..
151    0
67     1
25     1
196    2
175    0
Name: cluster, Length: 140, dtype: int32 (60,)
```

# 13. MODEL BUILDING

In [44]:
```
from sklearn import svm #SVM REFER SUPPORT VECTOR MACHINE
svm_model=svm.SVC(kernel='linear')
```

# 14. Train the Model

In [45]:
```
svm_model.fit(X_train,y_train)
```

Out[45]:
```
SVC(kernel='linear')
```

# 15. Test the Model

In [46]:
```
svm_pred=svm_model.predict(X_test)
svm_pred
```

Out[46]:
```
array([1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 0, 1, 1, 0, 1, 2, 0,
       0, 1, 2, 0, 1, 0, 1, 0, 1, 1, 2, 1, 2, 1, 2, 0, 1, 1, 0, 2, 1, 1,
       1, 1, 1, 0, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 2, 1])
```

# 16. Measure the performance using Evaluation Metrics

In [55]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
accuracy_score(y_test,svm_pred)
```

Out[55]:
```
1.0
```

In [56]:
```
metrics.confusion_matrix(y_test,svm_pred)
```

Out[56]:
```
array([[13,  0,  0],
       [ 0, 37,  0],
       [ 0,  0, 10]], dtype=int64)
```

In [53]:
```
pd.crosstab(y_test,svm_pred)
```

Out[53]:

| col_0 | 0 | 1 | 2 |
|---|---|---|---|
| **cluster** | | | |
| **0** | 13 | 0 | 0 |
| **1** | 0 | 37 | 0 |
| **2** | 0 | 0 | 10 |

In [54]:
```
print(classification_report(y_test,svm_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      1.00      1.00        37
           2       1.00      1.00      1.00        10

    accuracy                           1.00        60
   macro avg       1.00      1.00      1.00        60
weighted avg       1.00      1.00      1.00        60
```

In [ ]: