#1.download the dataset

```
import pandas as pd
df = pd.read_csv("/content/Mall_Customers.csv")
```

#2.load the dataset

```
df
```

```
     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
..          ...     ...  ...                 ...                    .
..
195         196  Female   35                 120
79
196         197  Female   45                 126
28
197         198    Male   32                 126
74
198         199    Male   32                 137
18
199         200    Male   30                 137
83

[200 rows x 5 columns]
```

#3.Visualization · Univariate,Bi- Variate,Multi-Variate Analysis

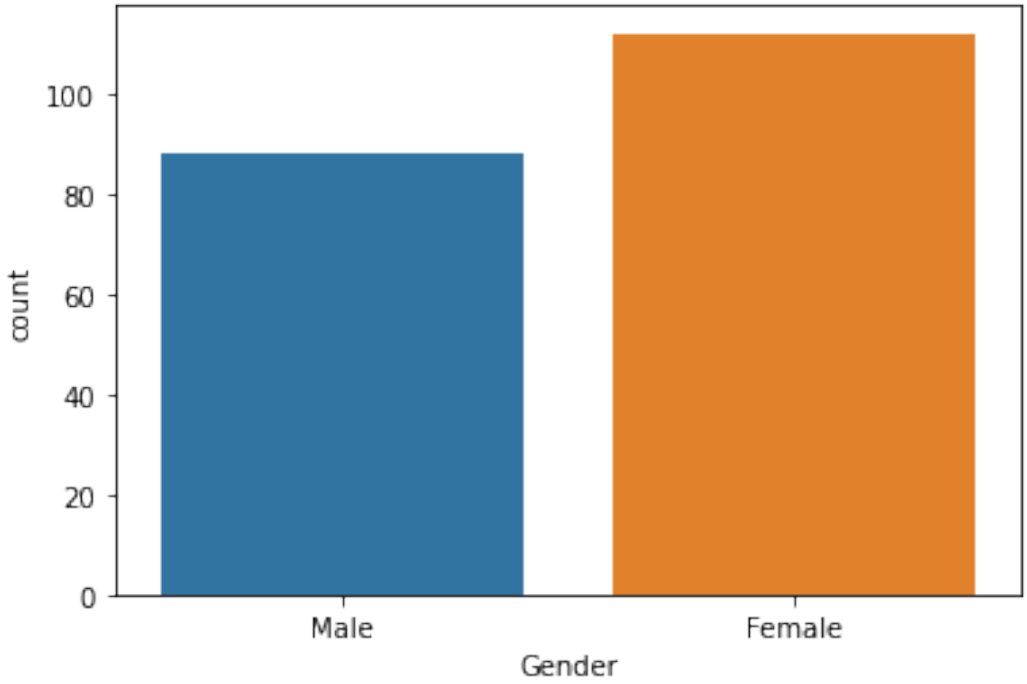
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

sns.countplot(df.Gender)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
```

```
error or misinterpretation.
  FutureWarning
```

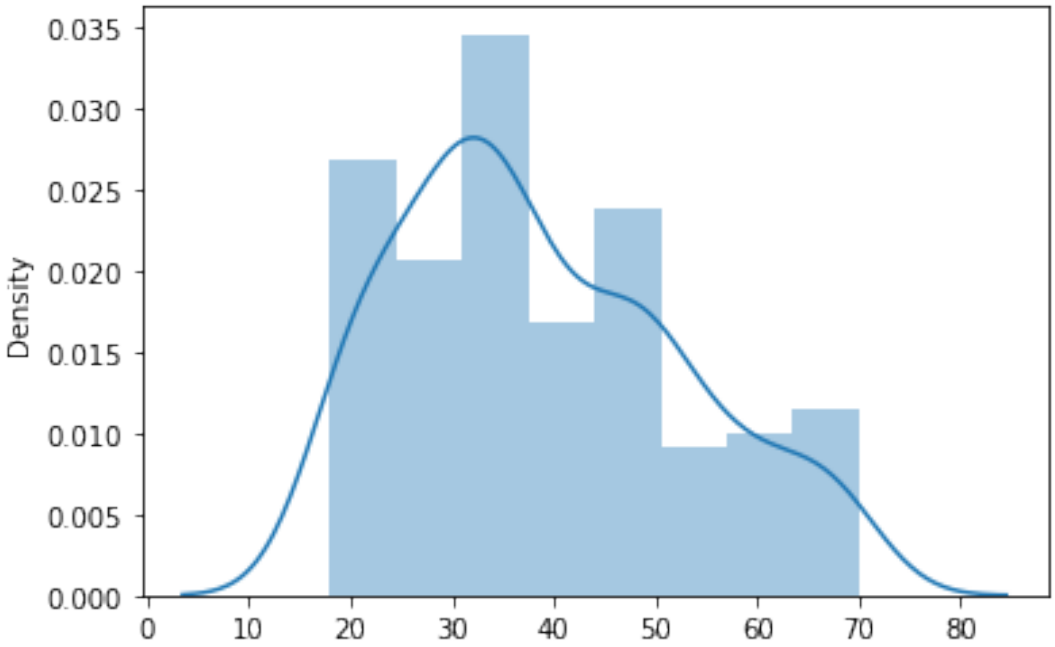
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d5cfa6090>
```




```
sns.distplot([df.Age])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

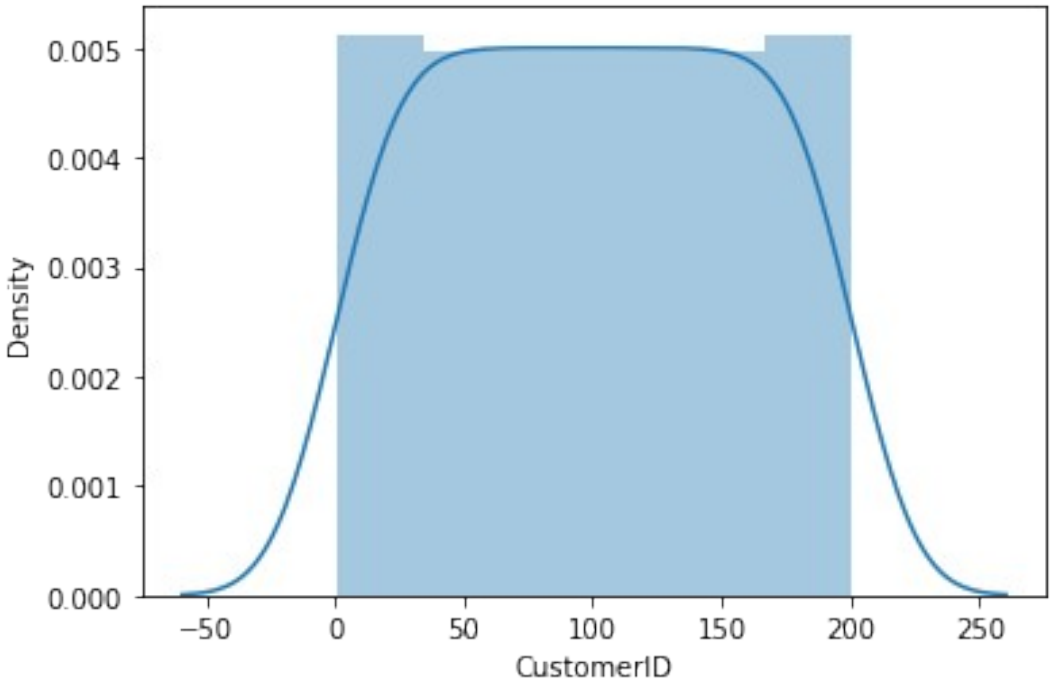
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d5cf2bd90>
```




```
sns.distplot(df.CustomerID)
```

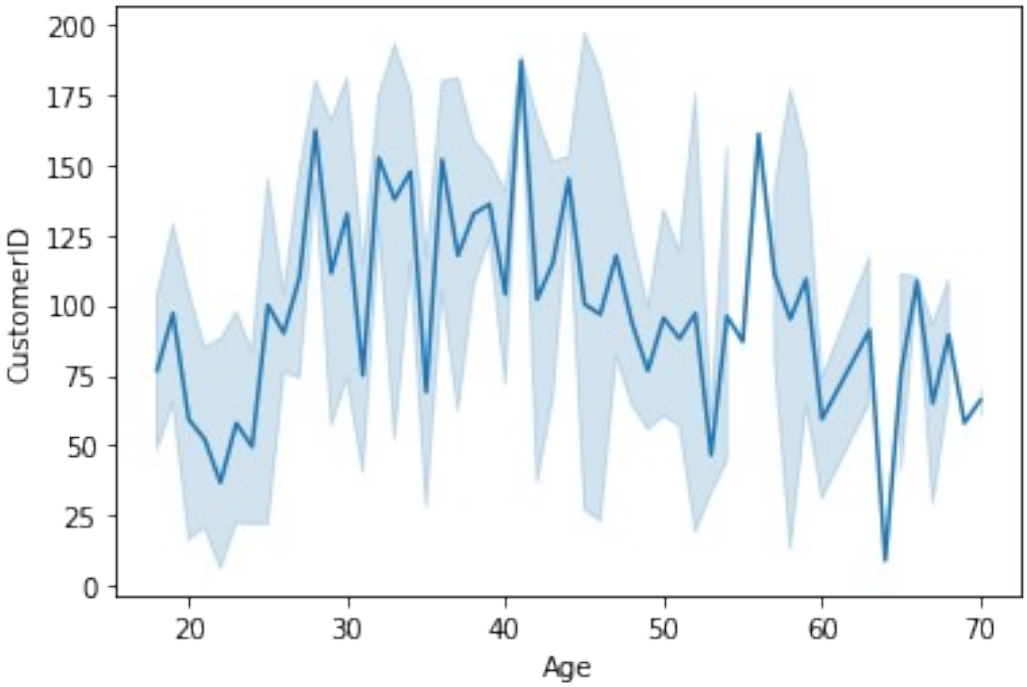
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d5ce549d0>
```



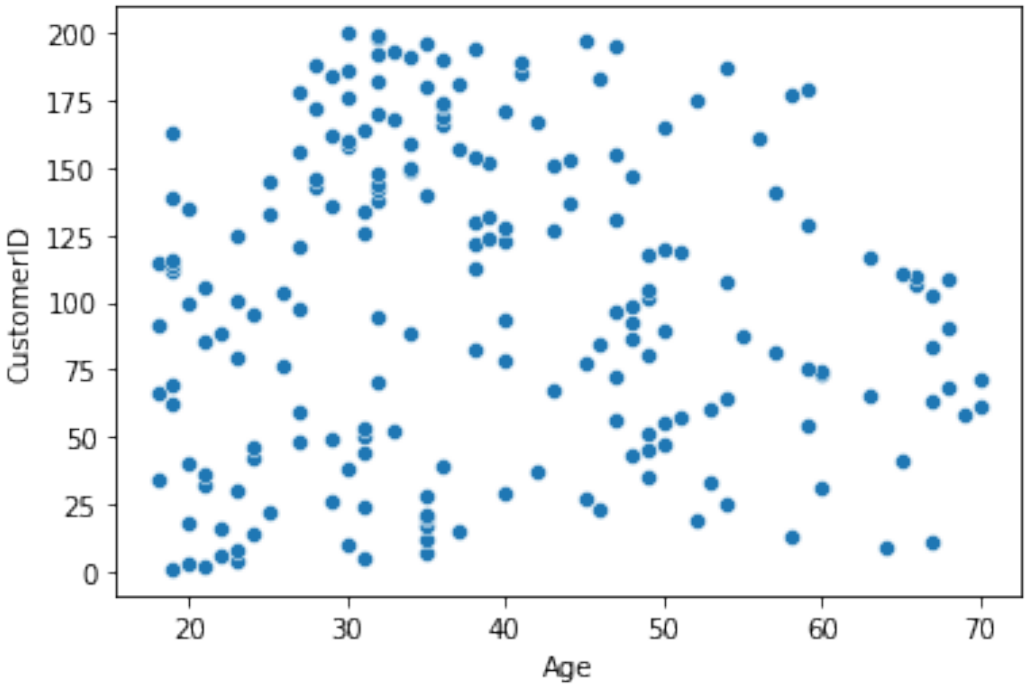

```
sns.lineplot(x=df.Age,y=df.CustomerID)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d5cdde650>
```



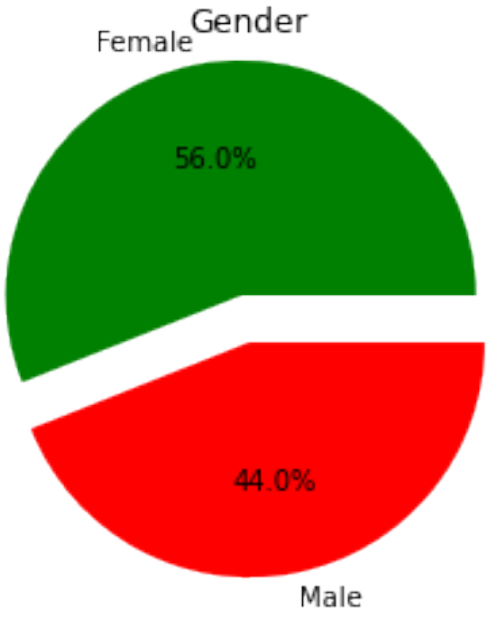

```
sns.scatterplot(x=df.Age,y=df.CustomerID)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d5cd724d0>
```



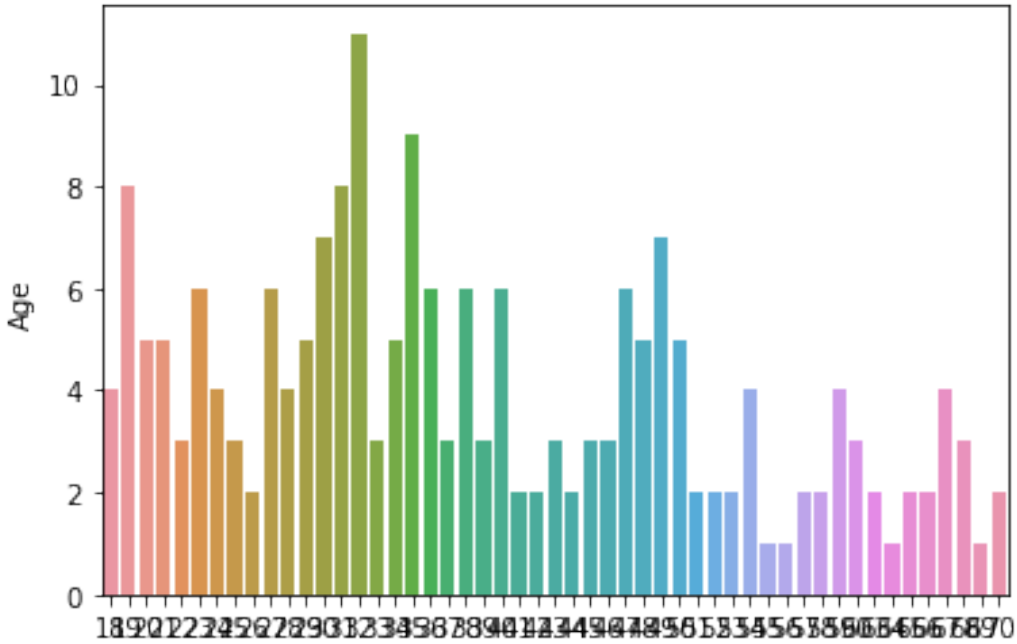

```
plt.pie(df.Gender.value_counts(),
[0.2,0],labels=['Female','Male'],autopct="%1.1f%
%",colors=['green','red'])
plt.title('Gender')
```

```
Text(0.5, 1.0, 'Gender')
```



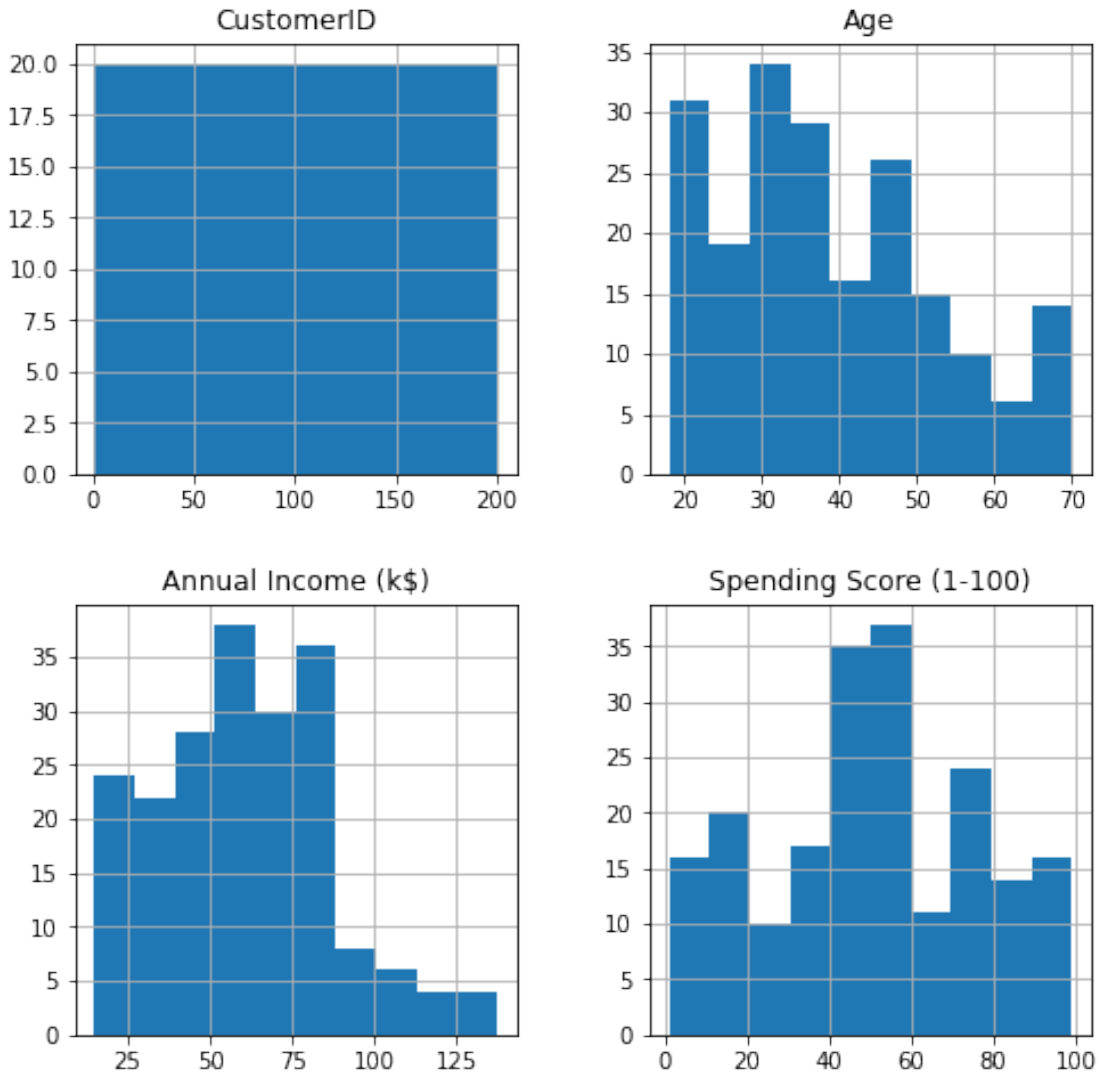

```
sns.barplot(x=df.Age.value_counts().index,y=df.Age.value_counts())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d5cbcb690>
```



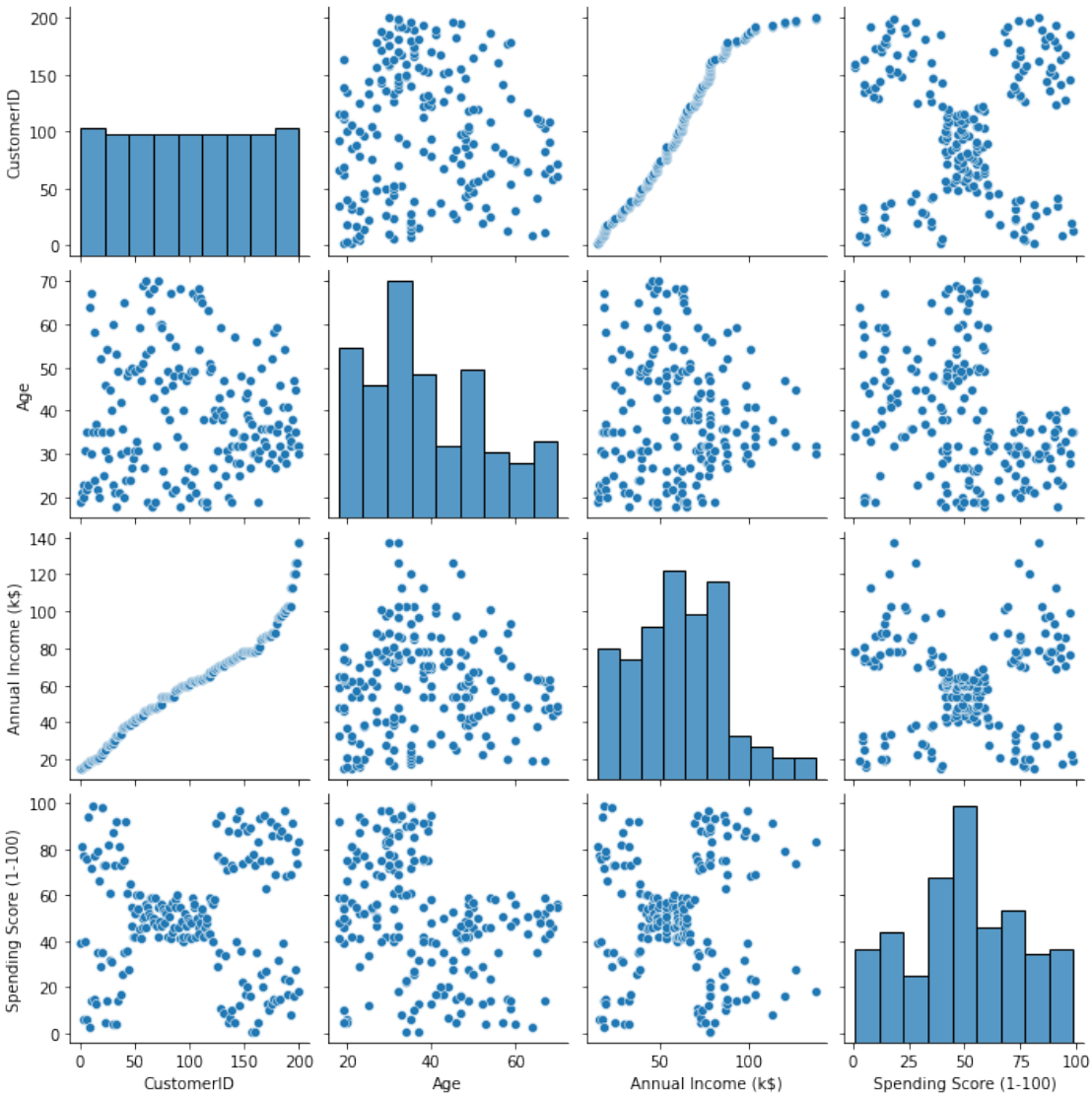

```
df.hist(figsize=(8,8))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at
0x7f3d5c83dc50>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f3d5c7fb1d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at
0x7f3d5c7af7d0>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f3d5c769dd0>]],
      dtype=object)
```




```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f3d5c59ffd0>
```




#4.Perform descriptive statistics on the dataset.

```
df.describe()
```

```
       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.Handle the Missing values.

```
df.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
```

```
df.shape
```

```
(200, 5)
```

```
df.isnull().any()
```

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

```
df.isnull().sum()
```

```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

FillNa

```
d1=df.fillna(100)
d1
```

```
     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
..          ...     ...  ...                 ...                       .
..
195         196  Female   35                 120
79
196         197  Female   45                 126
28
197         198    Male   32                 126
74
198         199    Male   32                 137
18
199         200    Male   30                 137
83

[200 rows x 5 columns]
```

Forward fill

```
d2=df.fillna(method='ffill')
d2
```

```
     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
..          ...     ...  ...                 ...                       .
..
195         196  Female   35                 120
79
196         197  Female   45                 126
28
197         198    Male   32                 126
74
198         199    Male   32                 137
18
199         200    Male   30                 137
83

[200 rows x 5 columns]
```

backwardfill

```
d3=df.fillna(method='bfill')
d3
```

```
     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
..          ...     ...  ...                 ...                    .
..
195         196  Female   35                 120
79
196         197  Female   45                 126
28
197         198    Male   32                 126
74
198         199    Male   32                 137
18
199         200    Male   30                 137
83

[200 rows x 5 columns]
```

fill with mean

```
df.fillna(df.mean())
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this
will raise TypeError.  Select only valid columns before calling the
reduction.
  """Entry point for launching an IPython kernel.
```

```
     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
..          ...     ...  ...                 ...                     .
..
195         196  Female   35                 120                      
79
196         197  Female   45                 126                      
28
197         198    Male   32                 126                      
74
198         199    Male   32                 137                      
18
199         200    Male   30                 137                      
83

[200 rows x 5 columns]
```

Dropna

```
df.dropna
```

```
<bound method DataFrame.dropna of      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
..          ...     ...  ...                 ...                     .
..
195         196  Female   35                 120                      
79
196         197  Female   45                 126                      
28
197         198    Male   32                 126                      
74
198         199    Male   32                 137                      
18
199         200    Male   30                 137                      
83

[200 rows x 5 columns]>
```

#6.Find the Outliers and replace the outliers

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

```
df.head()
```

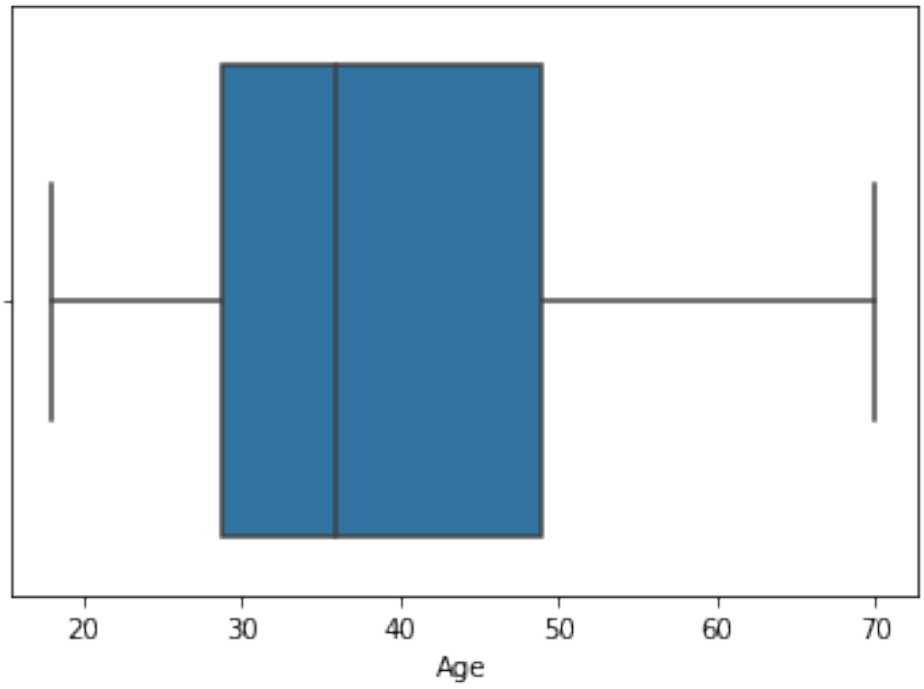
```
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
```

outliers

```
sns.boxplot(df.Age)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d5cbc2ad0>
```

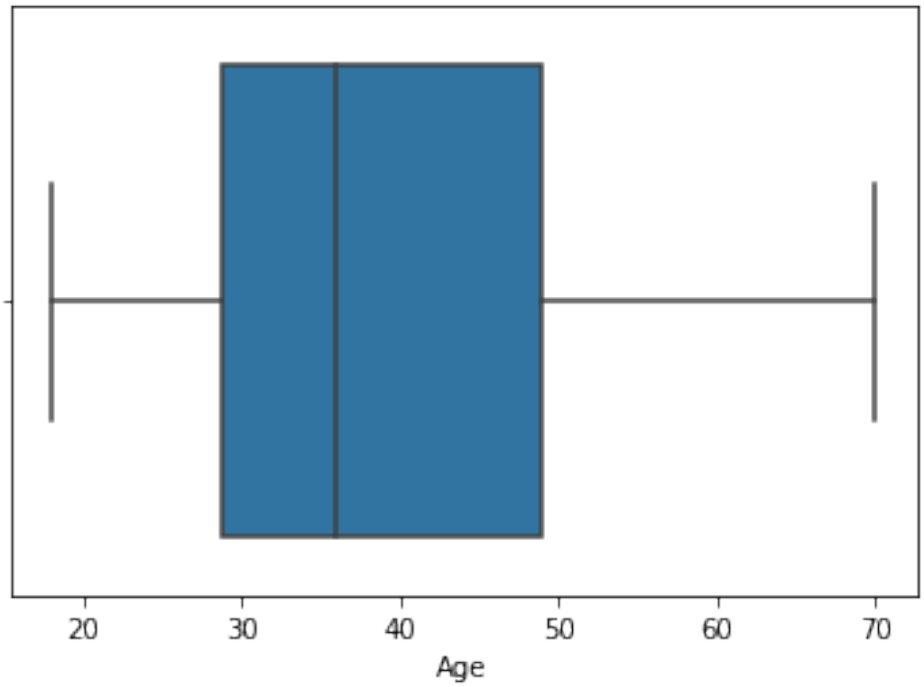



Outliers removal using IQR

```
q1=df.Age.quantile(0.25)  #(Q1)
q3=df.Age.quantile(0.75)  #(Q3)
IQR=q3-q1
upper_limit= q3 + 1.5*IQR
lower_limit= q1 - 1.5*IQR
df=df[df.Age<upper_limit]
sns.boxplot(df.Age)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d57e31710>
```

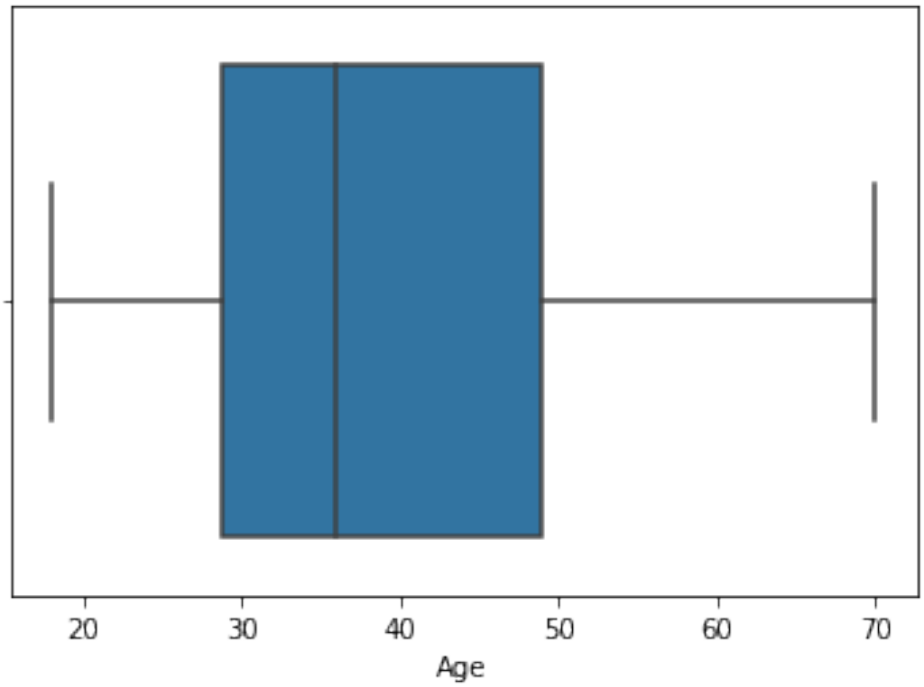



Replacement of outliers-median

```
sns.boxplot(df.Age)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d57da9750>
```




```
q1=df.Age.quantile(0.25)  #(Q1)
q3=df.Age.quantile(0.75)  #(Q3)
IQR=q3-q1
upper_limit= q3 + 1.5*IQR

lower_limit= q1 - 1.5*IQR
upper_limit
```

```
79.375
```

```
df.median()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this
will raise TypeError.  Select only valid columns before calling the
reduction.
  """Entry point for launching an IPython kernel.
```

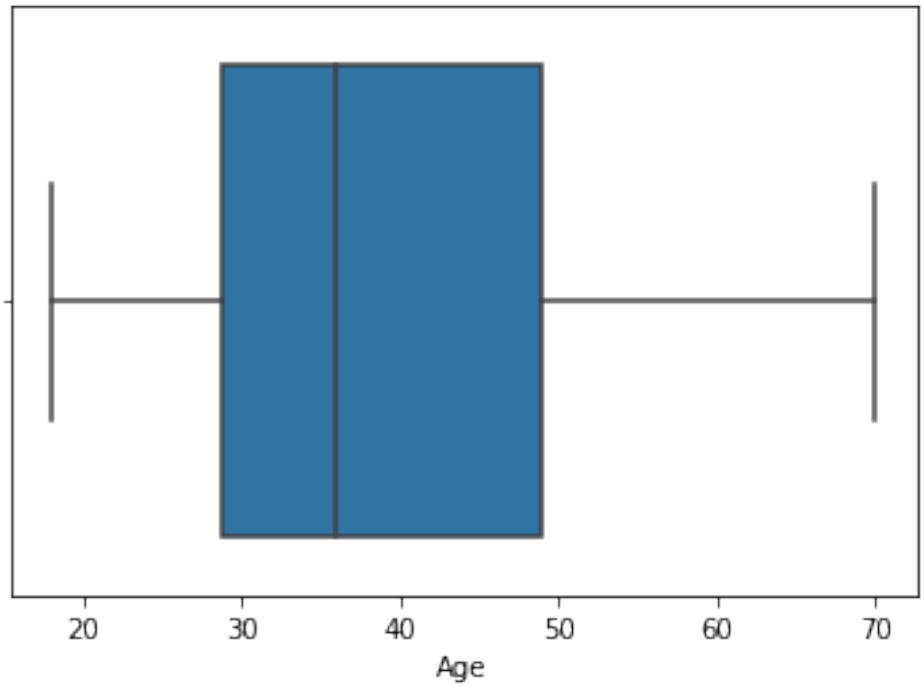
```
CustomerID                100.5
Age                        36.0
Annual Income (k$)         61.5
Spending Score (1-100)     50.0
dtype: float64
```

```
df['Age']= np.where(df['Age']>upper_limit,30,df['Age'])
sns.boxplot(df.Age)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
```

```
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d57d1d1d0>
```




#7.Check the Categorical columns and perform encoding

Categorical columns

```
df.Age.value_counts
```

```
<bound method IndexOpsMixin.value_counts of 0      19
1      21
2      20
3      23
4      31
       ..
195    35
196    45
197    32
198    32
199    30
Name: Age, Length: 200, dtype: int64>
```

```
df.CustomerID.value_counts
```

```
<bound method IndexOpsMixin.value_counts of 0        1
1        2
```

```
2        3
3        4
4        5
      ...
195    196
196    197
197    198
198    199
199    200
Name: CustomerID, Length: 200, dtype: int64>
```

```
df.CustomerID.value_counts
```

```
<bound method IndexOpsMixin.value_counts of 0        1
1        2
2        3
3        4
4        5
      ...
195    196
196    197
197    198
198    199
199    200
Name: CustomerID, Length: 200, dtype: int64>
```

```
df['Annual Income (k$)'].value_counts
```

```
<bound method IndexOpsMixin.value_counts of 0       15
1       15
2       16
3       16
4       17
      ...
195    120
196    126
197    126
198    137
199    137
Name: Annual Income (k$), Length: 200, dtype: int64>
```

```
df.Age.unique()
```

```
array([19, 21, 20, 23, 31, 22, 35, 64, 30, 67, 58, 24, 37, 52, 25, 46,
54,
       29, 45, 40, 60, 53, 18, 49, 42, 36, 65, 48, 50, 27, 33, 59, 47,
51,
       69, 70, 63, 43, 68, 32, 26, 57, 38, 55, 34, 66, 39, 44, 28, 56,
41])
```

```
df['Annual Income (k$)'].unique()
```

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

```
df['Spending Score (1-100)'].unique()
```

```
array([39, 81,  6, 77, 40, 76, 94,  3, 72, 14, 99, 15, 13, 79, 35, 66,
29,
       98, 73,  5, 82, 32, 61, 31, 87,  4, 92, 17, 26, 75, 36, 28, 65,
55,
       47, 42, 52, 60, 54, 45, 41, 50, 46, 51, 56, 59, 48, 49, 53, 44,
57,
       58, 43, 91, 95, 11,  9, 34, 71, 88,  7, 10, 93, 12, 97, 74, 22,
90,
       20, 16, 89,  1, 78, 83, 27, 63, 86, 69, 24, 68, 85, 23,  8,
18])
```

perform encoding.

1.Label encoding

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df.IsActiveMember=le.fit_transform(df['Annual Income (k$)'])
df.Tenure=le.fit_transform(df.Age)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3:
UserWarning: Pandas doesn't allow columns to be created via a new
attribute name - see
https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-
access
  This is separate from the ipykernel package so we can avoid doing
imports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4:
UserWarning: Pandas doesn't allow columns to be created via a new
attribute name - see
https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-
access
  after removing the cwd from sys.path.
```

```
df.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
```

2.One hot encoding

```
df_main=pd.get_dummies(df,columns=['Gender'])
df_main.head()
```

```
   CustomerID  Age  Annual Income (k$)  Spending Score (1-100)
Gender_Female  \
0           1   19                  15                      39
0
1           2   21                  15                      81
0
2           3   20                  16                       6
1
3           4   23                  16                      77
1
4           5   31                  17                      40
1

   Gender_Male
0            1
1            1
2            0
3            0
4            0
```

```
df_main.corr()
```

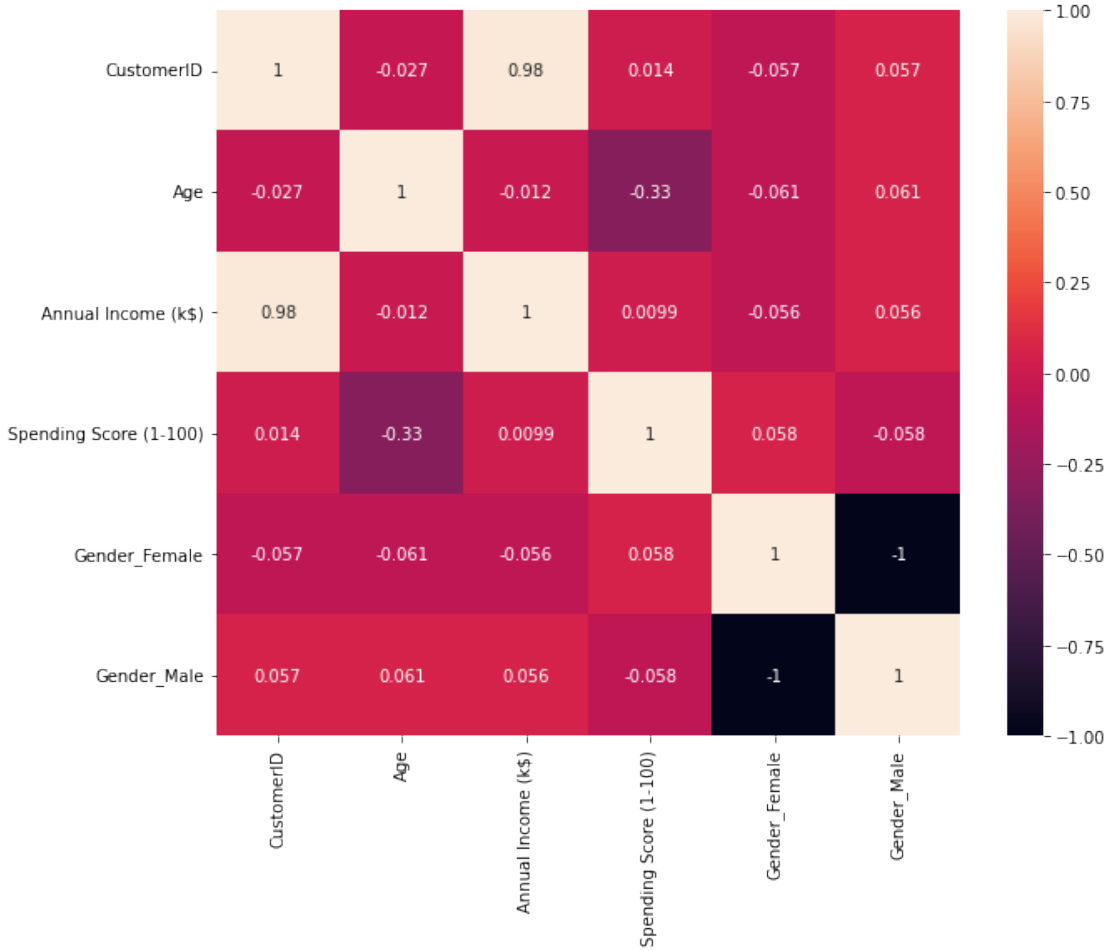
```
                        CustomerID       Age  Annual Income (k$)  \
CustomerID                1.000000 -0.026763            0.977548
Age                      -0.026763  1.000000           -0.012398
Annual Income (k$)        0.977548 -0.012398            1.000000
Spending Score (1-100)    0.013835 -0.327227            0.009903
Gender_Female            -0.057400 -0.060867           -0.056410
Gender_Male               0.057400  0.060867            0.056410

                        Spending Score (1-100)  Gender_Female
Gender_Male
CustomerID                            0.013835      -0.057400
0.057400
Age                                  -0.327227      -0.060867
0.060867
Annual Income (k$)                    0.009903      -0.056410
0.056410
Spending Score (1-100)                1.000000       0.058109    -
0.058109
Gender_Female                         0.058109       1.000000    -
1.000000
```

```
Gender_Male                             -0.058109      -1.000000
1.000000

plt.figure(figsize=(10,8))
sns.heatmap(df_main.corr(),annot=True)

<matplotlib.axes._subplots.AxesSubplot at 0x7f3d5750c150>
```




```
df_main.corr().Age.sort_values(ascending=False)

Age                     1.000000
Gender_Male             0.060867
Annual Income (k$)     -0.012398
CustomerID             -0.026763
Gender_Female          -0.060867
Spending Score (1-100) -0.327227
Name: Age, dtype: float64

df_main.head()

   CustomerID  Age  Annual Income (k$)  Spending Score (1-100)
Gender_Female  \
0           1   19                  15                      39
```

```
0
1           2   21                  15                      81
0
2           3   20                  16                       6
1
3           4   23                  16                      77
1
4           5   31                  17                      40
1

   Gender_Male
0            1
1            1
2            0
3            0
4            0
```

8.Scaling the data

```
from sklearn.preprocessing import scale
w=df.drop(df['Annual Income (k$)'],axis=0)
w
```

```
     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
..          ...     ...  ...                 ...                    .
..
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

[136 rows x 5 columns]
```

9.Perform any of the clustering algorithms

```
import pandas as pd
import numpy as np

df = pd.read_csv('/content/Mall_Customers.csv')
df.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 |

```
new_df =df.iloc[:,:-1]
new_df.head()
```

| | CustomerID | Gender | Age | Annual Income (k$) |
|---|---|---|---|---|
| 0 | 1 | Male | 19 | 15 |
| 1 | 2 | Male | 21 | 15 |
| 2 | 3 | Female | 20 | 16 |
| 3 | 4 | Female | 23 | 16 |
| 4 | 5 | Female | 31 | 17 |

```
new_df.shape
```

```
(200, 4)
```

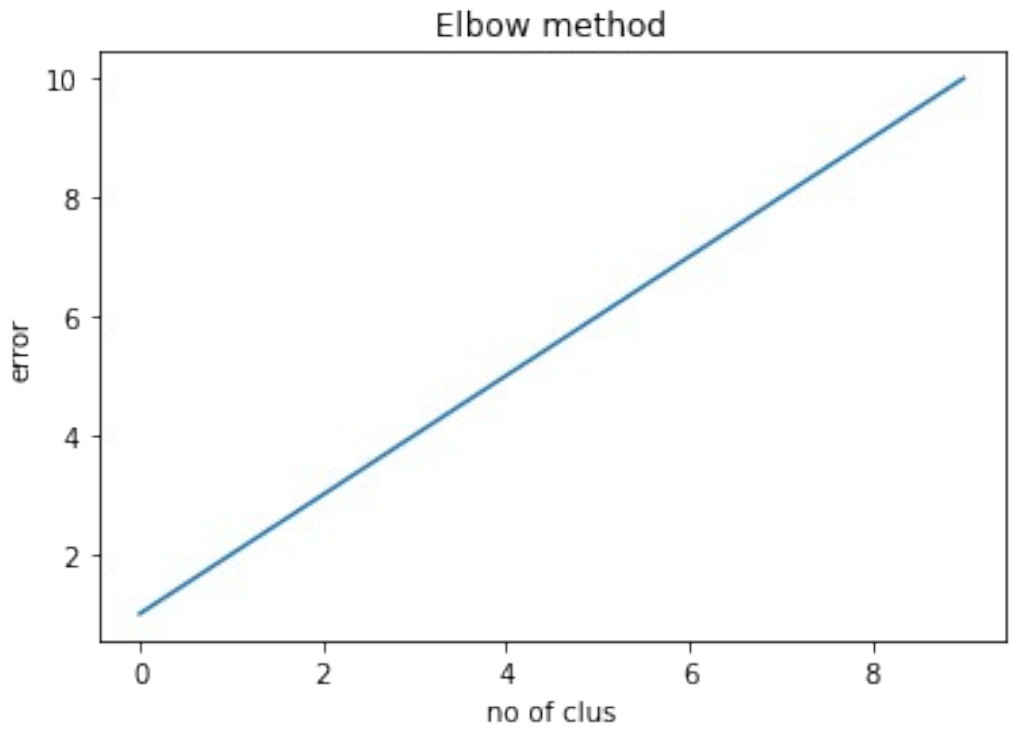
```
from sklearn import cluster

error =[]
for i in range(1,11):
    kmeans=cluster.KMeans(n_clusters=i,init='k-means+
+',random_state=0)

error
```

```
[]
```

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




10.Add the cluster data with the primary dataset

```
import pandas as pd
import numpy as np

df = pd.read_csv('/content/Mall_Customers.csv')
df.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 |

```
new_df =df.iloc[:,:-1]
new_df.head()
```

| | CustomerID | Gender | Age | Annual Income (k$) |
|---|---|---|---|---|
| 0 | 1 | Male | 19 | 15 |
| 1 | 2 | Male | 21 | 15 |
| 2 | 3 | Female | 20 | 16 |
| 3 | 4 | Female | 23 | 16 |
| 4 | 5 | Female | 31 | 17 |

```
new_df =df.iloc[:,:-1]
new_df.head()
```

```
   CustomerID  Gender  Age  Annual Income (k$)
0           1    Male   19                  15
1           2    Male   21                  15
2           3  Female   20                  16
3           4  Female   23                  16
4           5  Female   31                  17
```

```
from sklearn import cluster
```

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


km_model=cluster.KMeans(n_clusters=3,init='k-means++',random_state=0)
```

```
new_df['kclus'] = pd.Series
```

11.Split the data into dependent and independent variables.

dependent variables.

```
y=df_main['Age']
y
```

```
0      19
1      21
2      20
3      23
4      31
       ..
195    35
196    45
197    32
198    32
199    30
Name: Age, Length: 200, dtype: int64
```

independent variables

```
X=df_main.drop(columns=['Age'],axis=1)
X.head()
```

```
   CustomerID  Annual Income (k$)  Spending Score (1-100)
Gender_Female  \
0           1                  15                      39
0
1           2                  15                      81
0
2           3                  16                       6
1
```

```
3           4                   16                       77
1
4           5                   17                       40
1

   Gender_Male
0            1
1            1
2            0
3            0
4            0
```

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=0.2,random_state=0)

X_train.shape

(160, 5)

y_train.shape

(160,)

X_test.shape

(40, 5)

y_test.shape

(40,)

X_train

     CustomerID  Annual Income (k$)  Spending Score (1-100)
Gender_Female  \
134         135                  73                       5
0
66           67                  48                      50
1
26           27                  28                      32
1
113         114                  64                      46
0
168         169                  87                      27
1
..          ...                 ...                     ...
...
67           68                  48                      48
1
192         193                 113                       8
```

```
0
117         118                   65                        59
1
47           48                   40                        47
1
172         173                   87                        10
0

     Gender_Male
134            1
66             0
26             0
113            1
168            0
..           ...
67             0
192            1
117            0
47             0
172            1

[160 rows x 5 columns]
```

y_train

```
134    20
66     43
26     45
113    19
168    36
       ..
67     68
192    33
117    49
47     27
172    36
Name: Age, Length: 160, dtype: int64
```

X_test

```
     CustomerID  Annual Income (k$)  Spending Score (1-100)
Gender_Female  \
18           19                   23                        29
0
170         171                   87                        13
0
107         108                   63                        46
0
98           99                   61                        42
0
177         178                   88                        69
```

0
182 183 98 15
0
5 6 17 76
1
146 147 77 36
0
12 13 20 15
1
152 153 78 20
1
61 62 46 55
0
125 126 70 77
1
180 181 97 32
1
154 155 78 16
1
80 81 54 51
0
7 8 18 94
1
33 34 33 92
0
130 131 71 9
0
37 38 34 73
1
74 75 54 47
0
183 184 98 88
1
145 146 77 97
0
45 46 39 65
1
159 160 78 73
1
60 61 46 56
0
123 124 69 91
0
179 180 93 90
0
185 186 99 97
0
122 123 69 58
1
44 45 39 28

1
16 17 21 35
1
55 56 43 41
0
150 151 78 17
0
111 112 63 54
1
22 23 25 5
1
189 190 103 85
1
129 130 71 75
0
4 5 17 40
1
83 84 54 44
1
106 107 63 50
1

| | Gender_Male |
|---|---|
| 18 | 1 |
| 170 | 1 |
| 107 | 1 |
| 98 | 1 |
| 177 | 1 |
| 182 | 1 |
| 5 | 0 |
| 146 | 1 |
| 12 | 0 |
| 152 | 0 |
| 61 | 1 |
| 125 | 0 |
| 180 | 0 |
| 154 | 0 |
| 80 | 1 |
| 7 | 0 |
| 33 | 1 |
| 130 | 1 |
| 37 | 0 |
| 74 | 1 |
| 183 | 0 |
| 145 | 1 |
| 45 | 0 |
| 159 | 0 |
| 60 | 1 |
| 123 | 1 |
| 179 | 1 |

| | |
|---|---|
| 185 | 1 |
| 122 | 0 |
| 44 | 0 |
| 16 | 0 |
| 55 | 1 |
| 150 | 1 |
| 111 | 0 |
| 22 | 0 |
| 189 | 0 |
| 129 | 1 |
| 4 | 0 |
| 83 | 0 |
| 106 | 0 |

y_test

| | |
|---|---|
| 18 | 52 |
| 170 | 40 |
| 107 | 54 |
| 98 | 48 |
| 177 | 27 |
| 182 | 46 |
| 5 | 22 |
| 146 | 48 |
| 12 | 58 |
| 152 | 44 |
| 61 | 19 |
| 125 | 31 |
| 180 | 37 |
| 154 | 47 |
| 80 | 57 |
| 7 | 23 |
| 33 | 18 |
| 130 | 47 |
| 37 | 30 |
| 74 | 59 |
| 183 | 29 |
| 145 | 28 |
| 45 | 24 |
| 159 | 30 |
| 60 | 70 |
| 123 | 39 |
| 179 | 35 |
| 185 | 30 |
| 122 | 40 |
| 44 | 49 |
| 16 | 35 |
| 55 | 47 |
| 150 | 43 |
| 111 | 19 |
| 22 | 46 |

```
189    36
129    38
4      31
83     46
106    66
Name: Age, dtype: int64
```

13.Build the Model

```
from sklearn.linear_model import LinearRegression
model=LinearRegression() # initialzing the model

model.fit(X_train,y_train) # fitting the model on training data

LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

14.train the model

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()

lr.fit(X_train,y_train)

LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org

15.test the model

```
p=model.predict(X_test)
p

array([42.36050511, 43.1765722 , 39.0534294 , 39.98878139,
34.17795933,
       43.45353768, 35.05024728, 39.88461784, 44.44204676,
42.01767724,
       38.44771287, 33.87895033, 40.7207682 , 42.51446339,
38.85995582,
       32.25529851, 32.86894674, 44.33717665, 35.59104097,
39.85147387,
       31.96952129, 30.48430521, 36.94147362, 33.36357095,
38.35443708,
       31.80875954, 31.40103978, 30.66924147, 36.89085787,
42.74265449,
       41.21318046, 40.62786794, 42.70532548, 37.46651178,
45.979031  ,
```

```
       32.66805052, 34.16127762, 40.69631087, 39.6616389 ,
38.39618835])
```

```
y_pred=lr.predict(X_test)
y_pred
```

```
array([42.36050511, 43.1765722 , 39.0534294 , 39.98878139,
34.17795933,
       43.45353768, 35.05024728, 39.88461784, 44.44204676,
42.01767724,
       38.44771287, 33.87895033, 40.7207682 , 42.51446339,
38.85995582,
       32.25529851, 32.86894674, 44.33717665, 35.59104097,
39.85147387,
       31.96952129, 30.48430521, 36.94147362, 33.36357095,
38.35443708,
       31.80875954, 31.40103978, 30.66924147, 36.89085787,
42.74265449,
       41.21318046, 40.62786794, 42.70532548, 37.46651178,
45.979031  ,
       32.66805052, 34.16127762, 40.69631087, 39.6616389 ,
38.39618835])
```

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

```
array([42.36050511, 43.1765722 , 39.0534294 , 39.98878139,
34.17795933,
       43.45353768, 35.05024728, 39.88461784, 44.44204676,
42.01767724,
       38.44771287, 33.87895033, 40.7207682 , 42.51446339,
38.85995582,
       32.25529851, 32.86894674, 44.33717665, 35.59104097,
39.85147387,
       31.96952129, 30.48430521, 36.94147362, 33.36357095,
38.35443708,
       31.80875954, 31.40103978, 30.66924147, 36.89085787,
42.74265449,
       41.21318046, 40.62786794, 42.70532548, 37.46651178,
45.979031  ,
       32.66805052, 34.16127762, 40.69631087, 39.6616389 ,
38.39618835])
```

```
Sal= pd.DataFrame({'Annual Income (k$)': y_test,'Spending Score (1-
100)':pred})
Sal
```

```
     Annual Income (k$)  Spending Score (1-100)
18                   52               42.360505
170                  40               43.176572
107                  54               39.053429
```

| | | |
|---|---|---|
| 98 | 48 | 39.988781 |
| 177 | 27 | 34.177959 |
| 182 | 46 | 43.453538 |
| 5 | 22 | 35.050247 |
| 146 | 48 | 39.884618 |
| 12 | 58 | 44.442047 |
| 152 | 44 | 42.017677 |
| 61 | 19 | 38.447713 |
| 125 | 31 | 33.878950 |
| 180 | 37 | 40.720768 |
| 154 | 47 | 42.514463 |
| 80 | 57 | 38.859956 |
| 7 | 23 | 32.255299 |
| 33 | 18 | 32.868947 |
| 130 | 47 | 44.337177 |
| 37 | 30 | 35.591041 |
| 74 | 59 | 39.851474 |
| 183 | 29 | 31.969521 |
| 145 | 28 | 30.484305 |
| 45 | 24 | 36.941474 |
| 159 | 30 | 33.363571 |
| 60 | 70 | 38.354437 |
| 123 | 39 | 31.808760 |
| 179 | 35 | 31.401040 |
| 185 | 30 | 30.669241 |
| 122 | 40 | 36.890858 |
| 44 | 49 | 42.742654 |
| 16 | 35 | 41.213180 |
| 55 | 47 | 40.627868 |
| 150 | 43 | 42.705325 |
| 111 | 19 | 37.466512 |
| 22 | 46 | 45.979031 |
| 189 | 36 | 32.668051 |
| 129 | 38 | 34.161278 |
| 4 | 31 | 40.696311 |
| 83 | 46 | 39.661639 |
| 106 | 66 | 38.396188 |

16:.Measure the performance using Evaluation Metrics.

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()

model.fit(X_train,y_train)

KNeighborsClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
y_pred=model.predict(X_test)

y_pred1=model.predict(X_train)

from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report

print('Testing accuracy: ',accuracy_score(y_test,y_pred))
print('Training accuracy: ',accuracy_score(y_train,y_pred1))

Testing accuracy:  0.025
Training accuracy:  0.24375

pd.crosstab(y_test,y_pred)

col_0  18  19  20  21  23  25  28  29  30  32  34  35  36  37  40  41
49
Age

18      0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
0
19      2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
22      0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
0
23      0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
0
24      0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
0
27      0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
0
28      0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
0
29      0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
0
30      0   0   0   1   0   0   0   1   1   0   0   0   0   0   0   0
0
31      0   1   0   0   0   0   0   1   0   0   0   0   0   0   0   0
0
35      0   1   0   0   0   0   1   0   0   0   0   0   0   0   0   0
0
36      0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
0
37      0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
0
38      0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
0
39      0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
0
40      1   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
0
```

```
43      0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
44      0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
0
46      0   0   0   0   0   0   0   0   0   0   1   0   0   1   1   0
0
47      0   2   0   0   0   0   0   1   0   0   0   0   0   0   0   0
0
48      0   0   0   0   1   1   0   0   0   0   0   0   0   0   0   0
0
49      0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
0
52      0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
0
54      0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
57      0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
0
58      0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
0
59      0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
0
66      0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
1
70      1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

          18       0.00      0.00      0.00         1
          19       0.00      0.00      0.00         2
          20       0.00      0.00      0.00         0
          21       0.00      0.00      0.00         0
          22       0.00      0.00      0.00         1
          23       0.00      0.00      0.00         1
          24       0.00      0.00      0.00         1
          25       0.00      0.00      0.00         0
          27       0.00      0.00      0.00         1
          28       0.00      0.00      0.00         1
          29       0.00      0.00      0.00         1
          30       1.00      0.33      0.50         3
          31       0.00      0.00      0.00         2
          32       0.00      0.00      0.00         0
          34       0.00      0.00      0.00         0
          35       0.00      0.00      0.00         2
          36       0.00      0.00      0.00         1
          37       0.00      0.00      0.00         1
          38       0.00      0.00      0.00         1
```

```
           39       0.00      0.00      0.00         1
           40       0.00      0.00      0.00         2
           41       0.00      0.00      0.00         0
           43       0.00      0.00      0.00         1
           44       0.00      0.00      0.00         1
           46       0.00      0.00      0.00         3
           47       0.00      0.00      0.00         3
           48       0.00      0.00      0.00         2
           49       0.00      0.00      0.00         1
           52       0.00      0.00      0.00         1
           54       0.00      0.00      0.00         1
           57       0.00      0.00      0.00         1
           58       0.00      0.00      0.00         1
           59       0.00      0.00      0.00         1
           66       0.00      0.00      0.00         1
           70       0.00      0.00      0.00         1

    accuracy                           0.03        40
   macro avg       0.03      0.01      0.01        40
weighted avg       0.07      0.03      0.04        40


/usr/local/lib/python3.7/dist-packages/sklearn/metrics/
_classification.py:1318: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification
.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined
and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification
.py:1318: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification
.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined
and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification
.py:1318: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification
.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined
and being set to 0.0 in labels with no true samples. Use
```

```
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```