Assignment -4

Assignment Date	22 October 2022
Student Name	Renuka G
Student Roll Number	211419104220
Maximum Marks	2 Marks

Problem Statement: Customer Segmentation Analysis

Problem Statement

You own the mall and want to understand the customers who can quickly converge [Target Customers] so that the insight can be given to the marketing team and plan the strategy accordingly.

1. Download the dataset:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

2. Load the dataset.

In [2]:
```
data= pd.read_csv("F:Mall_Customers.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	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 [4]:
```
data.tail()
```

Out[4]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74

In [4]:
```
data.tail()
```
Out[4]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
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

In [5]:
```
data=data.iloc[:,1:]
```
In [6]:
```
data.head()
```
Out[6]:

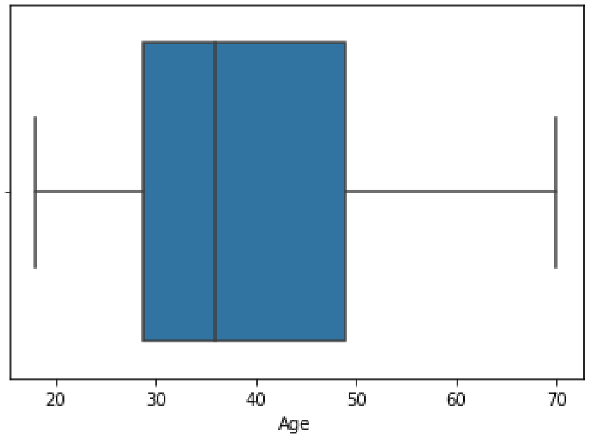
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40

3. Perform Below Visualizations

Univariate Analysis

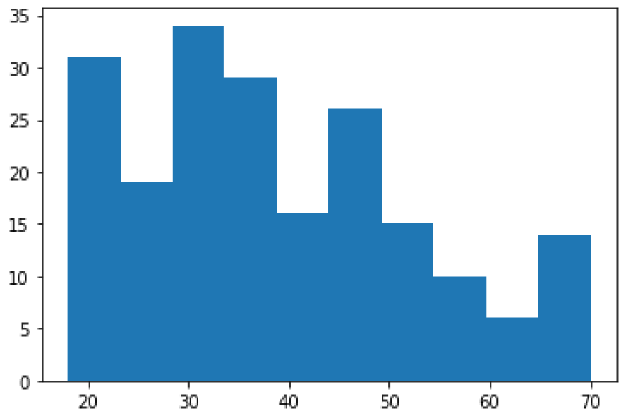
In [7]:
```
sns.boxplot(data['Age'])
```

Out[7]: <AxesSubplot:xlabel='Age'>

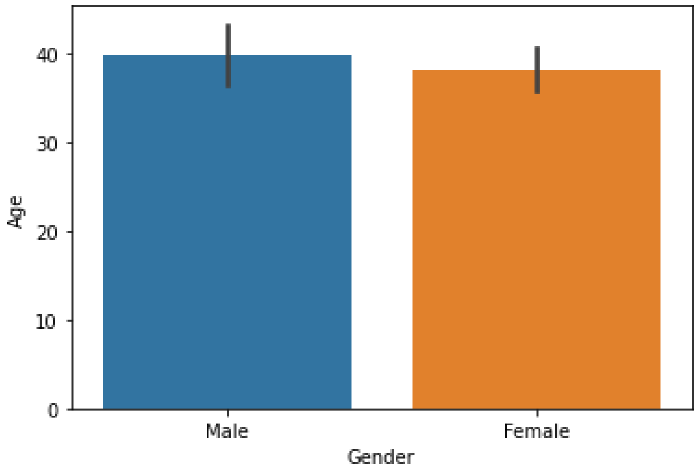


In [8]:
```
plt.hist(data['Age'])
```

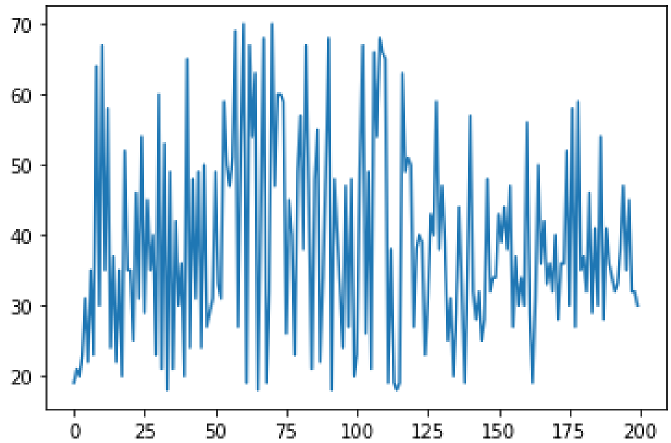
Out[8]:
```
(array([31., 19., 34., 29., 16., 26., 15., 10.,  6., 14.]),
 array([18. , 23.2, 28.4, 33.6, 38.8, 44. , 49.2, 54.4, 59.6, 64.8, 70. ]),
 <BarContainer object of 10 artists>)
```



In [9]:
```
sns.barplot(data['Gender'], data['Age'])
```
Out[9]: `<AxesSubplot:xlabel='Gender', ylabel='Age'>`

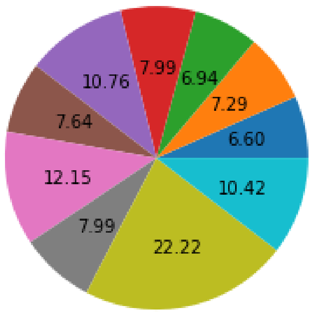


In [10]:
```
plt.plot(data['Age'])
```
Out[10]: `[<matplotlib.lines.Line2D at 0x1f4220bceb0>]`



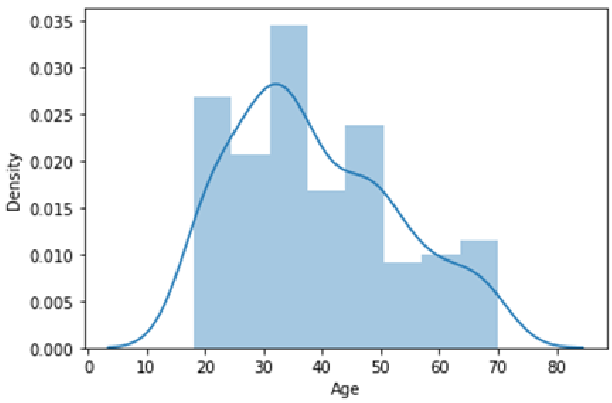
In [11]:
```
plt.pie(data['Age'].head(10), autopct="%.2f")
```

Out[11]:
```
([<matplotlib.patches.Wedge at 0x1f42212d970>,
  <matplotlib.patches.Wedge at 0x1f42213e130>,
  <matplotlib.patches.Wedge at 0x1f42213e850>,
  <matplotlib.patches.Wedge at 0x1f42213ef70>,
  <matplotlib.patches.Wedge at 0x1f42214b6d0>,
  <matplotlib.patches.Wedge at 0x1f42214bdf0>,
  <matplotlib.patches.Wedge at 0x1f422157550>,
  <matplotlib.patches.Wedge at 0x1f422157c70>,
  <matplotlib.patches.Wedge at 0x1f422165370>,
  <matplotlib.patches.Wedge at 0x1f422165a90>],
 [Text(1.07645875087365, 0.22635493736064408, ''),
  Text(0.8799412198934483, 0.6600783662054303, ''),
  Text(0.5079234694833851, 0.975711919138001, ''),
  Text(0.011998880619069147, 1.099934555718607, ''),
  Text(-0.601141928104972, 0.9212283795030332, ''),
  Text(-1.006842632792316, 0.443021345751915l, ''),
  Text(-1.07645875087365, -0.22635493736064374, ''),
  Text(-0.7342584950726593, -0.8190631614311766, ''),
  Text(0.23808358650393244, -1.0739256053551496, ''),
  Text(1.041623145203355З, -0.3535834037064947 7, '')],
 [Text(0.5871593186583546, 0.1234663294694422l, '6.60'),
  Text(0.4799679381236990З, 0.360042745202961 9, '7.29'),
  Text(0.2770491651727554 6, 0.532206501348000 5, '6.94'),
  Text(0.006544843974037716, 0.599964303119240 1, '7.99'),
  Text(-0.3278804688057257, 0.5024882070016544, '10.76'),
  Text(-0.5491868906139904, 0.24164800677377185, '7.64'),
  Text(-0.5871593186583546, -0.12346632946944203, '12.15'),
  Text(-0.4005046336759959, -0.446761724417005 4, '7.99'),
  Text(0.12986377445669042, -0.5857776029209907, '22.22'),
  Text(0.5681580792018301, -0.19286367474899713, '10.42')])
```



In [12]:
```
sns.distplot(data['Age'].head(200))
```

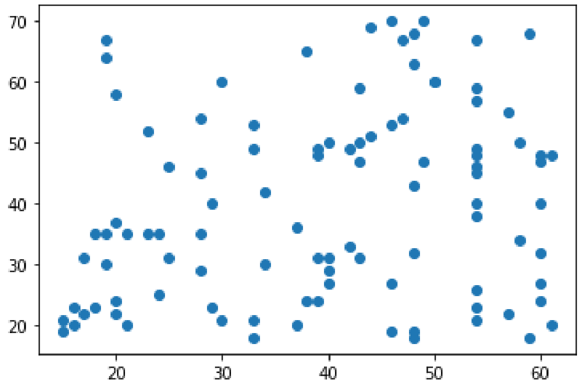
Out[12]:
```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



BI - Variate Analysis

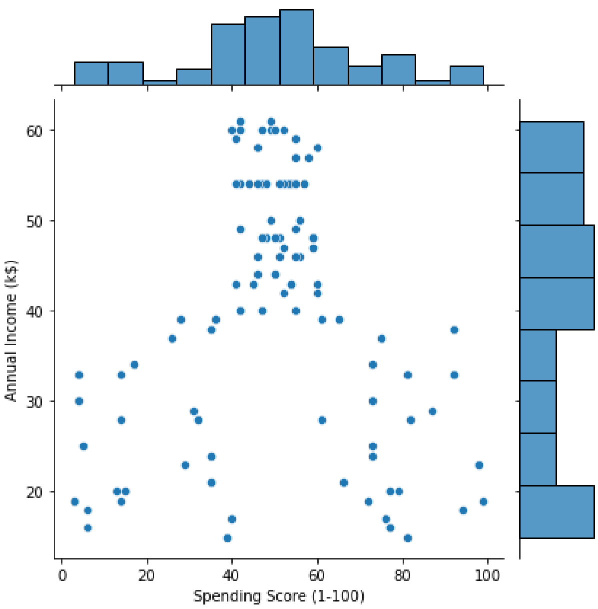
In [13]:
```
plt.scatter(data['Annual Income (k$)'].head(100),data['Age'].head(100))
```

Out[13]: `<matplotlib.collections.PathCollection at 0x1f42225e4c0>`



In [14]:
```
sns.jointplot(data['Spending Score (1-100)'].head(100) ,data['Annual Income (k$)'].head(100),  )
```

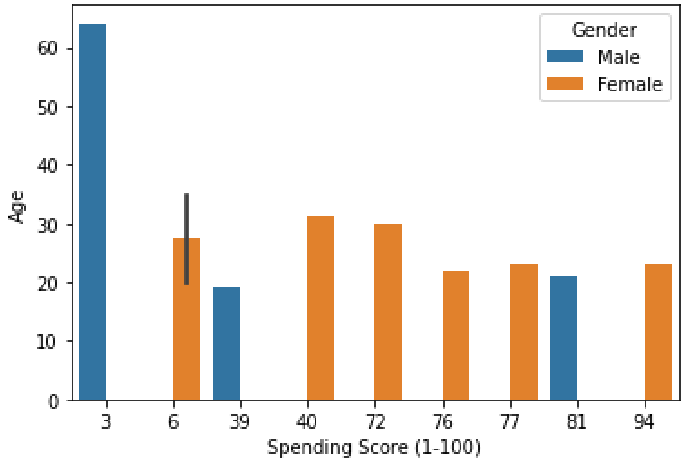
Out[14]: `<seaborn.axisgrid.JointGrid at 0x1f422283f70>`



Multi - Variate Analysis

In [15]:
```
sns.barplot('Spending Score (1-100)','Age',hue='Gender', data=data.head(10))
```

Out[15]: <AxesSubplot:xlabel='Spending Score (1-100)', ylabel='Age'>



In [16]:
```
sns.heatmap(data.head().corr(), annot = True)
```

Out[16]: <AxesSubplot:>

4. Perform descriptive statistics on the dataset.

In [17]:
```
data.mean()
```

Out[17]:
```
Age                       38.85
Annual Income (k$)        60.56
Spending Score (1-100)    50.20
dtype: float64
```

In [18]:
```
data.median()
```

Out[18]:
```
Age                       36.0
Annual Income (k$)        61.5
Spending Score (1-100)    50.0
dtype: float64
```

In [19]:
```
data.mode()
```

Out[19]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	Female	32.0	54	42.0
1	NaN	NaN	78	NaN

In [20]:
```
data.std()
```

Out[20]:
```
Age                       13.969007
Annual Income (k$)        26.264721
Spending Score (1-100)    25.823522
dtype: float64
```

In [21]:
```
data.var()
```

Out[21]:
```
Age                       195.133166
Annual Income (k$)        689.835578
Spending Score (1-100)    666.854271
dtype: float64
```

In [22]:
```
data.describe()
```

Out[22]:

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

In [23]:
```
data.skew()
```

Out[23]:
```
Age                       0.485569
Annual Income (k$)        0.321843
Spending Score (1-100)   -0.047220
dtype: float64
```

In [24]:
```
data.kurt()
```

Out[24]:
```
Age                      -0.671573
Annual Income (k$)       -0.098487
Spending Score (1-100)   -0.826629
dtype: float64
```

In [25]:
```
quantile= data['Age'].quantile(q=[0.75, 0.25])
quantile
```

Out[25]:
```
0.75    49.00
0.25    28.75
Name: Age, dtype: float64
```

5. Handle the Missing values.

In [26]:
```
data.isna().any()
```

Out[26]:
```
Gender                    False
Age                       False
Annual Income (k$)        False
Spending Score (1-100)    False
dtype: bool
```

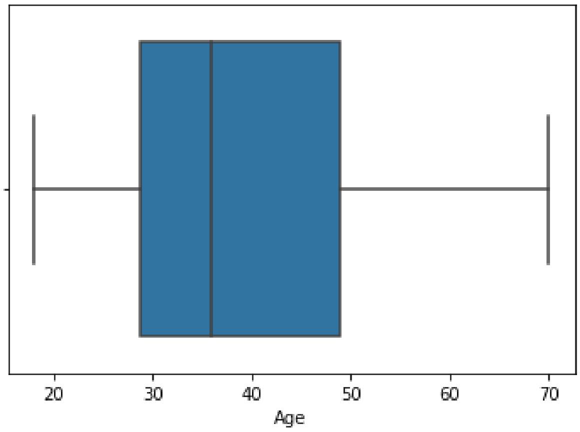
In [27]:
```
data.duplicated().any()
```

Out[27]:
```
False
```

6. Find the outliers and replace the outliers

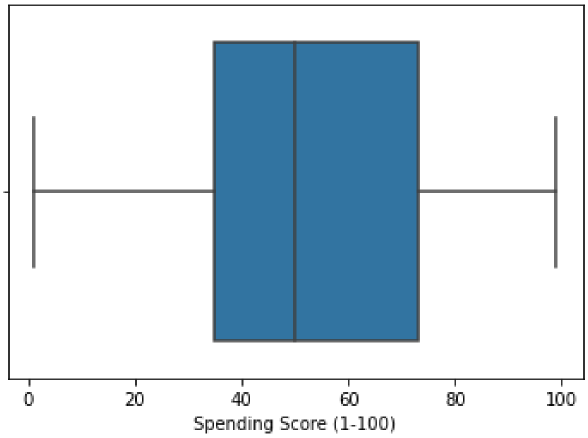
In [28]:
```
sns.boxplot(data['Age'])
```

Out[28]: <AxesSubplot:xlabel='Age'>



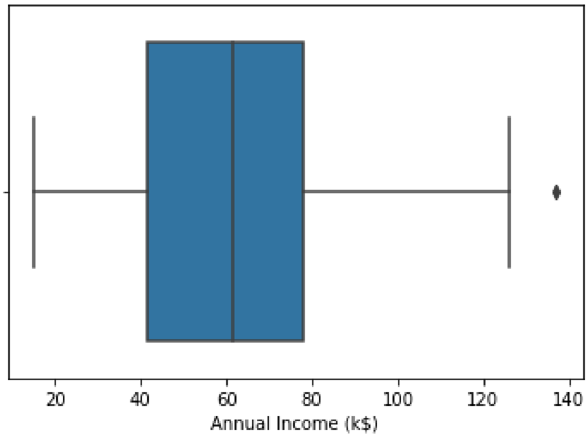
In [29]:
```
sns.boxplot(data['Spending Score (1-100)'])
```

Out[29]: <AxesSubplot:xlabel='Spending Score (1-100)'>



In [30]:
```
sns.boxplot(data['Annual Income (k$)'])
```

Out[30]: <AxesSubplot:xlabel='Annual Income (k$)'>



In [31]:
```
data['Annual Income (k$)'].mean()
```

Out[31]: 60.56

In [32]:
```
qut= data.quantile(q=[0.25,0.75])
qut
```

Out[32]:

	Age	Annual Income (k$)	Spending Score (1-100)
0.25	28.75	41.5	34.75
0.75	49.00	78.0	73.00

In [33]:
```
irq=qut.loc[0.75]- qut.loc[0.25] # q3-q1
irq
```

Out[33]:
```
Age                       20.25
Annual Income (k$)        36.50
Spending Score (1-100)    38.25
dtype: float64
```

In [34]:
```
lower= qut.loc[0.25]+(1.5*irq)
lower
```

Out[34]:
```
Age                       59.125
Annual Income (k$)        96.250
Spending Score (1-100)    92.125
dtype: float64
```

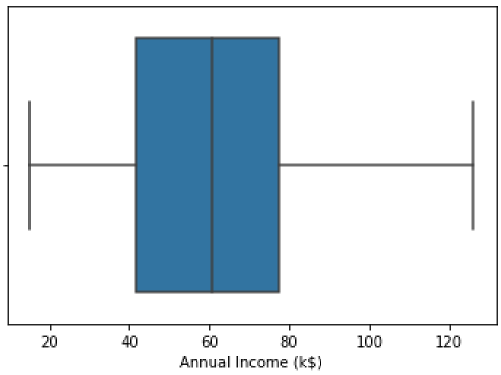
In [35]:
```
upper= qut.loc[0.75]+(1.5*irq)
upper
```

Out[35]:
```
Age                        79.375
Annual Income (k$)        132.750
Spending Score (1-100)    130.375
dtype: float64
```

In [36]:
```
data['Annual Income (k$)']=np.where(data['Annual Income (k$)']>131,60.56, data['Annual Income (k$)'])
```

In [37]:
```
sns.boxplot(data['Annual Income (k$)'])
```

Out[37]: <AxesSubplot:xlabel='Annual Income (k$)'>



7. Check for Categorical columns and perform encoding.

In [38]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Gender                  200 non-null    object
 1   Age                     200 non-null    int64
 2   Annual Income (k$)      200 non-null    float64
 3   Spending Score (1-100)  200 non-null    int64
dtypes: float64(1), int64(2), object(1)
memory usage: 6.4+ KB
```

In [39]:
```
data.Gender.unique()
```

Out[39]:
```
array(['Male', 'Female'], dtype=object)
```

In [40]:
```
data['Gender'].replace({'Female':0, 'Male': 1 }, inplace=True)
```

In [41]:
```
data.head()
```

Out[41]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
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

8. Scale the data

In [42]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [43]:
```
sc=MinMaxScaler()
```

In [44]:
```
data1=sc.fit_transform(data)
data1
```

```
       [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],
```

9. Perform any of the clustering algorithms

In [45]:
```
from sklearn.cluster import KMeans
```

In [46]:
```
TWSS=[]
k=list(range(2,9))

for i in k:
    kmeans=KMeans(n_clusters=i,init='k-means++')
    kmeans.fit(data1)
    TWSS.append(kmeans.inertia_)
```

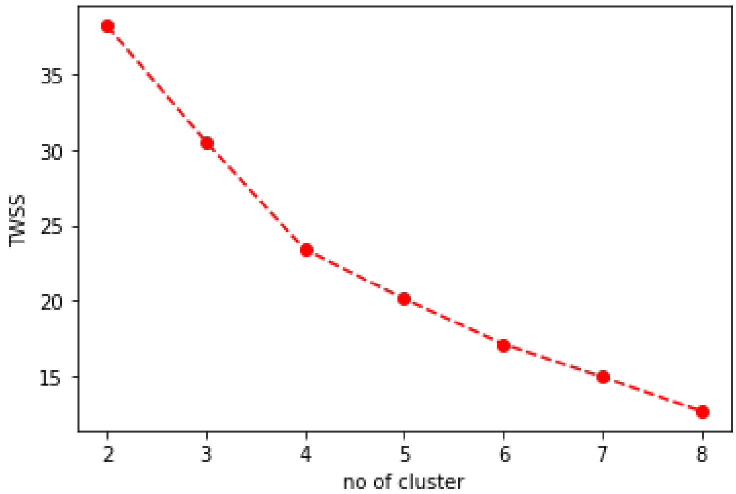
In [47]:
```
TWSS
```

Out[47]:
```
[38.256100195670456,
 30.535982817704678,
 23.373803580051828,
 20.099663171218396,
 17.095717979699913,
 14.90873757676125,
 12.635266048983096]
```

In [49]:
```
#scree plot

plt.plot(k,TWSS,'ro--')
plt.xlabel('no of cluster')
plt.ylabel('TWSS')
```

Out[49]:
```
Text(0, 0.5, 'TWSS')
```



In [50]:
```
#selecting 4 clusters

model=KMeans(n_clusters=4)

model.fit(data1)
```

Out[50]:
```
KMeans
KMeans(n_clusters=4)
```

In [51]:
```
model.labels_
```

Out[51]:
```
array([3, 3, 2, 0, 0, 0, 2, 0, 1, 0, 1, 0, 2, 0, 1, 3, 2, 3, 1, 0, 1, 3,
       2, 3, 2, 3, 2, 3, 2, 0, 1, 0, 1, 3, 2, 0, 2, 0, 2, 0, 2, 3, 1, 0,
       2, 0, 2, 0, 0, 0, 2, 3, 0, 1, 2, 1, 2, 1, 0, 1, 1, 3, 2, 2, 1, 3,
       2, 2, 3, 0, 1, 2, 2, 2, 1, 3, 2, 1, 0, 2, 1, 3, 1, 2, 0, 1, 2, 0,
       0, 2, 2, 3, 1, 2, 0, 3, 2, 0, 1, 3, 0, 2, 1, 3, 1, 0, 2, 1, 1, 1,
       1, 0, 2, 3, 0, 0, 2, 2, 2, 2, 3, 2, 0, 3, 0, 0, 1, 3, 1, 3, 1, 3,
       0, 0, 1, 0, 2, 3, 1, 0, 2, 3, 0, 0, 1, 3, 1, 0, 2, 3, 1, 3, 2, 0,
       2, 0, 1, 0, 1, 0, 2, 0, 1, 0, 1, 0, 1, 0, 2, 3, 1, 3, 1, 3, 2, 0,
       1, 3, 1, 3, 2, 0, 1, 0, 2, 3, 2, 3, 2, 0, 2, 0, 1, 0, 2, 0, 2, 3,
       1, 3])
```

In [52]:
```
mb=pd.Series(model.labels_)
```

10. Add the cluster data with the primary dataset

In [53]:
```
data.head()
```

Out[53]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
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 [54]:
```
#creating a new column with labels
data['clust']=mb
```

In [55]:
```
data.head()
```

Out[55]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	clust
0	1	19	15.0	39	3
1	1	21	15.0	81	3
2	0	20	16.0	6	2
3	0	23	16.0	77	0
4	0	31	17.0	40	0

In [56]:
```
data['clust'].unique()
```
Out[56]:
```
array([3, 2, 0, 1])
```
In [57]:
```
#storing the data set in csv format

data.to_csv('Kmeans_Mall.csv',encoding='utf-8')
```
In [58]:
```
import os
os.getcwd()
```
Out[58]:
```
'C:\\Users\\asdmin\\Downloads'
```

11. Split the data into dependent and independent variables.

In [149]:
```
x=data.iloc[:,:4]
y=data['clust']
x
```
Out[149]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19	15.00	39
1	1	21	15.00	81
2	0	20	16.00	6
3	0	23	16.00	77
4	0	31	17.00	40
...	...	...	...	...
195	0	35	120.00	79
196	0	45	126.00	28
197	1	32	126.00	74
198	1	32	60.56	18
199	1	30	60.56	83

200 rows × 4 columns

In [150]:
```
y
```

Out[150]:
```
0      3
1      3
2      2
3      0
4      0
      ..
195    0
196    2
197    3
198    1
199    3
Name: clust, Length: 200, dtype: int32
```

In [151]:
```
from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler()
```

In [152]:
```
x=sc.fit_transform(data1)
x
```

Out[152]:
```
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],
```

12. Split the data into training and testing

In [154]:
```
from sklearn.model_selection import train_test_split
```

In [155]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2)
```

In [156]:
```
print(x_train.shape, x_test.shape)
```

```
(160, 4) (40, 4)
```

13. Build the Model

In [157]:
```
from sklearn.tree import DecisionTreeRegressor
regr_1 = DecisionTreeRegressor(max_depth=5)
```

14. Train the Model

In [158]:
```
regr_1.fit(x_train, y_train)
```

Out[158]:
```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=5)
```

15. Test the Model

In [159]:
```
# predict on test data

y_predict = regr_1.predict(x_test)
y_predict
```

Out[159]:
```
array([1., 1., 2., 2., 2., 3., 1., 0., 3., 0., 2., 0., 2., 2., 0., 2., 3.,
       2., 0., 0., 3., 0., 2., 1., 0., 3., 0., 0., 0., 1., 0., 3., 3., 1.,
       2., 0., 0., 2., 0., 1.])
```

16. Measure the performance using Evaluation Metrics.

In [160]:
```
from sklearn.metrics import r2_score
acc=r2_score(y_test,y_predict)
acc
```

Out[160]: 0.9206349206349207

RandomForest

In [161]:
```
from sklearn.ensemble  import RandomForestRegressor
```

In [162]:
```
rf=RandomForestRegressor(max_depth=9)
```

In [163]:
```
rf.fit(x_train, y_train)
```

Out[163]:
```
RandomForestRegressor
RandomForestRegressor(max_depth=9)
```

In [164]:
```
## prediction

pred= rf.predict(x_test)
```

In [165]:
```
from sklearn.metrics import r2_score
acc=r2_score(y_test,pred)
acc
```

Out[165]: 0.9650171856440989