```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings("ignore")
```

**2. Load the dataset into the tool.**

```
from google.colab import files
upload=files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving Mall_Customers.csv to Mall_Customers.csv
```

```
mydata = pd.read_csv('Mall_Customers.csv')
```

```
mydata.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
```

```
mydata.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
```

```
mydata.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
```

```
mydata.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
```

```
mydata.dtypes
```

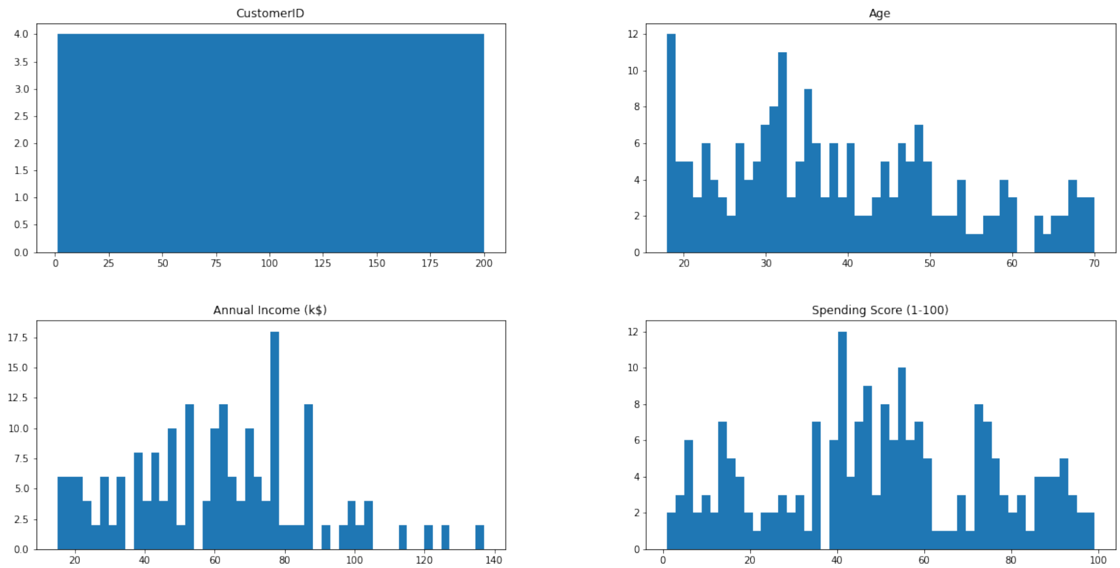
```
CustomerID                 int64
Gender                    object
Age                        int64
Annual Income (k$)         int64
Spending Score (1-100)     int64
dtype: object
```

**3.Perform visualisations Univariate Analysis**

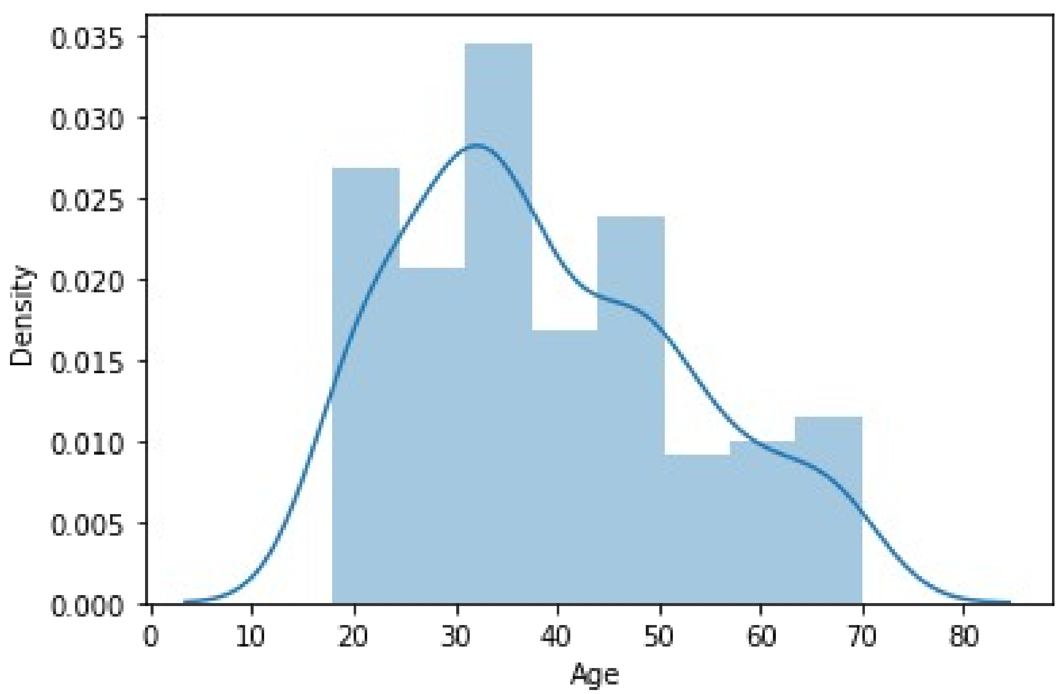
```
mydata.hist(figsize=(20,10), grid = False, bins = 50)
```

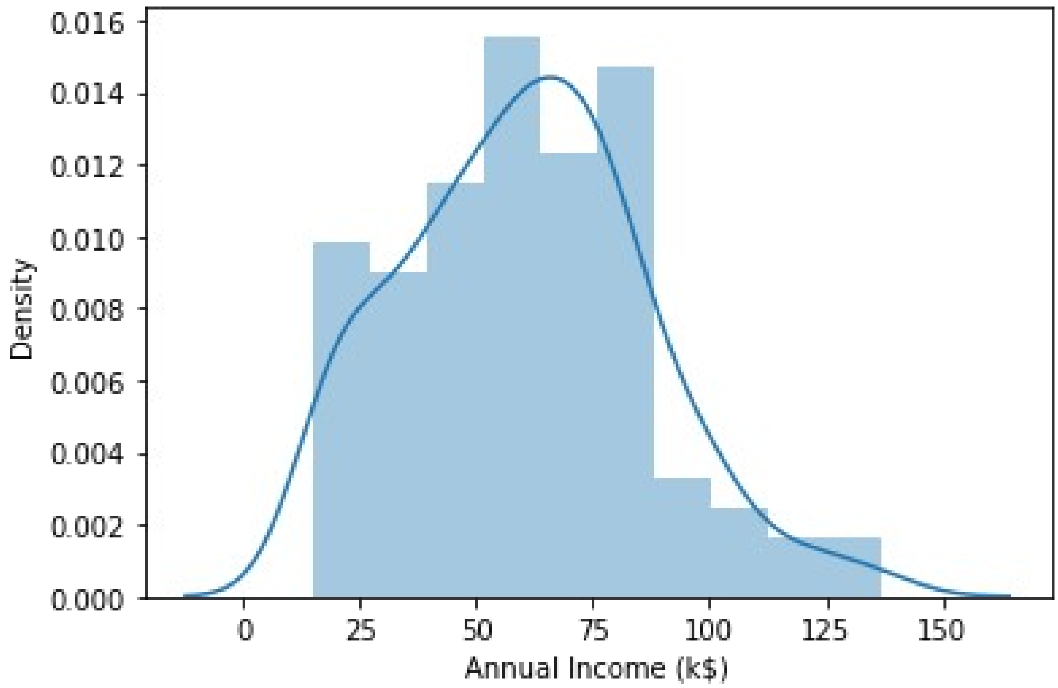
```
array([[<matplotlib.axes._subplots.AxesSubplot object at
0x7f9dff9e9450>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f9dff9e0610>],
       [<matplotlib.axes._subplots.AxesSubplot object at
0x7f9dff999c10>,
        <matplotlib.axes._subplots.AxesSubplot object at
```

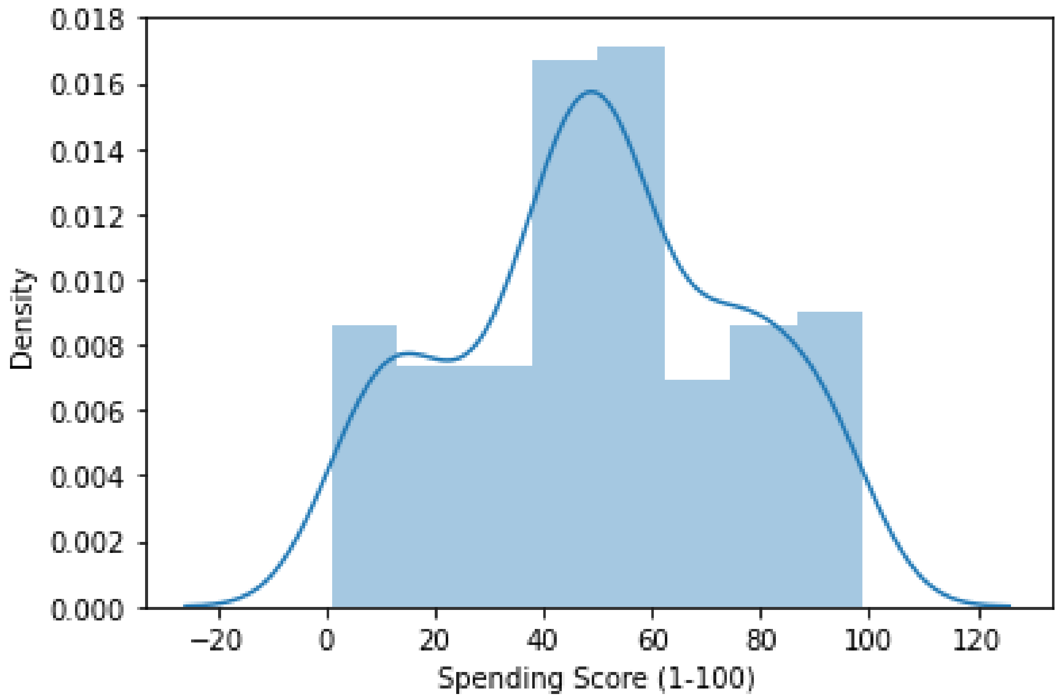
```
0x7f9dff95d250>]],
      dtype=object)
```



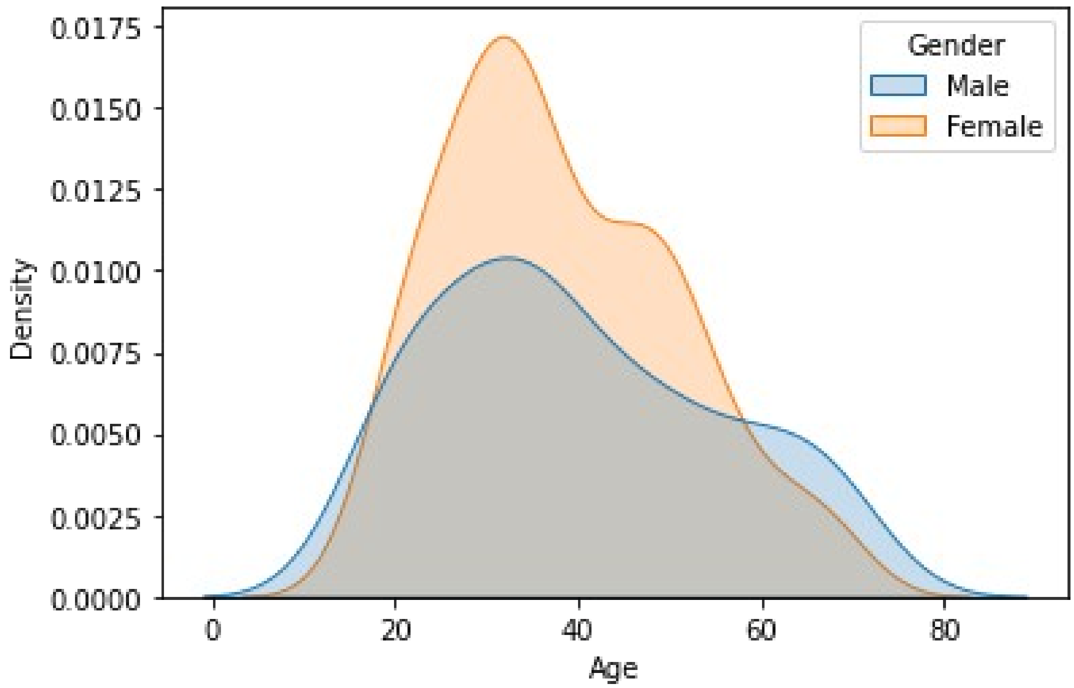

```python
columns = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']
for i in columns:
  plt.figure()
  sns.distplot(mydata[i])
```

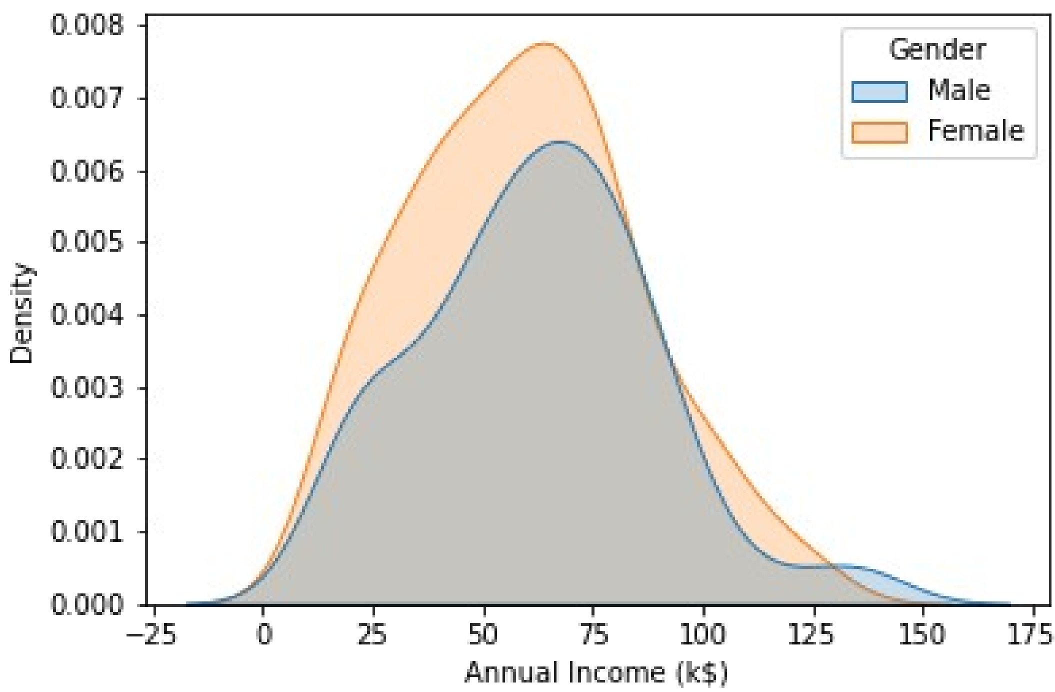


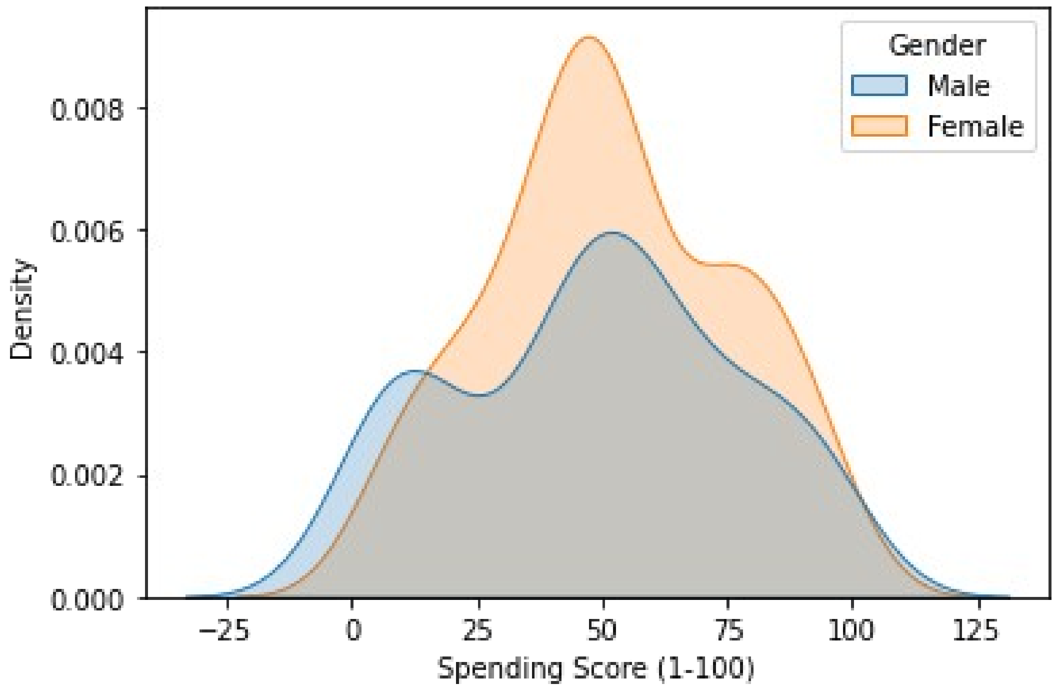


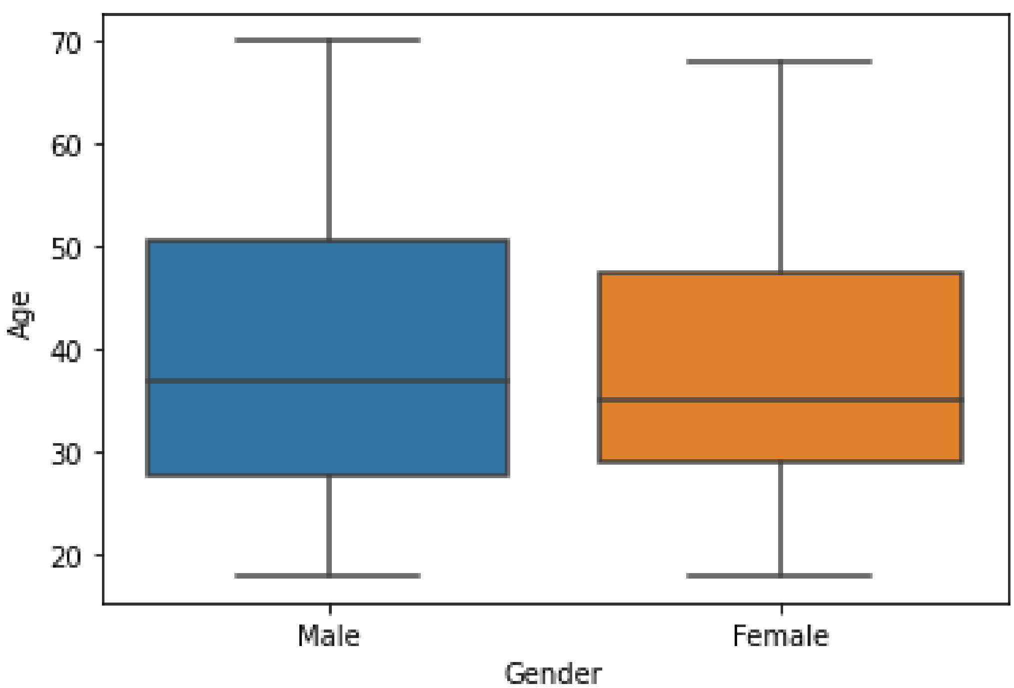

```
columns = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']
for i in columns:
  plt.figure()
  sns.kdeplot(mydata[i], shade=True, hue=mydata['Gender'])
```

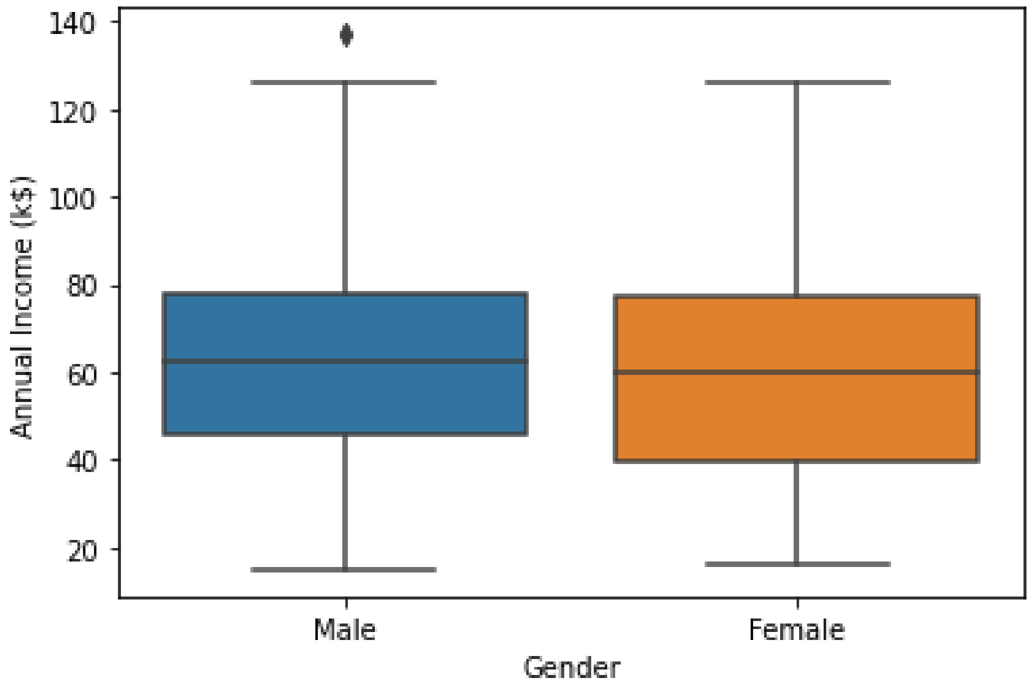

Gender
Male
Female
Density
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
20
40
60
80
Age

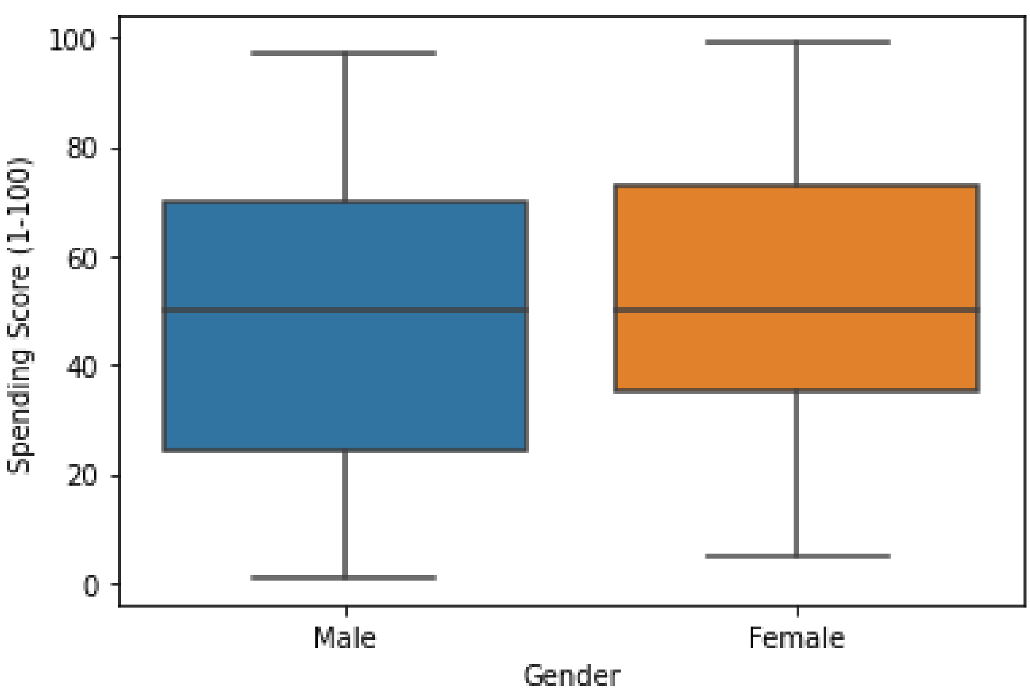

Gender
Male
Female
Density
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
−25
0
25
50
75
100
125
150
175
Annual Income (k$)




```
columns = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']
for i in columns:
  plt.figure()
  sns.boxplot(data=mydata,x='Gender',y=mydata[i])
```





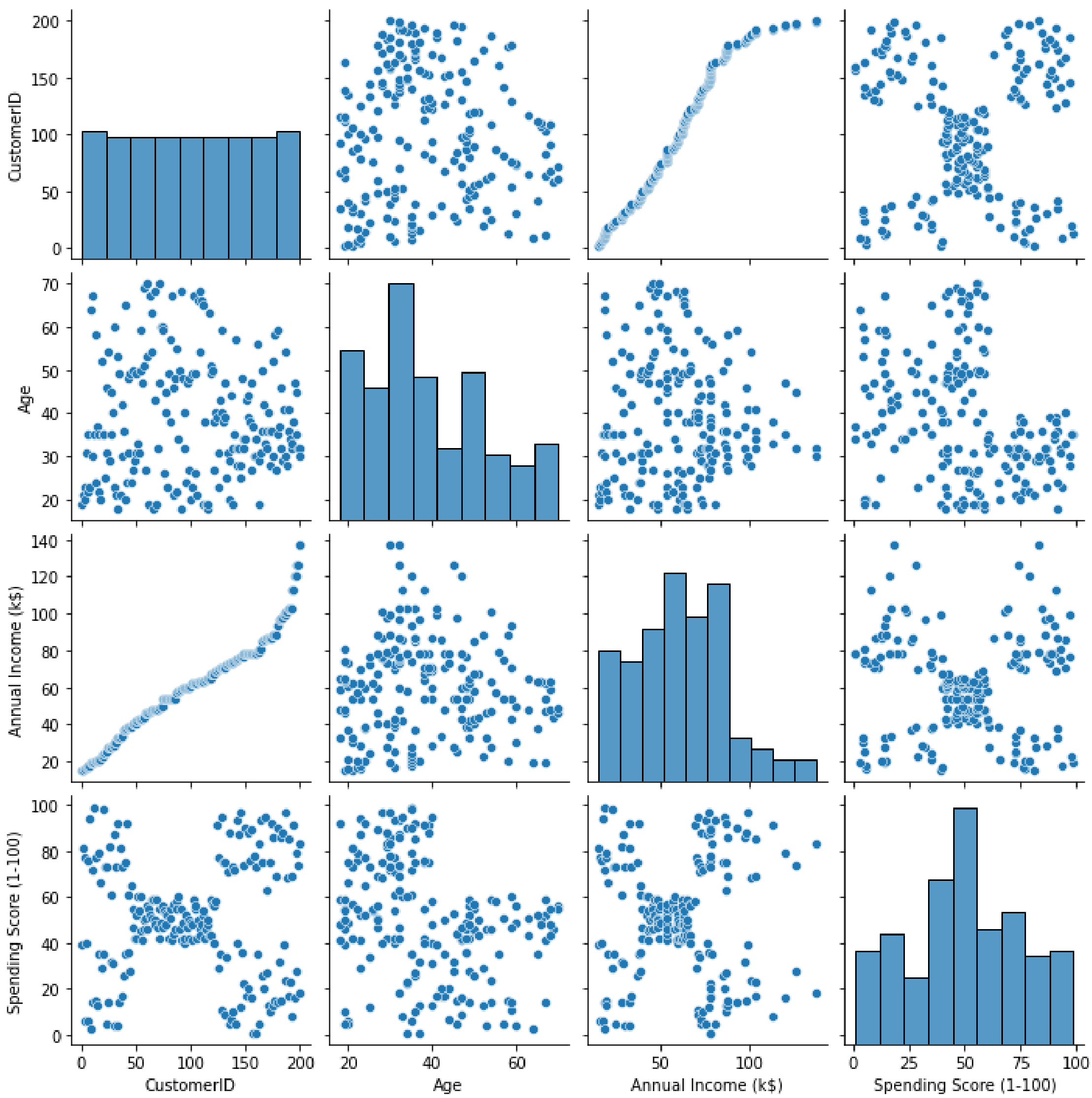

```
mydata['Gender'].value_counts(normalize=True)
```

```
Female    0.56
Male      0.44
Name: Gender, dtype: float64
```

**Bi- Variate Analysis**

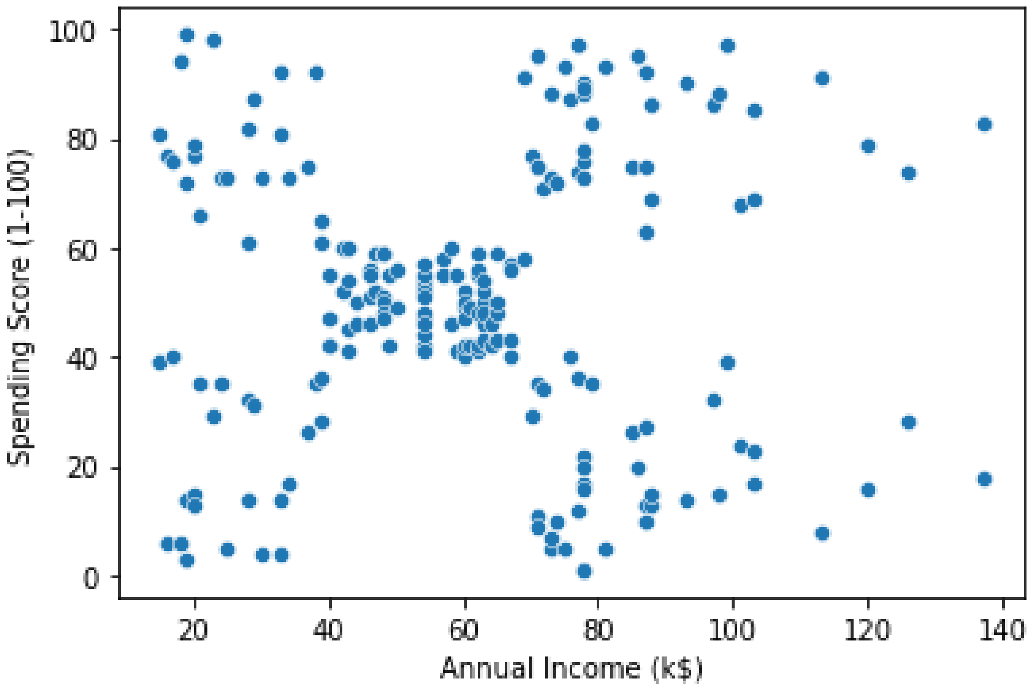
```
numerical_features = mydata.select_dtypes(include=[np.number]).columns
categorical_features =
mydata.select_dtypes(include=[np.object]).columns
numerical_features
categorical_features
sns.pairplot(mydata[numerical_features])
```

```
<seaborn.axisgrid.PairGrid at 0x7f9dfedfe090>
```



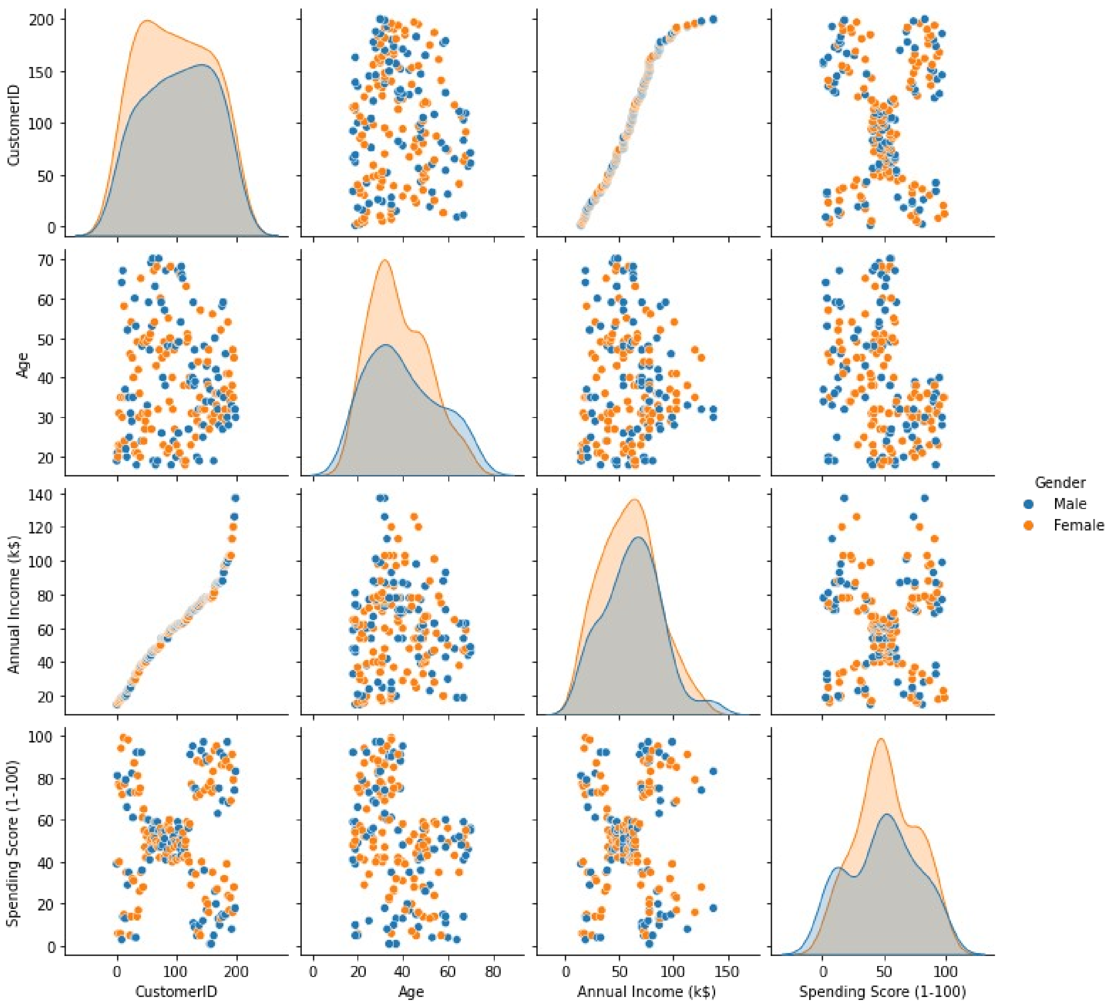

```
sns.scatterplot(data=mydata, x='Annual Income (k$)', y='Spending Score
(1-100)')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9dff837910>
```




```
sns.pairplot(mydata,hue='Gender')
```

```
<seaborn.axisgrid.PairGrid at 0x7f9dff8d1890>
```




```
mydata.groupby(['Gender'])['Age', 'Annual Income (k$)','Spending Score
(1-100)'].mean()
```

```
              Age  Annual Income (k$)  Spending Score (1-100)
Gender
Female  38.098214           59.250000               51.526786
Male    39.806818           62.227273               48.511364
```

```
mydata.columns
```

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

```
mydata.corr()
```

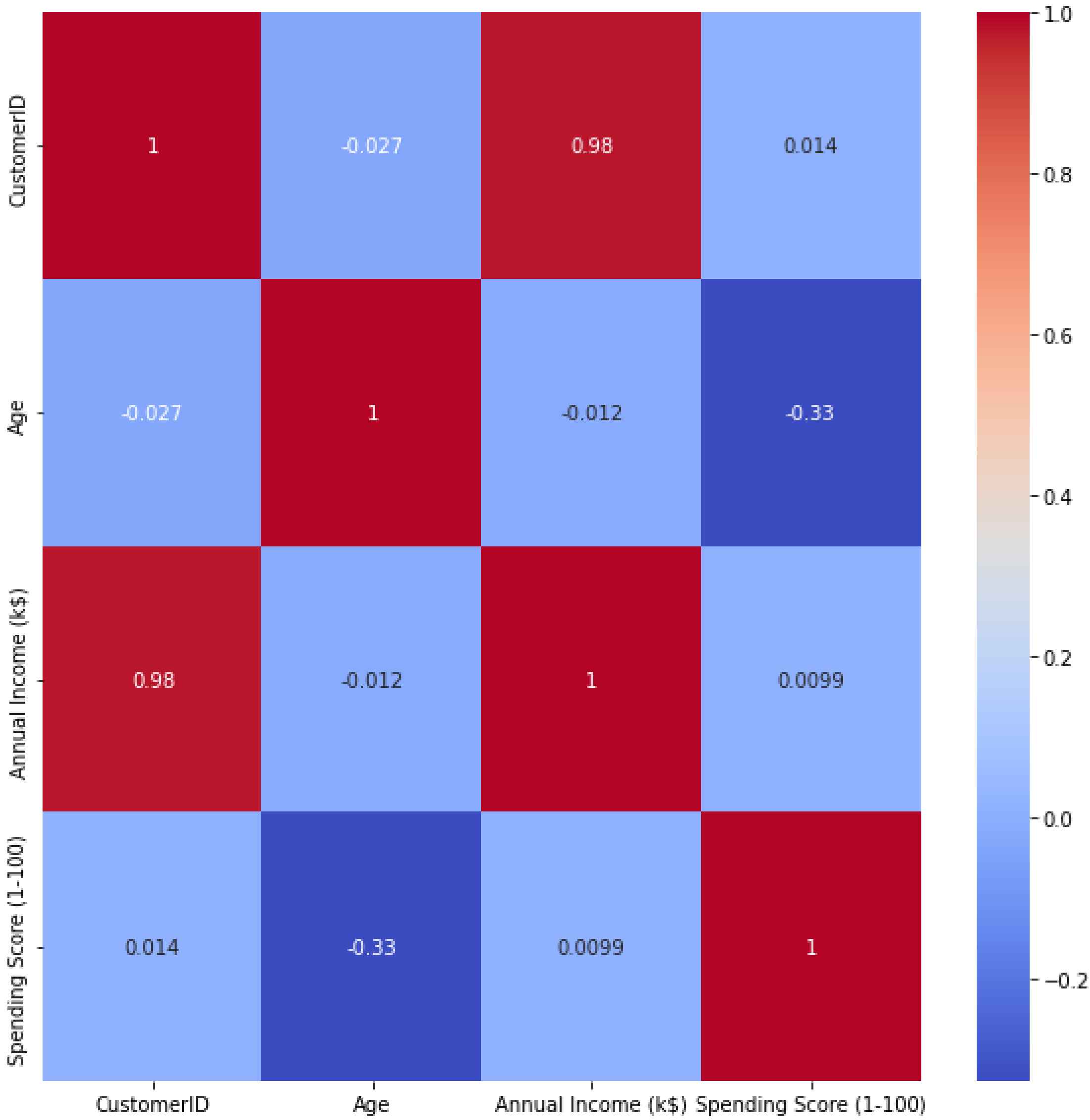
```
                        CustomerID       Age  Annual Income (k$)  \
CustomerID                1.000000 -0.026763            0.977548
Age                      -0.026763  1.000000           -0.012398
Annual Income (k$)        0.977548 -0.012398            1.000000
Spending Score (1-100)    0.013835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

```
                        Spending Score (1-100)
CustomerID                            0.013835
Age                                  -0.327227
Annual Income (k$)                    0.009903
Spending Score (1-100)                1.000000
```

**Multivariate Analysis**

```
plt.figure(figsize=(10, 10))
sns.heatmap(mydata.corr(),annot=True,cmap='coolwarm')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9dfedc3f90>
```




**4.Perform descriptive statistics on the dataset**

```
mydata.mean()
```

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

```
mydata.std()
```

```
CustomerID                57.879185
Age                       13.969007
Annual Income (k$)        26.264721
Spending Score (1-100)    25.823522
dtype: float64
```

```
mydata.sum()
```

```
CustomerID
20100
Gender
MaleMaleFemaleFemaleFemaleFemaleFemaleFemaleMa...
Age
7770
Annual Income (k$)
12112
Spending Score (1-100)
10040
dtype: object
```

```
mydata.count()
```

```
CustomerID                200
Gender                    200
Age                       200
Annual Income (k$)        200
Spending Score (1-100)    200
dtype: int64
```

```
mydata.min()
```

```
CustomerID                     1
Gender                    Female
Age                           18
Annual Income (k$)            15
Spending Score (1-100)         1
dtype: object
```

```
mydata.var()
```

```
CustomerID                3350.000000
Age                        195.133166
Annual Income (k$)         689.835578
```

```
Spending Score (1-100)     666.854271
dtype: float64
```

```
mydata.mode()
```

```
     CustomerID  Gender   Age  Annual Income (k$)  Spending Score (1-
100)
0             1  Female  32.0                54.0
42.0
1             2     NaN   NaN                78.0
NaN
2             3     NaN   NaN                 NaN
NaN
3             4     NaN   NaN                 NaN
NaN
4             5     NaN   NaN                 NaN
NaN
..          ...     ...   ...                 ...
...
195         196     NaN   NaN                 NaN
NaN
196         197     NaN   NaN                 NaN
NaN
197         198     NaN   NaN                 NaN
NaN
198         199     NaN   NaN                 NaN
NaN
199         200     NaN   NaN                 NaN
NaN

[200 rows x 5 columns]
```

```
mydata.median()
```

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

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

```
mydata.duplicated().sum()
```

```
0
```

```
mydata.isna().sum()
```

```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
```

```
Spending Score (1-100)    0
dtype: int64
```

```
mydata.nunique()
```

```
CustomerID                200
Gender                      2
Age                        51
Annual Income (k$)         64
Spending Score (1-100)     84
dtype: int64
```

```
mydata.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
```

```
mydata.sum()
```

```
CustomerID
20100
Gender
MaleMaleFemaleFemaleFemaleFemaleFemaleFemaleMa...
Age
7770
Annual Income (k$)
12112
Spending Score (1-100)
10040
dtype: object
```

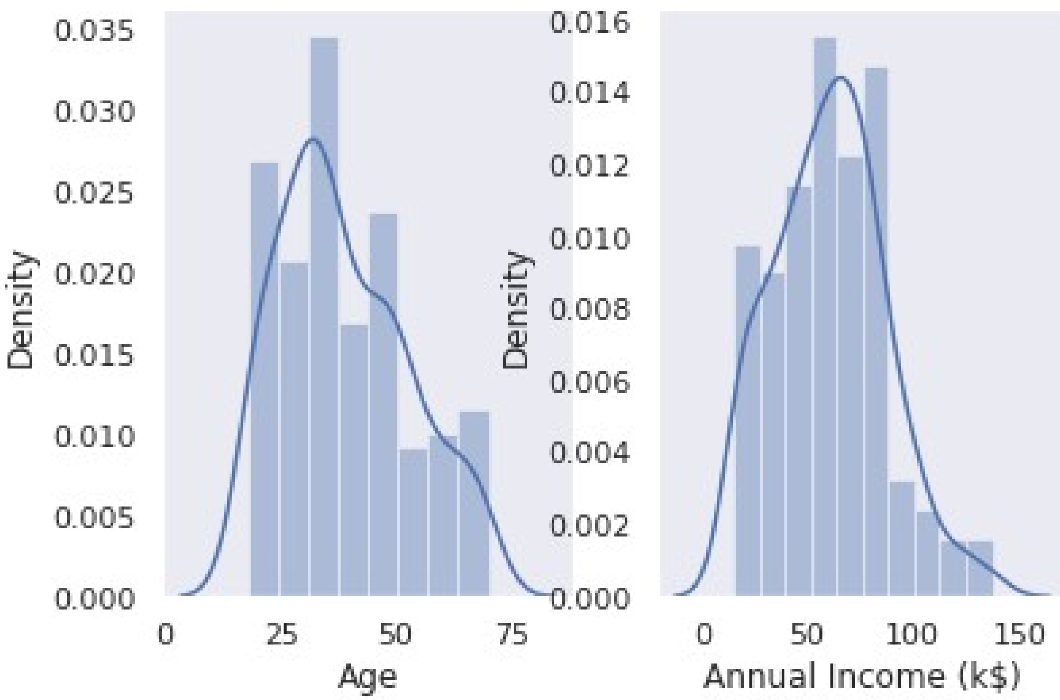
```
mydata.isnull().any()
```

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

**6. Find the outliers and replace them outliers**

```
sns.set(style = 'dark')
plt.subplot(1,2,1)
sns.distplot(mydata['Age'])
plt.subplot(1,2,2)
sns.distplot(mydata['Annual Income (k$)'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9dff5ab810>
```




```
print("Highest allowed",mydata['Age'].mean() + 3*mydata['Age'].std())
print("Lowest allowed",mydata['Age'].mean() - 3*mydata['Age'].std())
```

```
Highest allowed 80.75702199467665
Lowest allowed -3.0570219946766386
```

```
new_mydata = mydata[(mydata['Age'] < 50) & (mydata['Age'] > 25)]
new_mydata
```

```
     CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-
100)
4             5  Female   31                  17
40
6             7  Female   35                  18
6
9            10  Female   30                  19
72
11           12  Female   35                  19
99
14           15    Male   37                  20
13
```

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

[117 rows x 5 columns]
```

```
upper_limit = mydata['Age'].mean() + 5*mydata['Age'].std()
lower_limit = mydata['Age'].mean() - 5*mydata['Age'].std()

mydata['Age'] = np.where(
    mydata['Age']>upper_limit,
    upper_limit,
    np.where(
        mydata['Age']<lower_limit,
        lower_limit,
        mydata['Age']
    )
)

mydata['Age'].describe()
```

```
count    200.000000
mean      38.850000
std       13.969007
min       18.000000
25%       28.750000
50%       36.000000
75%       49.000000
max       70.000000
Name: Age, dtype: float64
```

**7. Check for Categorical columns and perform encoding.**

```
mydata.columns
```

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

```
mydata_numeric = mydata[['CustomerID','Age', 'Annual Income
(k$)','Spending Score (1-100)' ]]
mydata_categorical = mydata[['Gender']]
```

```
mydata_numeric.head()
```

```
   CustomerID   Age  Annual Income (k$)  Spending Score (1-100)
0           1  19.0                  15                      39
1           2  21.0                  15                      81
2           3  20.0                  16                       6
3           4  23.0                  16                      77
4           5  31.0                  17                      40
```

```
mydata_categorical.head()
```

```
   Gender
0    Male
1    Male
2  Female
3  Female
4  Female
```

```
print(mydata['CustomerID'].unique())
print(mydata['Gender'].unique())
print(mydata['Age'].unique())
print(mydata['Annual Income (k$)'].unique())
print(mydata['Spending Score (1-100)'].unique())
```

```
[  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
18
  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
36
  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
54
  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
72
  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
90
  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
108
 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126
 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
144
 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
162
 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
180
 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
198
 199 200]
['Male' 'Female']
[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.]
[ 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]
[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]
```

```
gender_encoder = LabelEncoder()
```

```
gender_encoder.fit(mydata_categorical['Gender'])
```

```
LabelEncoder()
```

```
gender_values = gender_encoder.transform(mydata_categorical['Gender'])
```

```
print("Before Encoding:", list(mydata_categorical['Gender'][-10:]))
print("After Encoding:", gender_values[-10:])
print("The inverse from the encoding result:",
gender_encoder.inverse_transform(gender_values[-10:]))
```

```
Before Encoding: ['Female', 'Female', 'Male', 'Female', 'Female',
'Female', 'Female', 'Male', 'Male', 'Male']
After Encoding: [0 0 1 0 0 0 0 1 1 1]
The inverse from the encoding result: ['Female' 'Female' 'Male'
'Female' 'Female' 'Female' 'Female' 'Male'
 'Male' 'Male']
```

### 8. Scaling the data

```
numCol  = [col for col in mydata.columns if  mydata[col].dtype != "O"]
```

```
numCol
```

```
['CustomerID', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']
```

```
catColumn  = [col for col in mydata.columns if  mydata[col].dtype ==
"O"]
catColumn
```

```
['Gender']
```

```
from sklearn.preprocessing import scale
X=mydata.drop(columns=['Age'],axis=1)
X.head()
```

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

```
scaler=MinMaxScaler()
```

```
mydata[["Age"]]=scaler.fit_transform(mydata[["Age"]])
print(mydata)
```

```
     CustomerID  Gender       Age  Annual Income (k$)  Spending Score
(1-100)
0             1    Male  0.019231                  15
39
1             2    Male  0.057692                  15
81
2             3  Female  0.038462                  16
6
3             4  Female  0.096154                  16
77
4             5  Female  0.250000                  17
40
..          ...     ...       ...                 ...
...
195         196  Female  0.326923                 120
79
196         197  Female  0.519231                 126
28
197         198    Male  0.269231                 126
74
198         199    Male  0.269231                 137
18
199         200    Male  0.230769                 137
83

[200 rows x 5 columns]
```

**9. Perform any of the clustering algorithms**

```
clustering1 = KMeans(n_clusters=6)
```

```
clustering1.fit(mydata[['Annual Income (k$)']])
```

```
KMeans(n_clusters=6)
```

```
clustering1.labels_
```

```
array([3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 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, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 2, 2, 2, 2, 2,
2,
       2, 2], dtype=int32)
```

```
mydata['Income Cluster']=clustering1.labels_
mydata.head()
```

```
   CustomerID  Gender       Age  Annual Income (k$)  Spending Score
(1-100)  \
0           1    Male  0.019231                  15
39
1           2    Male  0.057692                  15
81
2           3  Female  0.038462                  16
6
3           4  Female  0.096154                  16
77
4           5  Female  0.250000                  17
40

   Income Cluster
0               3
1               3
2               3
3               3
4               3
```

```
mydata['Income Cluster'].value_counts()
```

```
4    48
1    42
0    42
3    32
5    28
```

```
2     8
Name: Income Cluster, dtype: int64
```

```
clustering1.inertia_
```

```
5050.9047619047615
```

**10. Add the cluster data with the primary dataset**

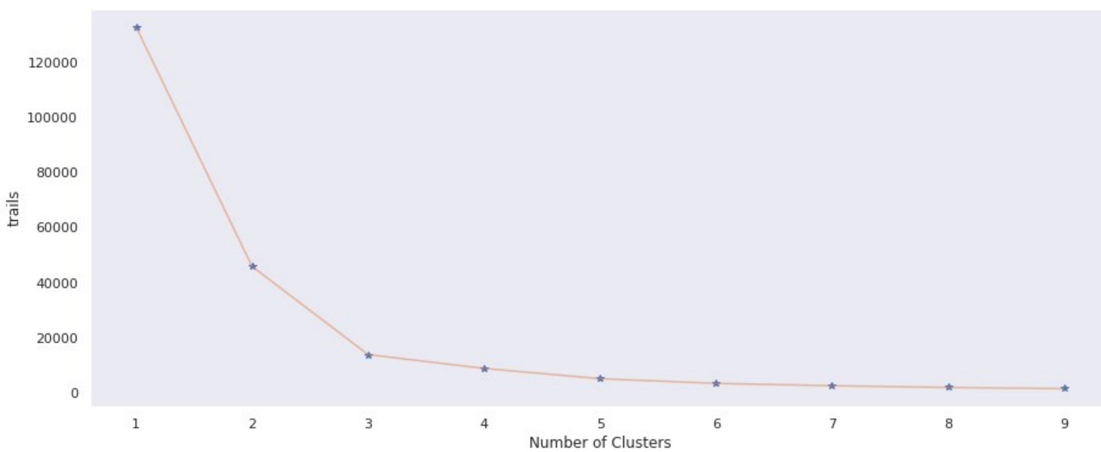
```
inertia_scores=[mydata]
for i in range(1,11):
  kmeans=KMeans(n_clusters=i)
  kmeans.fit(mydata[['Annual Income (k$)']])
  inertia_scores.append(kmeans.inertia_)
```

```
inertia_scores
```

```
[     CustomerID  Gender       Age  Annual Income (k$)  Spending Score
(1-100)  \
 0             1    Male  0.019231                  15
39
 1             2    Male  0.057692                  15
81
 2             3  Female  0.038462                  16
6
 3             4  Female  0.096154                  16
77
 4             5  Female  0.250000                  17
40
 ..          ...     ...       ...                 ...
...
 195         196  Female  0.326923                 120
79
 196         197  Female  0.519231                 126
28
 197         198    Male  0.269231                 126
74
 198         199    Male  0.269231                 137
18
 199         200    Male  0.230769                 137
83

      Income Cluster
 0                 3
 1                 3
 2                 3
 3                 3
 4                 3
 ..              ...
 195               2
 196               2
```

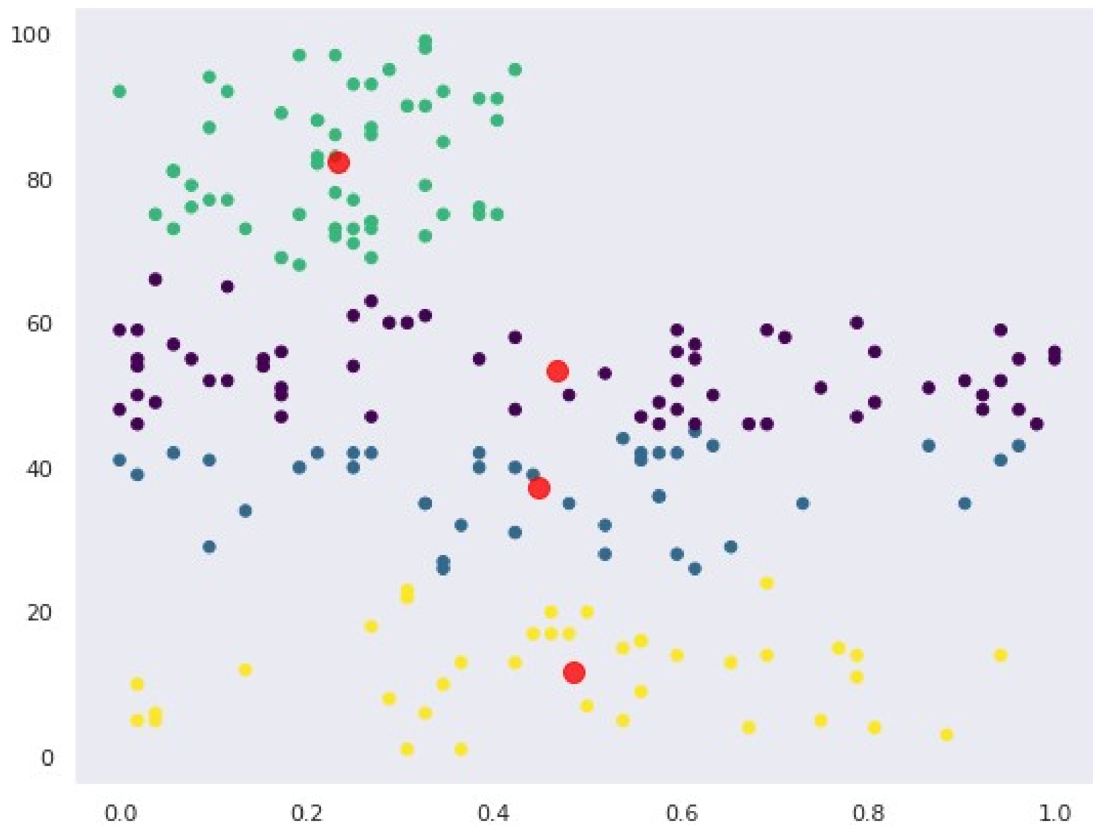
```
197                 2
198                 2
199                 2

[200 rows x 6 columns],
137277.28000000006,
48660.888888888876,
23517.33093093093,
13278.112713472483,
8481.49619047619,
5050.9047619047615,
3949.275613275612,
2822.4996947496966,
2168.4787157287165,
1748.8686813186823]
```

```
mydata_one = mydata[['Age','Spending Score (1-100)']]
data=[]
for n in range(1,10):
    kmeans = (KMeans(n_clusters = n ,init='k-means++', n_init =
10 ,max_iter=400,
                    tol=0.0001,  random_state= 45 ) )
    kmeans.fit(mydata_one)
    data.append(kmeans.inertia_)

plt.figure(1 , figsize = (15 ,6))
plt.plot(np.arange(1 , 10) , data , '*')
plt.plot(np.arange(1 , 10) , data , '-' , alpha = 0.5)
plt.xlabel('Number of Clusters') , plt.ylabel('trails')
plt.show()
```




```
kmeans = KMeans(n_clusters = 4, random_state = 45)
k_fit = kmeans.fit(mydata_one)
clusters = k_fit.labels_

plt.figure(figsize = [9, 7], clear = False)
clusters = k_fit.labels_
```

```
centers = k_fit.cluster_centers_
plt.scatter(mydata_one['Age'],mydata_one['Spending Score (1-100)'],c =
clusters,s = 30,cmap = "viridis")
plt.scatter(centers[:, 0],centers[:, 1],c = "red",s = 100,alpha =
0.8);
```




```
data = pd.DataFrame({"CustomerID": mydata['CustomerID'], "Group":
(k_fit.labels_ + 1)})
data.head(10)
```

| | CustomerID | Group |
|---|---|---|
| 0 | 1 | 2 |
| 1 | 2 | 3 |
| 2 | 3 | 4 |
| 3 | 4 | 3 |
| 4 | 5 | 2 |
| 5 | 6 | 3 |
| 6 | 7 | 4 |
| 7 | 8 | 3 |
| 8 | 9 | 4 |
| 9 | 10 | 3 |

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

```
# independent variable
X = mydata.iloc[:,0:4]
X.head()
```

```
   CustomerID       Age  Annual Income (k$)  Spending Score (1-100)
0           1  0.019231                  15                      39
1           2  0.057692                  15                      81
2           3  0.038462                  16                       6
3           4  0.096154                  16                      77
4           5  0.250000                  17                      40
```

```
# dependent variable
y = mydata.iloc[:,4:]
y.head()
```

```
   Income Cluster
0               3
1               3
2               3
3               3
4               3
```

**12. Split the data into training and testing**

```
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.3,random_state=1)
X_train.head()
```

```
     CustomerID       Age  Annual Income (k$)  Spending Score (1-100)
116         117  0.865385                  65                      43
67           68  0.961538                  48                      48
78           79  0.096154                  54                      52
42           43  0.576923                  39                      36
17           18  0.038462                  21                      66
```

```
X_test.head()
```

```
     CustomerID       Age  Annual Income (k$)  Spending Score (1-100)
58           59  0.173077                  46                      51
40           41  0.903846                  38                      35
34           35  0.596154                  33                      14
102         103  0.942308                  62                      59
184         185  0.442308                  99                      39
```

```
y_train.head()
```

```
     Income Cluster
116               4
67                1
78                4
42                1
17                3
```

```
y_test.head()
```

| | Income Cluster |
|---|---|
| 58 | 1 |
| 40 | 1 |
| 34 | 1 |
| 102 | 4 |
| 184 | 5 |

```
X_train
```

| | CustomerID | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| 116 | 117 | 0.865385 | 65 | 43 |
| 67 | 68 | 0.961538 | 48 | 48 |
| 78 | 79 | 0.096154 | 54 | 52 |
| 42 | 43 | 0.576923 | 39 | 36 |
| 17 | 18 | 0.038462 | 21 | 66 |
| .. | ... | ... | ... | ... |
| 133 | 134 | 0.250000 | 72 | 71 |
| 137 | 138 | 0.269231 | 73 | 73 |
| 72 | 73 | 0.807692 | 50 | 49 |
| 140 | 141 | 0.750000 | 75 | 5 |
| 37 | 38 | 0.230769 | 34 | 73 |

[140 rows x 4 columns]

```
X_test
```

| | CustomerID | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| 58 | 59 | 0.173077 | 46 | 51 |
| 40 | 41 | 0.903846 | 38 | 35 |
| 34 | 35 | 0.596154 | 33 | 14 |
| 102 | 103 | 0.942308 | 62 | 59 |
| 184 | 185 | 0.442308 | 99 | 39 |
| 198 | 199 | 0.269231 | 137 | 18 |
| 95 | 96 | 0.115385 | 60 | 52 |
| 4 | 5 | 0.250000 | 17 | 40 |
| 29 | 30 | 0.096154 | 29 | 87 |
| 168 | 169 | 0.346154 | 87 | 27 |
| 171 | 172 | 0.192308 | 87 | 75 |
| 18 | 19 | 0.653846 | 23 | 29 |
| 11 | 12 | 0.326923 | 19 | 99 |
| 89 | 90 | 0.615385 | 58 | 46 |
| 110 | 111 | 0.903846 | 63 | 52 |
| 118 | 119 | 0.634615 | 67 | 43 |
| 159 | 160 | 0.230769 | 78 | 73 |
| 35 | 36 | 0.057692 | 33 | 81 |
| 136 | 137 | 0.500000 | 73 | 7 |
| 59 | 60 | 0.673077 | 46 | 46 |
| 51 | 52 | 0.288462 | 42 | 60 |
| 16 | 17 | 0.326923 | 21 | 35 |

| | | | | |
|---|---|---|---|---|
| 44 | 45 | 0.596154 | 39 | 28 |
| 94 | 95 | 0.269231 | 60 | 42 |
| 31 | 32 | 0.057692 | 30 | 73 |
| 162 | 163 | 0.019231 | 81 | 5 |
| 38 | 39 | 0.346154 | 37 | 26 |
| 28 | 29 | 0.423077 | 29 | 31 |
| 193 | 194 | 0.384615 | 113 | 91 |
| 27 | 28 | 0.326923 | 28 | 61 |
| 47 | 48 | 0.173077 | 40 | 47 |
| 165 | 166 | 0.346154 | 85 | 75 |
| 194 | 195 | 0.557692 | 120 | 16 |
| 177 | 178 | 0.173077 | 88 | 69 |
| 176 | 177 | 0.769231 | 88 | 15 |
| 97 | 98 | 0.173077 | 60 | 50 |
| 174 | 175 | 0.653846 | 88 | 13 |
| 73 | 74 | 0.807692 | 50 | 56 |
| 69 | 70 | 0.269231 | 48 | 47 |
| 172 | 173 | 0.346154 | 87 | 10 |
| 108 | 109 | 0.961538 | 63 | 43 |
| 107 | 108 | 0.692308 | 63 | 46 |
| 189 | 190 | 0.346154 | 103 | 85 |
| 14 | 15 | 0.365385 | 20 | 13 |
| 56 | 57 | 0.634615 | 44 | 50 |
| 19 | 20 | 0.326923 | 23 | 98 |
| 114 | 115 | 0.000000 | 65 | 48 |
| 39 | 40 | 0.038462 | 37 | 75 |
| 185 | 186 | 0.230769 | 99 | 97 |
| 124 | 125 | 0.096154 | 70 | 29 |
| 98 | 99 | 0.576923 | 61 | 42 |
| 123 | 124 | 0.403846 | 69 | 91 |
| 119 | 120 | 0.615385 | 67 | 57 |
| 53 | 54 | 0.788462 | 43 | 60 |
| 33 | 34 | 0.000000 | 33 | 92 |
| 179 | 180 | 0.326923 | 93 | 90 |
| 181 | 182 | 0.269231 | 97 | 86 |
| 106 | 107 | 0.923077 | 63 | 50 |
| 199 | 200 | 0.230769 | 137 | 83 |
| 138 | 139 | 0.019231 | 74 | 10 |

Y_train

```
134    0
66     1
26     3
113    4
168    5
      ..
67     1
192    2
117    4
47     1
```

```
172    5
Name: Income Cluster, Length: 160, dtype: int32
```

**13. Build the Model**

```
# classification algorithm
from sklearn.svm import SVC
classifier_model = SVC(decision_function_shape='ovo')
```

**14.Train the model**

```
classifier_model.fit(X_train,y_train.values.flatten())
```

```
SVC(decision_function_shape='ovo')
```

**15. Test the model**

```
pred_y = classifier_model.predict(X_test)
pred_y[0:5]
```

```
array([1, 1, 3, 4, 5], dtype=int32)
```

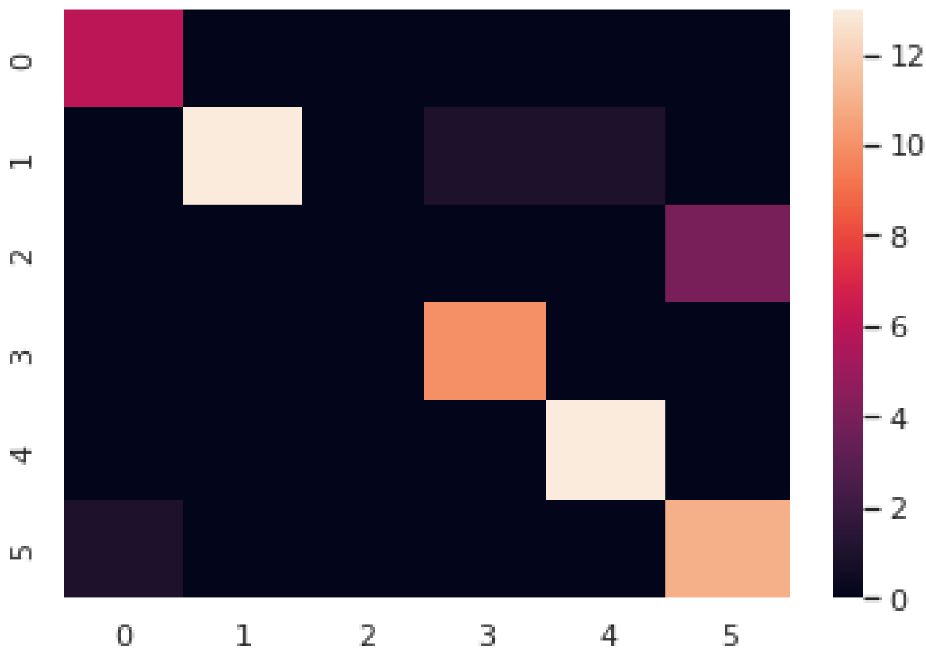
**16. Measure the performance using Evaluation Metrics**

```
print('Classification Report: ')
from sklearn.metrics import
classification_report,accuracy_score,f1_score,hamming_loss,confusion_m
atrix,roc_auc_score

print(classification_report(y_test, pred_y))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.86      1.00      0.92         6
           1       1.00      0.87      0.93        15
           2       0.00      0.00      0.00         4
           3       0.91      1.00      0.95        10
           4       0.93      1.00      0.96        13
           5       0.73      0.92      0.81        12

    accuracy                           0.88        60
   macro avg       0.74      0.80      0.76        60
weighted avg       0.84      0.88      0.85        60
```

```
print('Confusion Matrix: ')
sns.heatmap(confusion_matrix(y_test,pred_y))
```

```
Confusion Matrix:

<matplotlib.axes._subplots.AxesSubplot at 0x7f9df91125d0>
```




```
print('F1 Score: ',f1_score(y_test,pred_y, average='weighted'))
```

```
F1 Score:  0.8547856464523131
```

```
# Hamming loss gives the fraction of labels that are incorrectly
predicted
print('Hamming Loss: ',hamming_loss(y_test,pred_y))
```

```
Hamming Loss:  0.11666666666666667
```

```
print('Accuracy: ',accuracy_score(y_test,pred_y))
```

```
Accuracy:  0.8833333333333333
```