##Task 1: Download the dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings

warnings.filterwarnings('ignore')
```

##Task 2: Load the dataset into the tool

```
dataset=pd.read_csv("/content/Mall_Customers.csv")

dataset.head()
```

```
   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
```

```
dataset.tail()
```

```
     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
```

```
dataset.columns
```

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

##Task 3: Visualization

- Univariate Analysis

  barchart,histogram,pie chart,frequency polygram

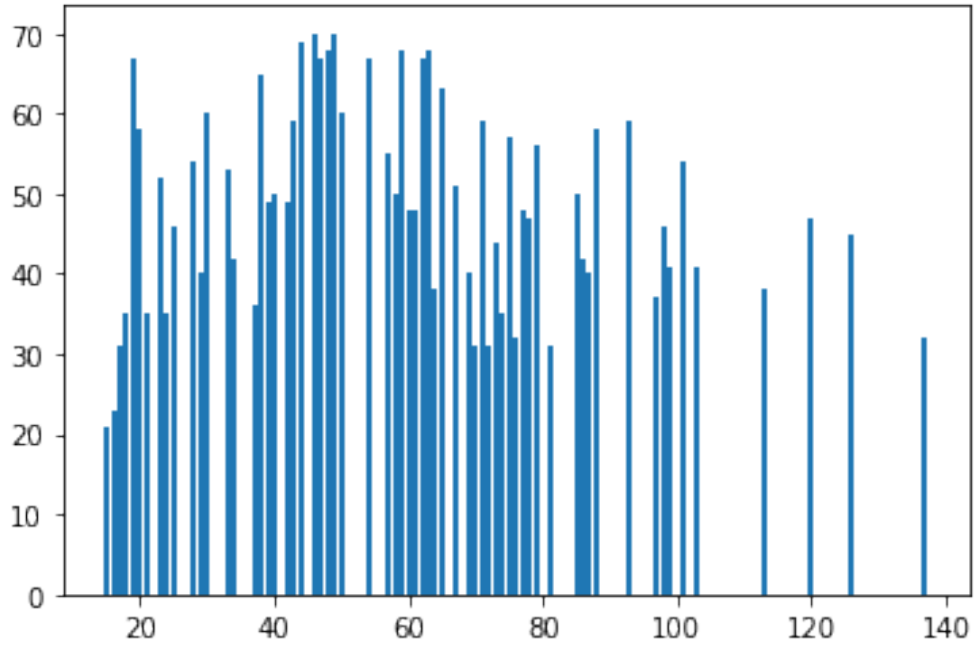
- Bi - Variate Analysis

  scatterplot,Linear correleation

- Multi - Variate Analysis

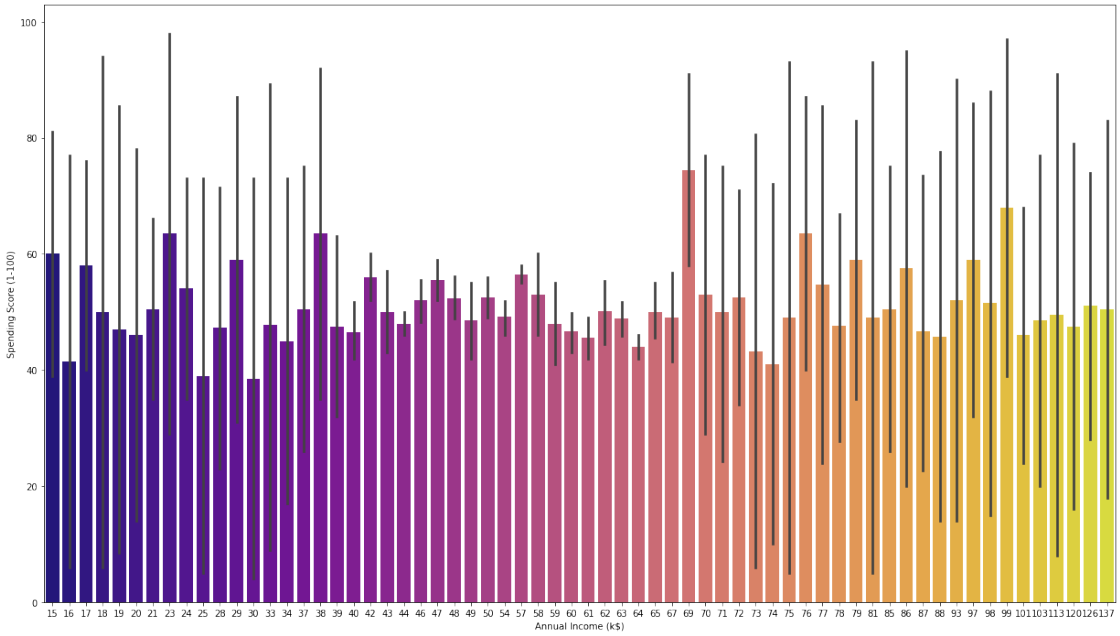
```
#barplot
plt.bar(data=dataset,height="Age",x="Annual Income (k$)")
```

```
<BarContainer object of 200 artists>
```



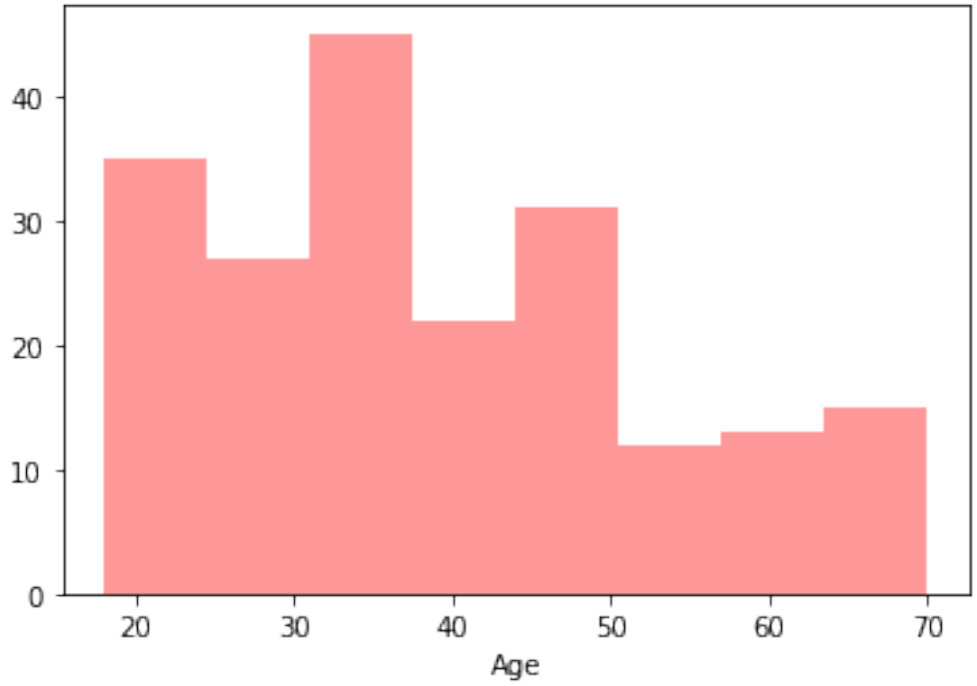

```
fig=plt.figure(figsize=(21,12))
sns.barplot(x ='Annual Income (k$)', y ='Spending Score (1-100)', data
= dataset,
            palette ='plasma')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb564f64550>
```



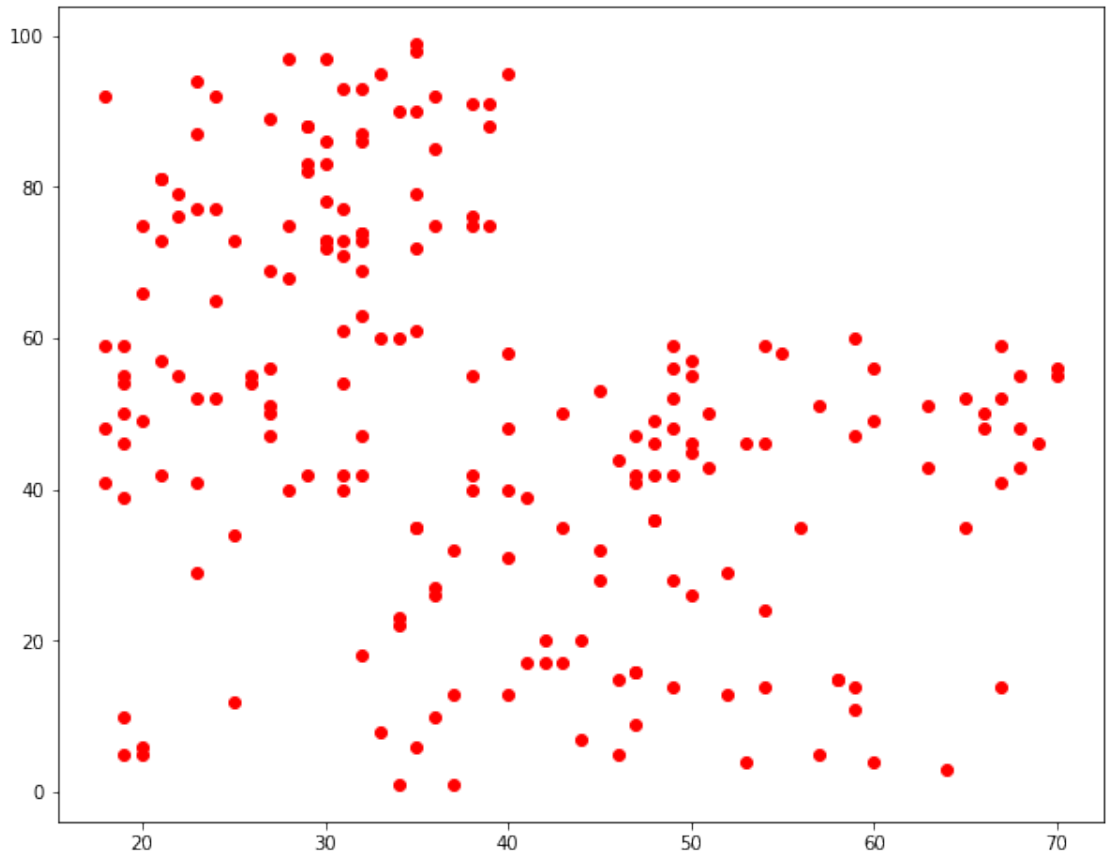

```
#histogram
sns.distplot(dataset["Age"],kde=False,color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb564d01bd0>
```



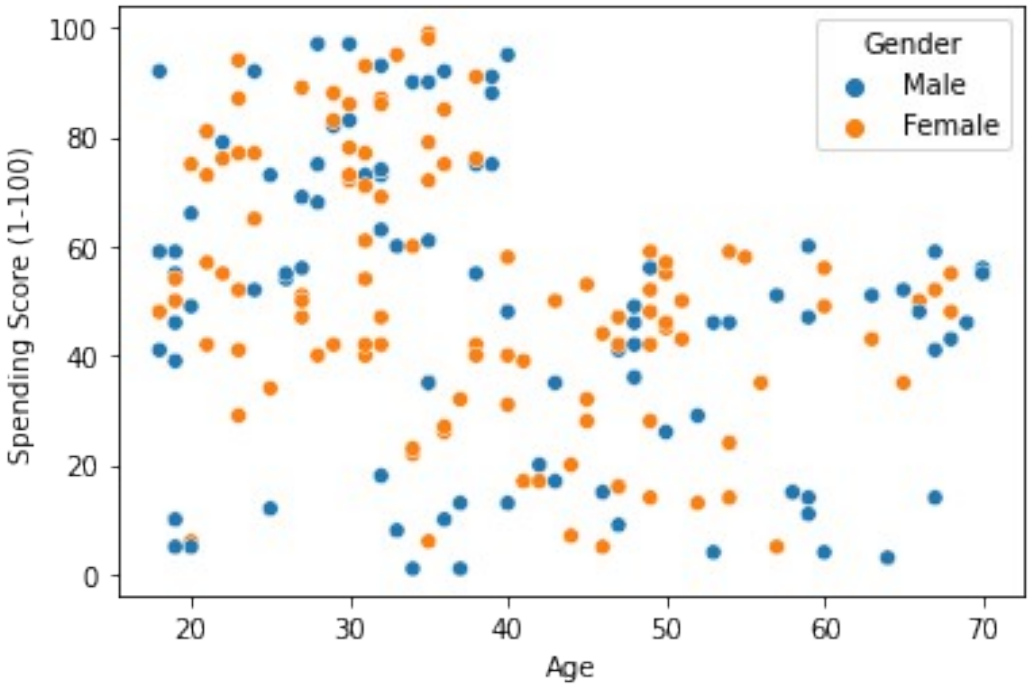

```
#scatterplot
fig=plt.figure(figsize=(10,8))
plt.scatter(dataset['Age'],dataset["Spending Score (1-
100)"],color='red')
```

```
<matplotlib.collections.PathCollection at 0x7fb564befc10>
```



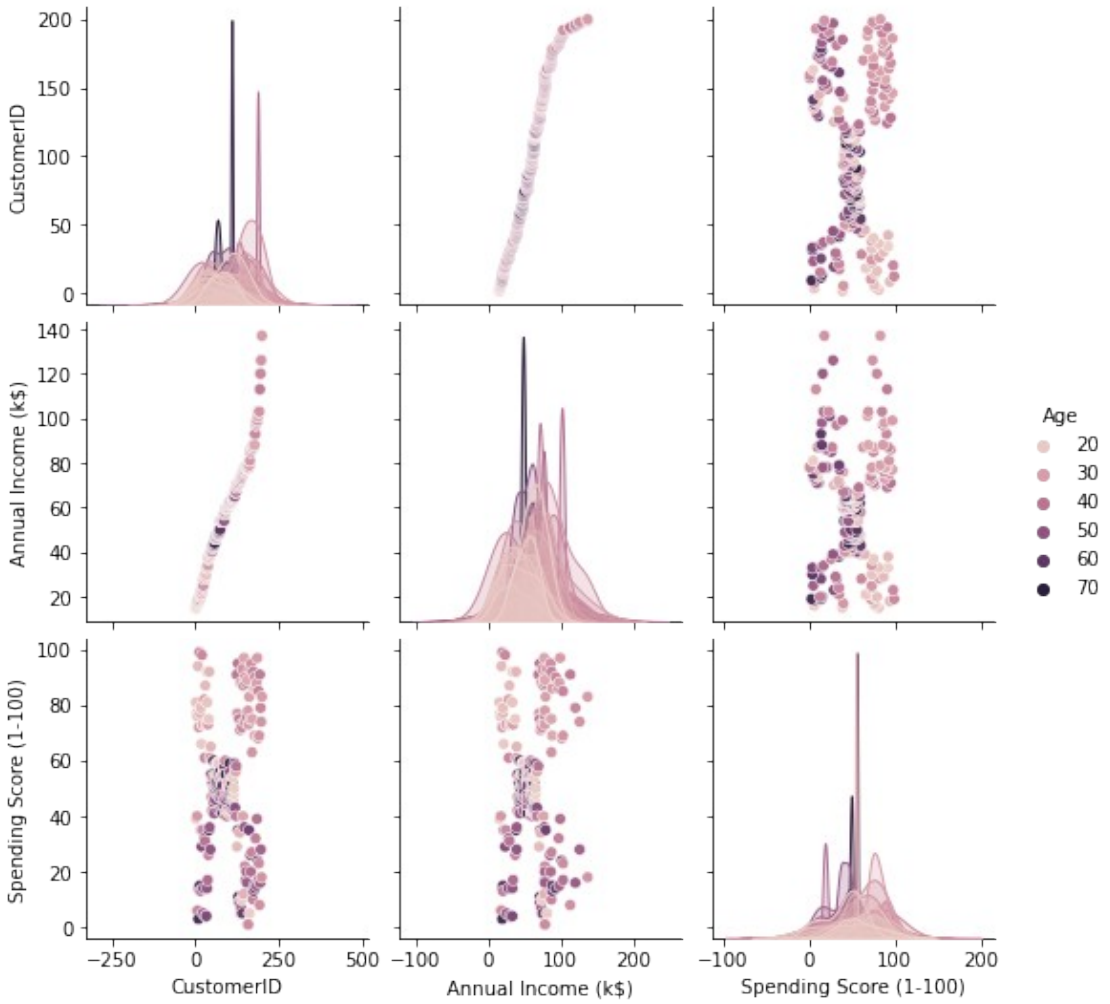

```
#scattorplot
sns.scatterplot(x='Age',y='Spending Score (1-
100)',hue='Gender',data=dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb564b8da90>
```



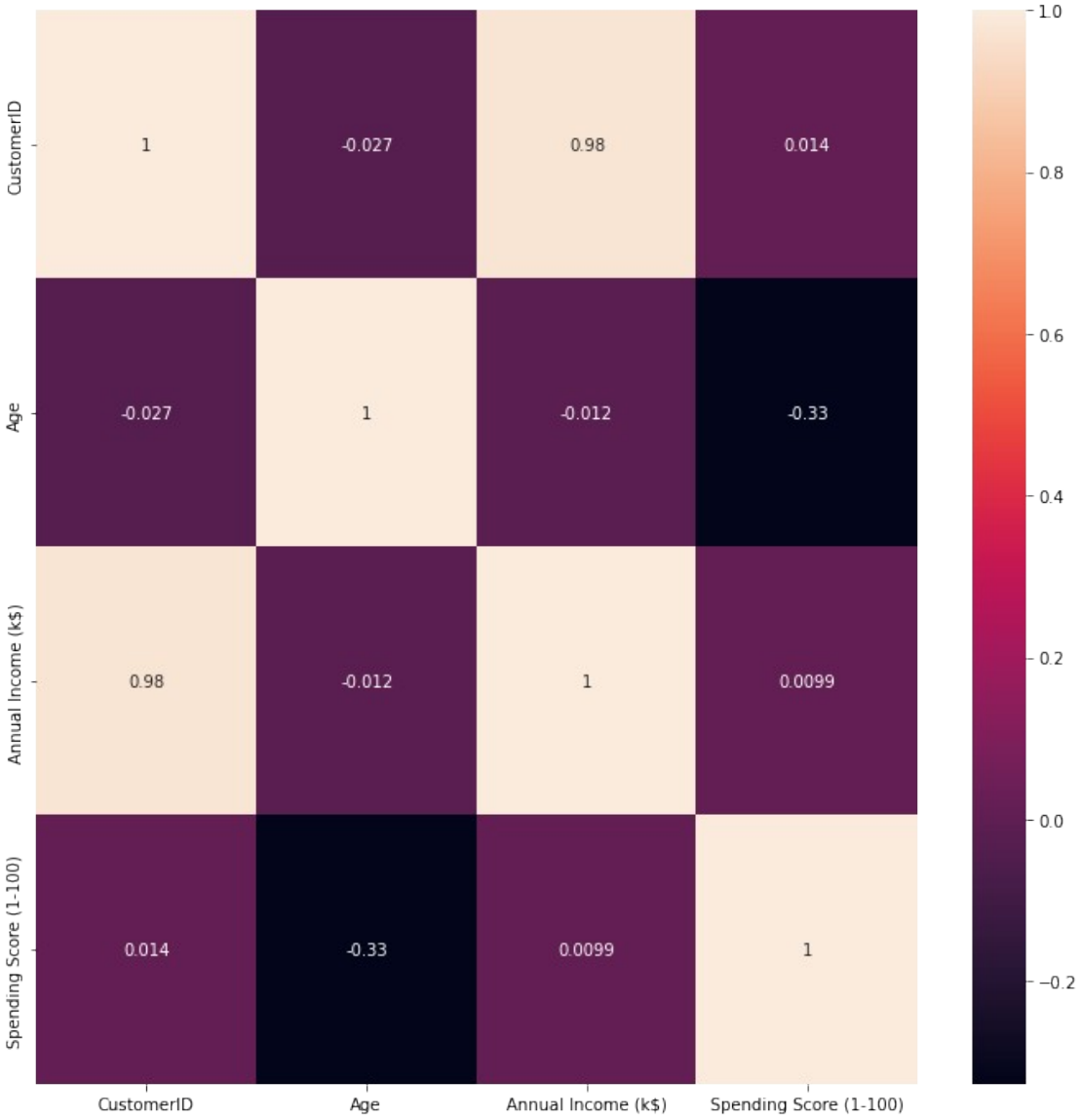

```
#pairplot
sns.pairplot(dataset,hue="Age")
```

```
<seaborn.axisgrid.PairGrid at 0x7fb564b30650>
```



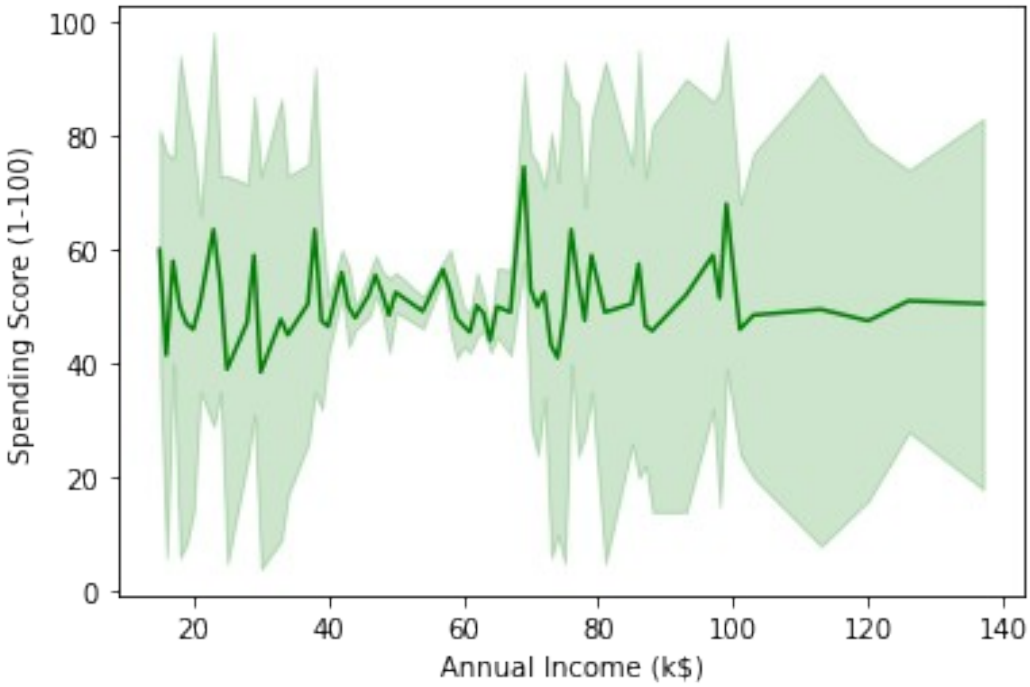

```
#heatmap
corr=dataset.corr()
fig=plt.figure(figsize=(12,12))
sns.heatmap(corr,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb564564e90>
```

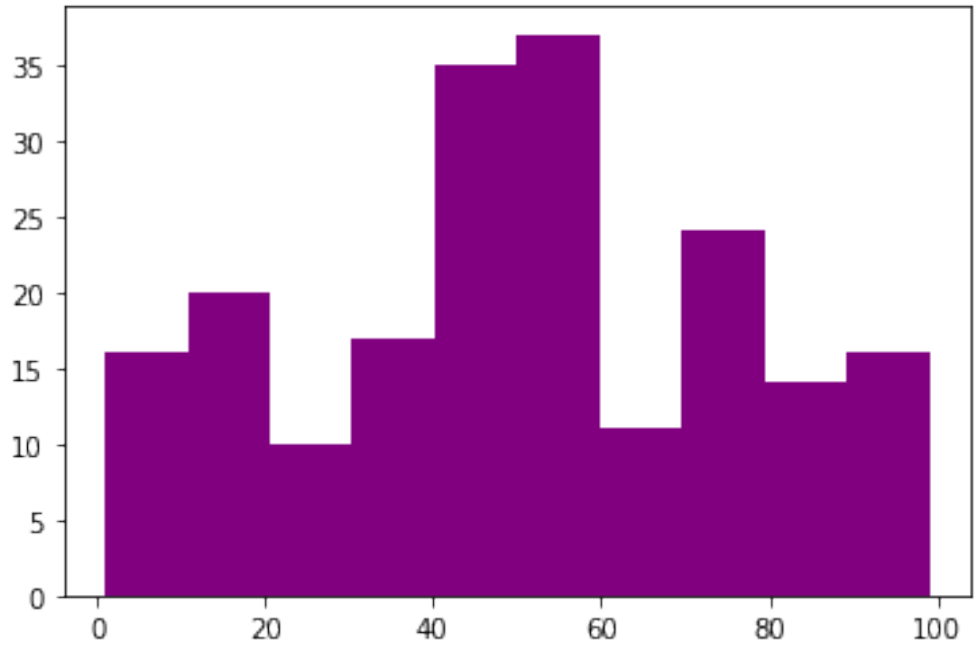



```
#lineplot
sns.lineplot(dataset['Annual Income (k$)'],dataset['Spending Score (1-100)'],color='green')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb564505650>
```

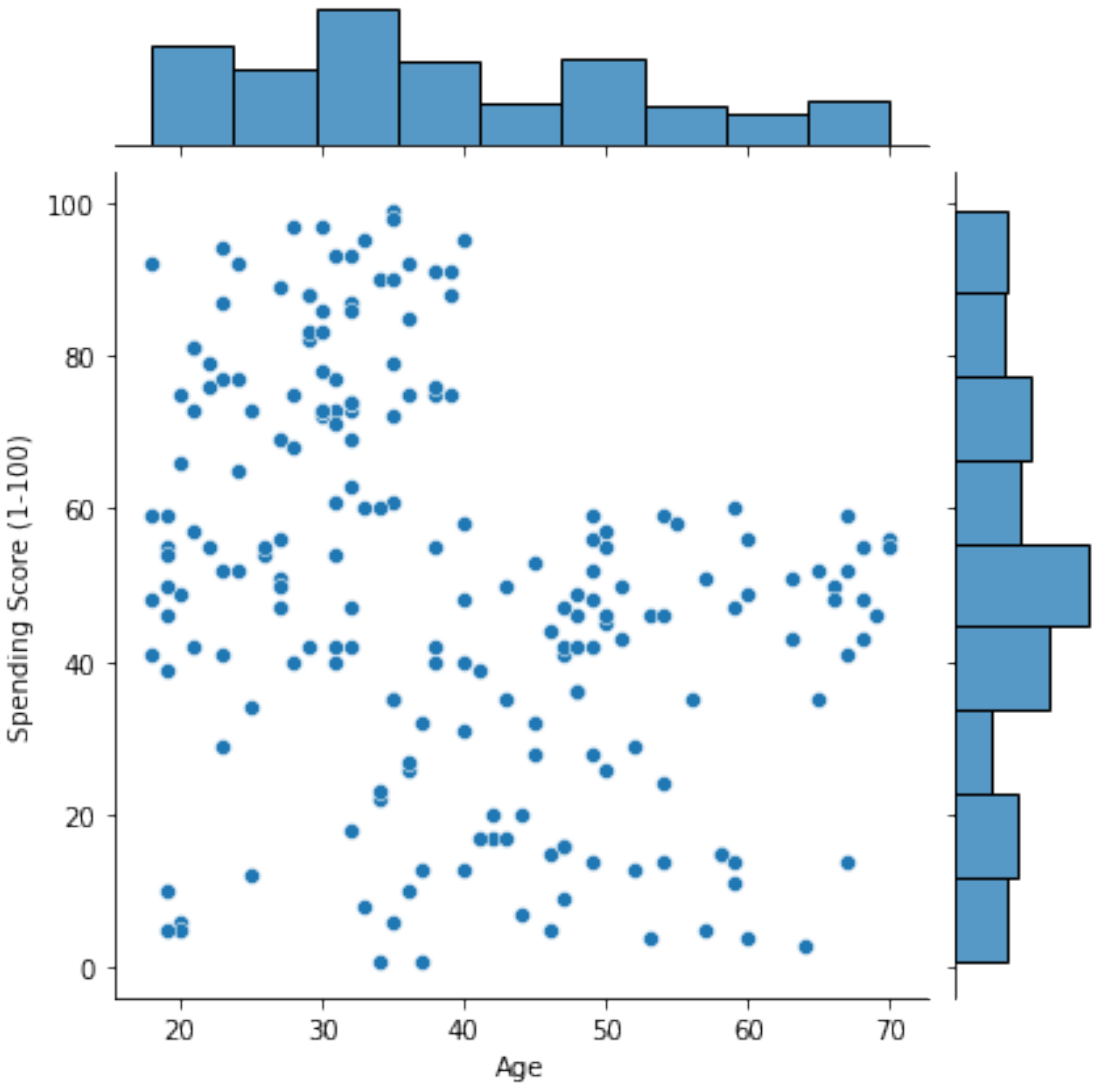



```
#histogram
plt.hist(dataset['Spending Score (1-100)'],color='purple')
plt.show()
```




```
#jointplot
sns.jointplot(dataset['Age'],dataset['Spending Score (1-100)'])
```

```
<seaborn.axisgrid.JointGrid at 0x7fb5643f0650>
```



##Task 4: Perform descriptive statistics on the dataset

```
dataset.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 (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
des=dataset.describe().T
des
```

```
                         count    mean        std   min    25%    50%
75%  \
CustomerID               200.0  100.50  57.879185   1.0  50.75  100.5
150.25
Age                      200.0   38.85  13.969007  18.0  28.75   36.0
49.00
Annual Income (k$)       200.0   60.56  26.264721  15.0  41.50   61.5
78.00
Spending Score (1-100)   200.0   50.20  25.823522   1.0  34.75   50.0
73.00

                           max
CustomerID               200.0
Age                       70.0
Annual Income (k$)       137.0
Spending Score (1-100)    99.0
```

##Task 5:Check for Missing values and deal with them

```
dataset.isna().sum().sum()
```

```
0
```

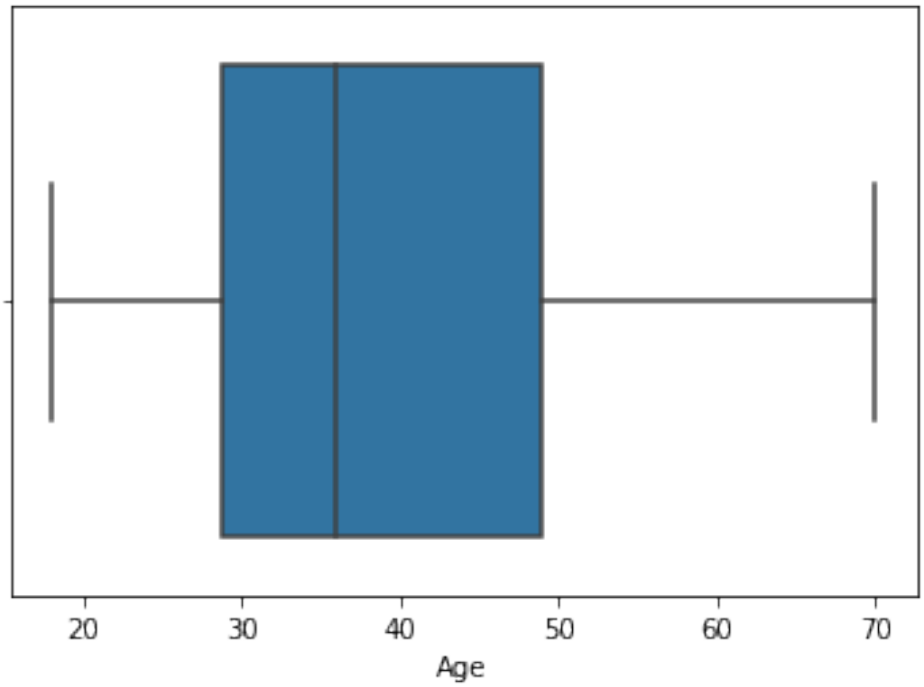
##Task 6:Find the outliers and replace them outliers

```
dataset.head()
```

```
   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
```

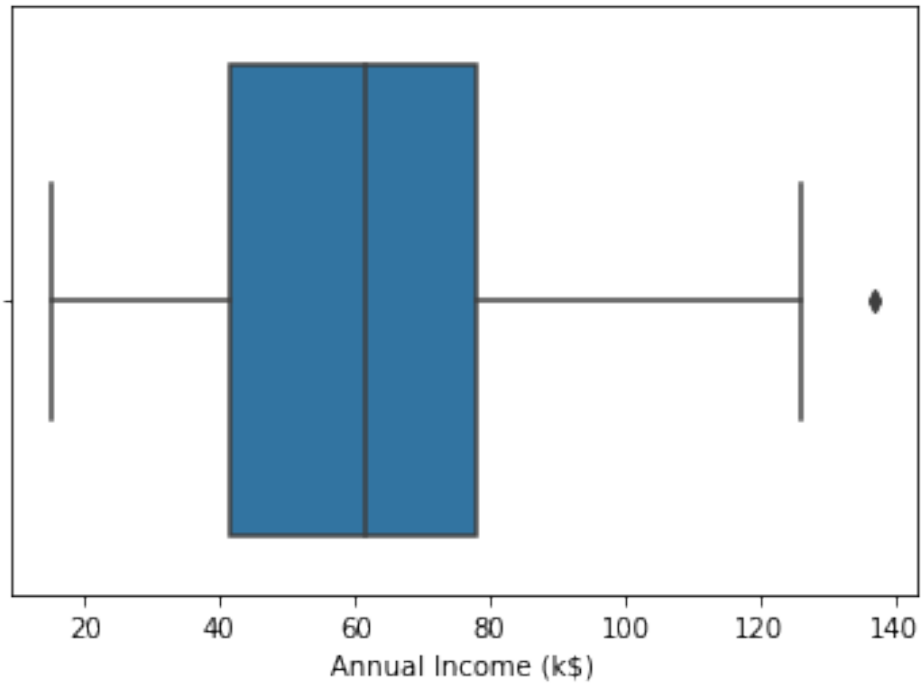
```
#checking outliers
sns.boxplot(dataset['Age'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb564247d50>
```



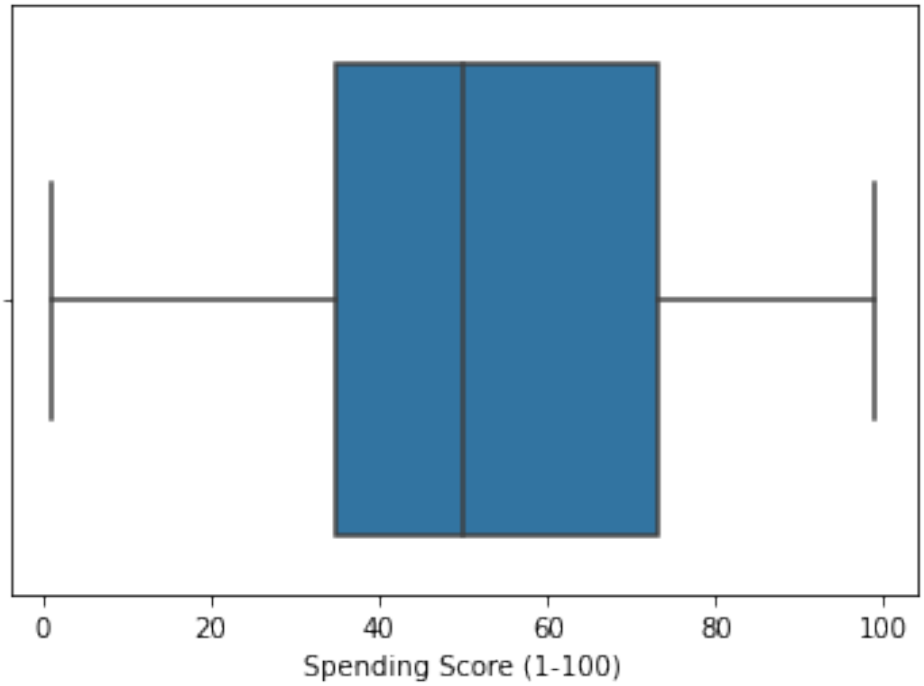

```
#checking outliers
sns.boxplot(dataset['Annual Income (k$)'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb564241890>
```



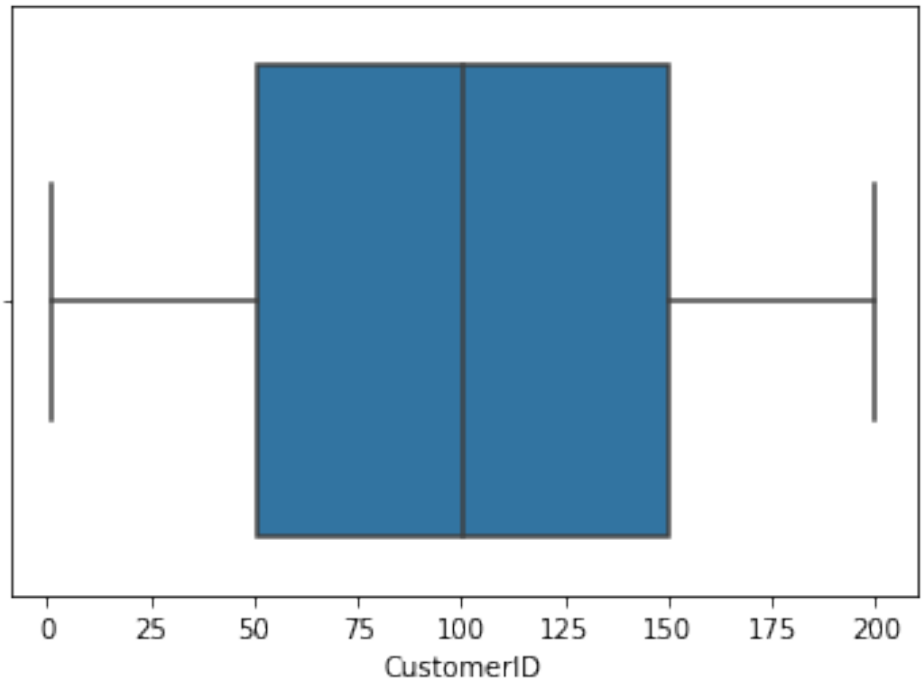

```
#checking outliers
sns.boxplot(dataset['Spending Score (1-100)'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5641a06d0>
```




```
#checking outliers
sns.boxplot(dataset['CustomerID'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb56411eb90>
```



```
#quantile
qnt=dataset.quantile(q=(0.75,0.25))
qnt
```

```
      CustomerID    Age  Annual Income (k$)  Spending Score (1-100)
0.75      150.25  49.00                78.0                   73.00
0.25       50.75  28.75                41.5                   34.75
```

```
#IQR calculations
iqr=qnt.loc[0.75]-qnt.loc[0.25]
iqr
```

```
CustomerID                99.50
Age                       20.25
Annual Income (k$)        36.50
Spending Score (1-100)    38.25
dtype: float64
```

```
#lower extreme value
lower=qnt.loc[0.25]-1.5*iqr
lower
```

```
CustomerID               -98.500
Age                       -1.625
Annual Income (k$)       -13.250
Spending Score (1-100)   -22.625
dtype: float64
```

```
#upper extreme value
upper=qnt.loc[0.75]+1.5*iqr
upper
```

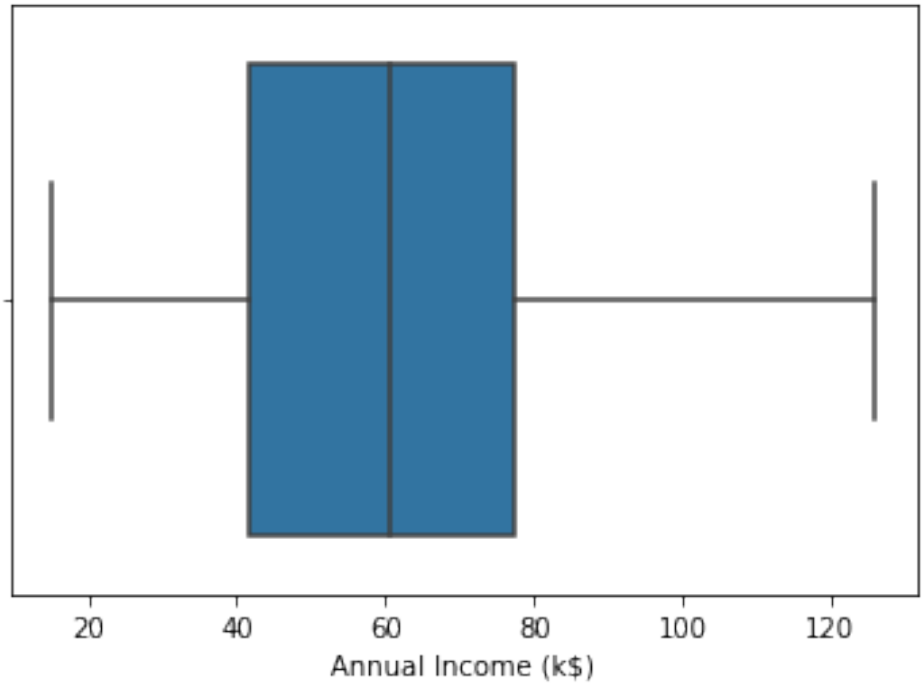
```
CustomerID                299.500
Age                        79.375
Annual Income (k$)        132.750
Spending Score (1-100)    130.375
dtype: float64
```

```
#mean value
dataset.mean()
```

```
CustomerID                100.50
Age                        38.85
Annual Income (k$)         60.56
Spending Score (1-100)     50.20
dtype: float64
```

```
#replacing the outliers
dataset['Annual Income (k$)']=np.where(dataset['Annual Income
(k$)']>132.750,60.56,dataset['Annual Income (k$)'])

sns.boxplot(dataset["Annual Income (k$)"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5640ab550>
```



##Task 7: Check for Categorical columns and perform encoding

```
dataset.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 (k$)      200 non-null    float64
 4   Spending Score (1-100)  200 non-null    int64  
dtypes: float64(1), int64(3), object(1)
memory usage: 7.9+ KB
```

```
dataset['Gender'].unique()
```

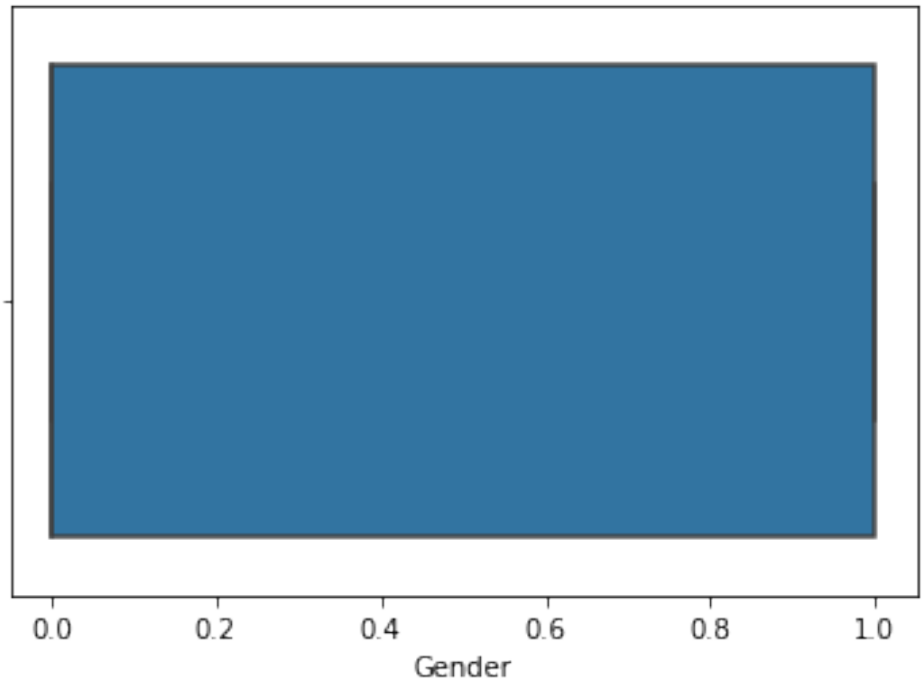
```
array(['Male', 'Female'], dtype=object)
```

```
#encoding
from sklearn.preprocessing import LabelEncoder
en=LabelEncoder()

dataset['Gender']=en.fit_transform(dataset['Gender'])

dataset.head()
```

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

```
sns.boxplot(dataset['Gender'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb564335e10>
```




```
#dropping the unwanted column
df=dataset.drop(['CustomerID'],axis=1)
```

```
df.head()
```

	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

```
df.tail()
```

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

```
df.shape
```

```
(200, 4)
```

```
df.duplicated().sum()
```

```
0
```

```
df.iloc[:,:-1]
```

```
     Gender  Age  Annual Income (k$)
0         1   19                15.00
1         1   21                15.00
2         0   20                16.00
3         0   23                16.00
4         0   31                17.00
..      ...  ...                  ...
195       0   35               120.00
196       0   45               126.00
197       1   32               126.00
198       1   32                60.56
199       1   30                60.56

[200 rows x 3 columns]
```

##Task 8: Scaling the data

```
#MinMaxScaling
from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler()

data=sc.fit_transform(df.iloc[:,:-1])
```

##Task 9: Perform any of the clustering algorithms

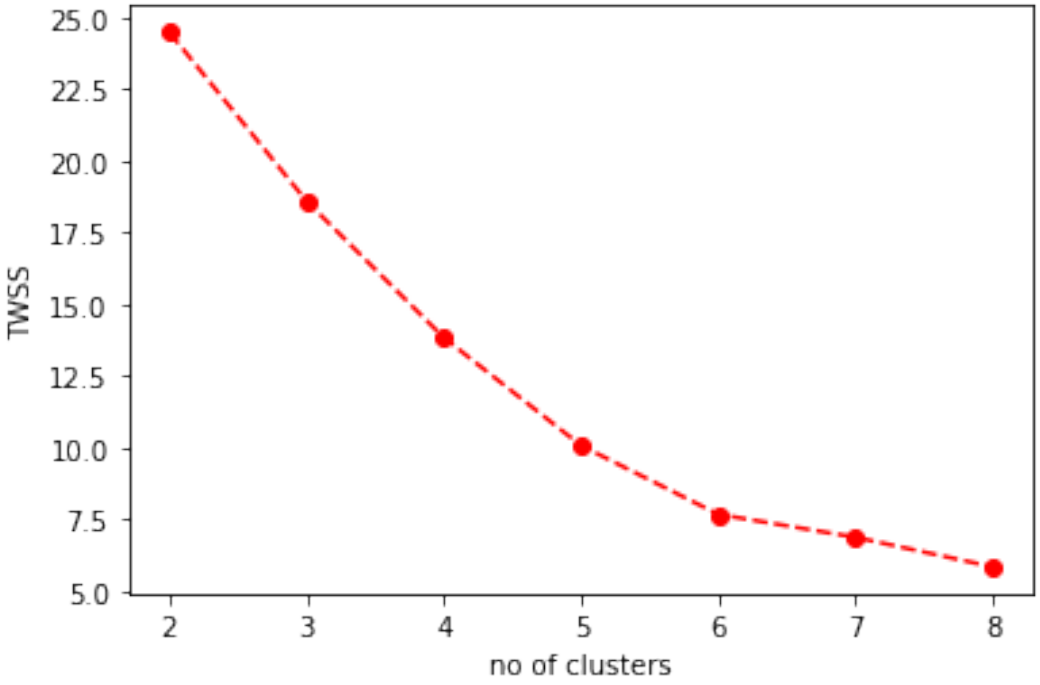
```
#building the model
from sklearn.cluster import KMeans

TWSS=[]
k=list(range(2,9))
for i in k:
  kmeans=KMeans(n_clusters=i,init='k-means++')
  kmeans.fit(data)
  TWSS.append(kmeans.inertia_)

TWSS
```

```
[24.485180966263883,
 18.596754361562283,
 13.831213096343365,
```

```
 10.065537883369897,
 7.6657094194924404,
 6.864996744019052,
 5.833851605278124]
```

```
#finding the best k value
plt.plot(k,TWSS,'ro--')
plt.xlabel('no of clusters')
plt.ylabel('TWSS')
```

```
Text(0, 0.5, 'TWSS')
```




```
model=KMeans(n_clusters=3)
model.fit(data)
```

```
KMeans(n_clusters=3)
```

```
model.labels_
```

```
array([1, 1, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0, 1, 1, 0, 1, 2, 0, 1,
1,
       0, 1, 0, 1, 0, 1, 0, 0, 2, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 2,
0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 0, 2, 0, 2, 0, 2, 2, 1, 0, 0, 2,
1,
       0, 0, 1, 0, 2, 0, 0, 0, 2, 1, 0, 1, 0, 0, 2, 1, 2, 0, 0, 2, 0,
0,
       0, 0, 0, 1, 2, 0, 0, 1, 0, 0, 2, 1, 0, 0, 2, 1, 2, 0, 0, 2, 2,
2,
       2, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 2, 1, 2,
```

```
1,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 2, 0, 0, 1, 1, 1, 0,
0,
       0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 2, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0,
0,
       2, 1, 2, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
1,
       1, 1], dtype=int32)
```

```
#converting the labels into series
out=pd.Series(model.labels_)
```

```
df.head()
```

```
   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
```

##Task 10: Add the cluster data with the primary dataset

```
#creating a new column in the original dataset
df['clust']=out
```

```
df.head()
```

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

##Task 11: Split the data into dependent and independent variables

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

```
x
```

```
     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 x 4 columns]
```

```
#dependent variable
y=df.iloc[:,-1]
```

```
y
```

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

##Task 12: Split the data into training and testing

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,shuf
fle=True)
```

```
x_train.shape
```

```
(160, 4)
```

```
y_train.shape
```

```
(160,)
```

```
x_test.shape
```

```
(40, 4)
```

```
y_test.shape
```

```
(40,)
```

##Task 13: Build the Model

```
from sklearn import svm
from sklearn.metrics import accuracy_score,confusion_matrix
```

```
model=svm.SVC(kernel='linear')
```

##Task 14: Train the Model

```
model.fit(x_train,y_train)
```

```
SVC(kernel='linear')
```

##Task 15: Test the Model

```
pred=model.predict(x_test)
```

```
pred
```

```
array([2, 1, 1, 2, 0, 0, 1, 0, 1, 1, 1, 0, 0, 2, 2, 1, 0, 0, 1, 0, 0,
2,
       2, 2, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
dtype=int32)
```

##Task 16: Measure the performance using Evaluation Metrics

```
#accuracy score
accuracy_score(y_test,pred)
```

```
1.0
```

```
#confusion matrix
confusion_matrix(y_test,pred)
```

```
array([[22,  0,  0],
       [ 0, 11,  0],
       [ 0,  0,  7]])
```