

DBSCAN

September 13, 2022

```
[1]: import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[2]: df=pd.read_csv("/content/Mall_Customers.csv")
```

```
[3]: df.head()
```

```
[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

```
[4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   CustomerID                            200 non-null    int64
1   Gender                                200 non-null    object
2   Age                                    200 non-null    int64
3   Annual Income (k$)                    200 non-null    int64
4   Spending Score (1-100)                200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
[5]: df.describe()
```

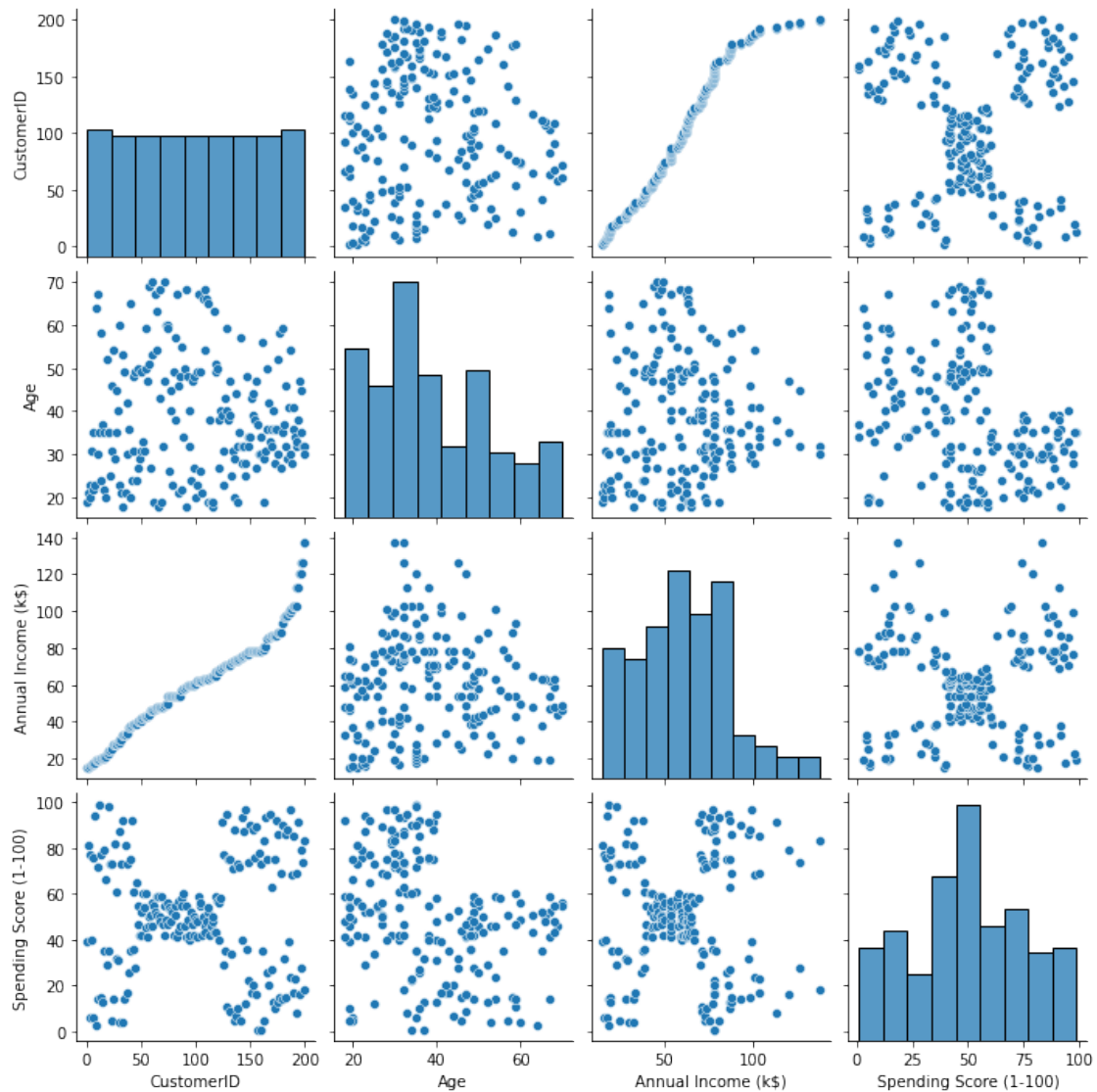
```
[5]:
```

	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

```
[6]: sns.pairplot(df)
```

```
[6]: <seaborn.axisgrid.PairGrid at 0x7f867bb554d0>
```



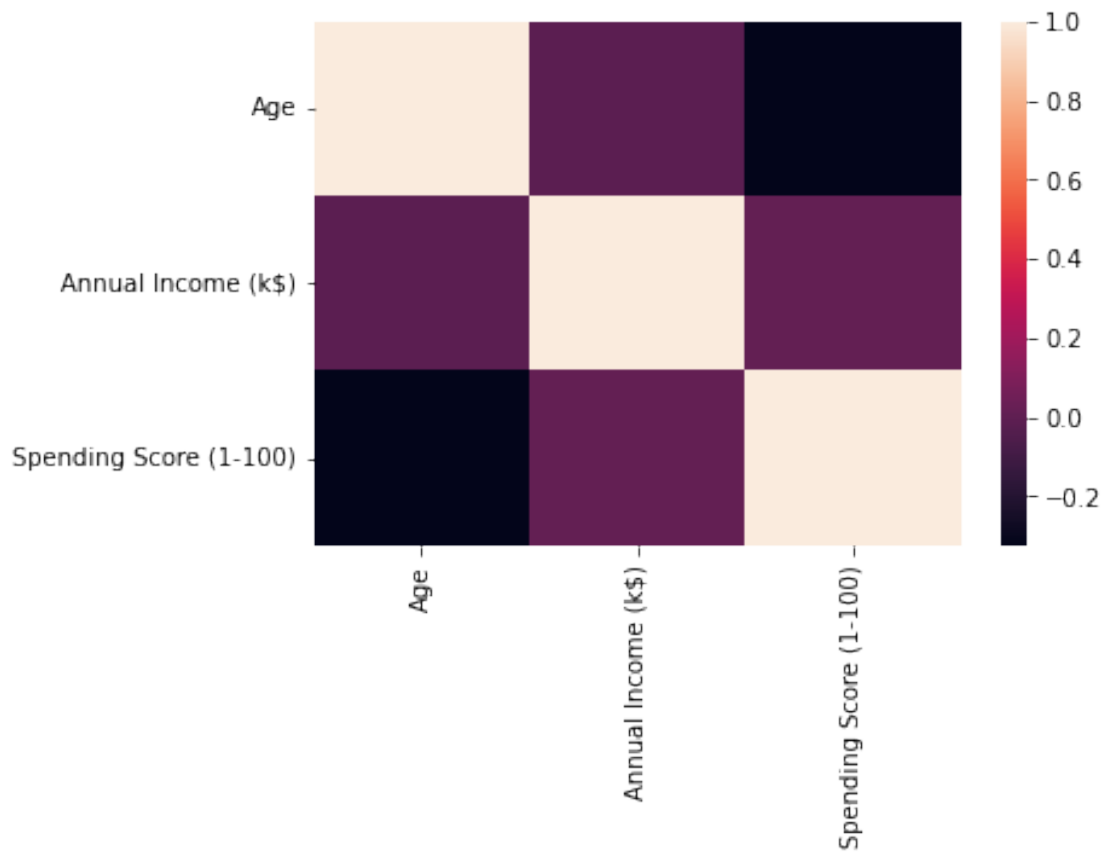
```
[7]: df=df.drop(['CustomerID'],axis=1)
df.head()
```

```
[7]:
```

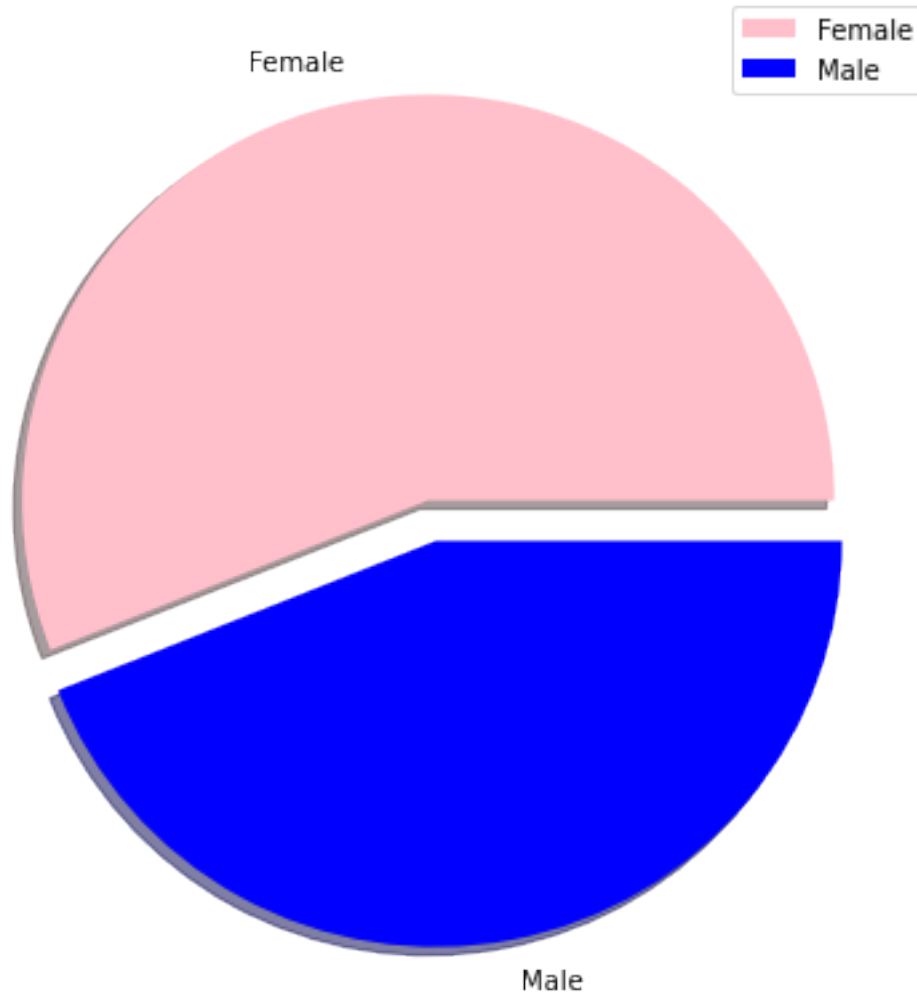
	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40

```
[8]: sns.heatmap(df.corr())
```

```
[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8678b71f90>
```



```
[9]: plt.figure(figsize=(7,7))
size=df['Gender'].value_counts()
label=['Female','Male']
color=['Pink','Blue']
explode=[0,0.1]
plt.pie(size,explode=explode,labels=label,colors=color,shadow=True)
plt.legend()
plt.show()
```

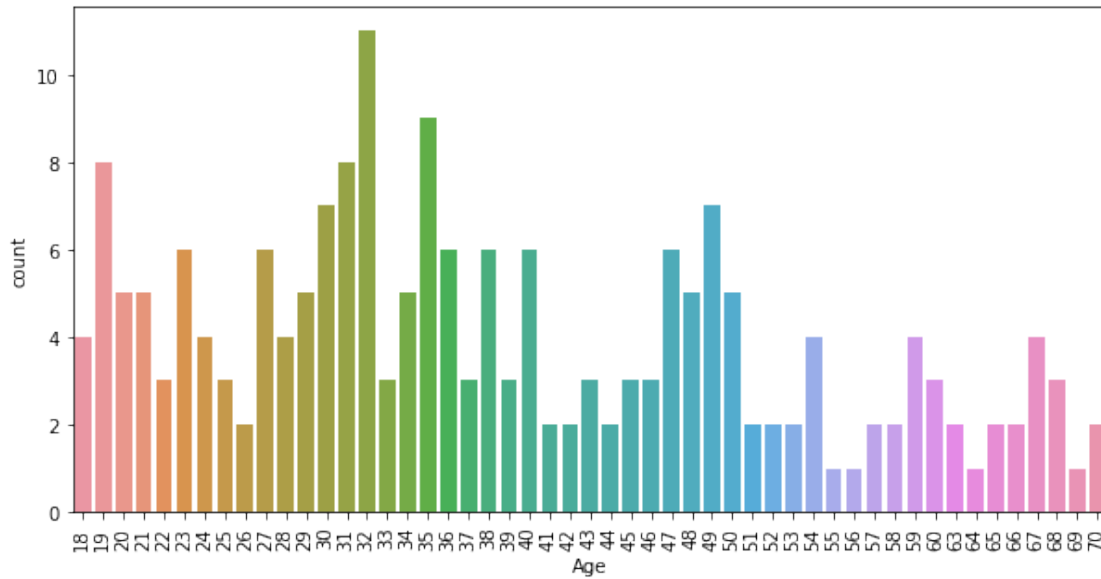


```
[10]: plt.figure(figsize=(10,5))
      sns.countplot(df['Age'])
      plt.xticks(rotation=90)
```

/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

```
[10]: (array([ 0,  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]),
      <a list of 51 Text major ticklabel objects>)
```

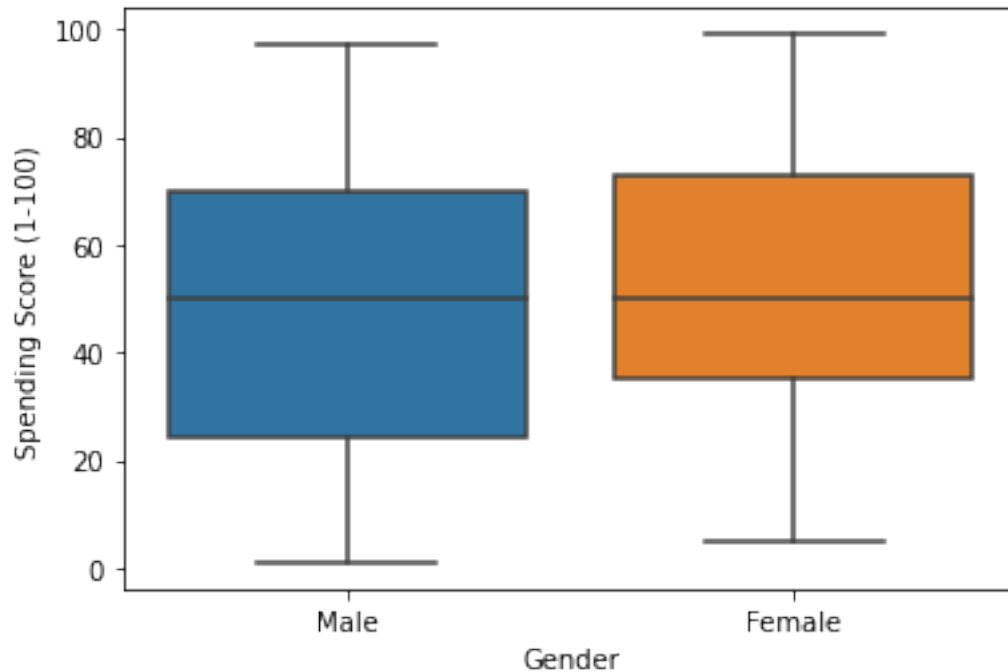


```
[12]: sns.boxplot(df['Gender'],df['Spending Score (1-100)'])
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. 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

```
[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8676bee9d0>
```

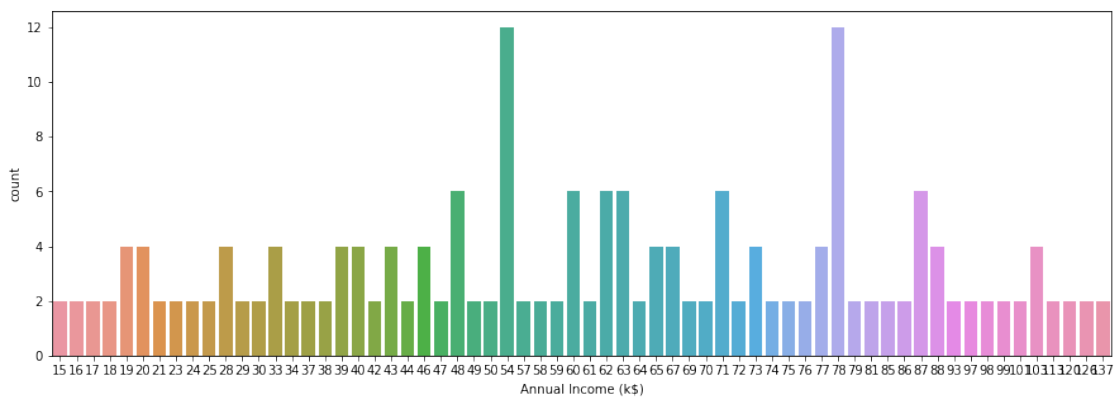


```
[14]: plt.figure(figsize=(15,5))
      sns.countplot(df['Annual Income (k$)'])
```

/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

```
[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8676985650>
```



DBSCAN

```
[15]: x=df.iloc[:,[2,3]].values  
      x.shape
```

```
[15]: (200, 2)
```

```
[16]: from sklearn.cluster import DBSCAN  
      db=DBSCAN(eps=3,min_samples=4,metric='euclidean')
```

```
[17]: model=db.fit(x)
```

```
[19]: label=model.labels_  
      label
```

```
[19]: array([-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  
          -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  
          -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  
          -1, -1, -1, -1, -1,  0,  0,  0,  0, -1, -1,  0, -1,  0, -1,  0,  0,  
          -1,  0, -1, -1,  0, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  
           1,  1,  1, -1,  2,  1,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  
           2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  3,  2,  
           3,  3, -1,  3, -1, -1,  4, -1, -1, -1,  4,  5,  4, -1,  4,  5, -1,  
           5,  4, -1,  4,  5, -1, -1,  6, -1, -1, -1,  7, -1,  6, -1,  6, -1,  
           7, -1,  6, -1,  7, -1,  7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  
           8, -1,  8, -1,  8, -1,  8, -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])
```

```
[20]: from sklearn import metrics  
  
      #identifying the points which makes up our core points  
      sample_cores=np.zeros_like(label,dtype=bool)  
  
      sample_cores[db.core_sample_indices_]=True  
  
      #Calculating the number of clusters  
  
      n_clusters=len(set(label))- (1 if -1 in label else 0)  
      print('No of clusters:',n_clusters)
```

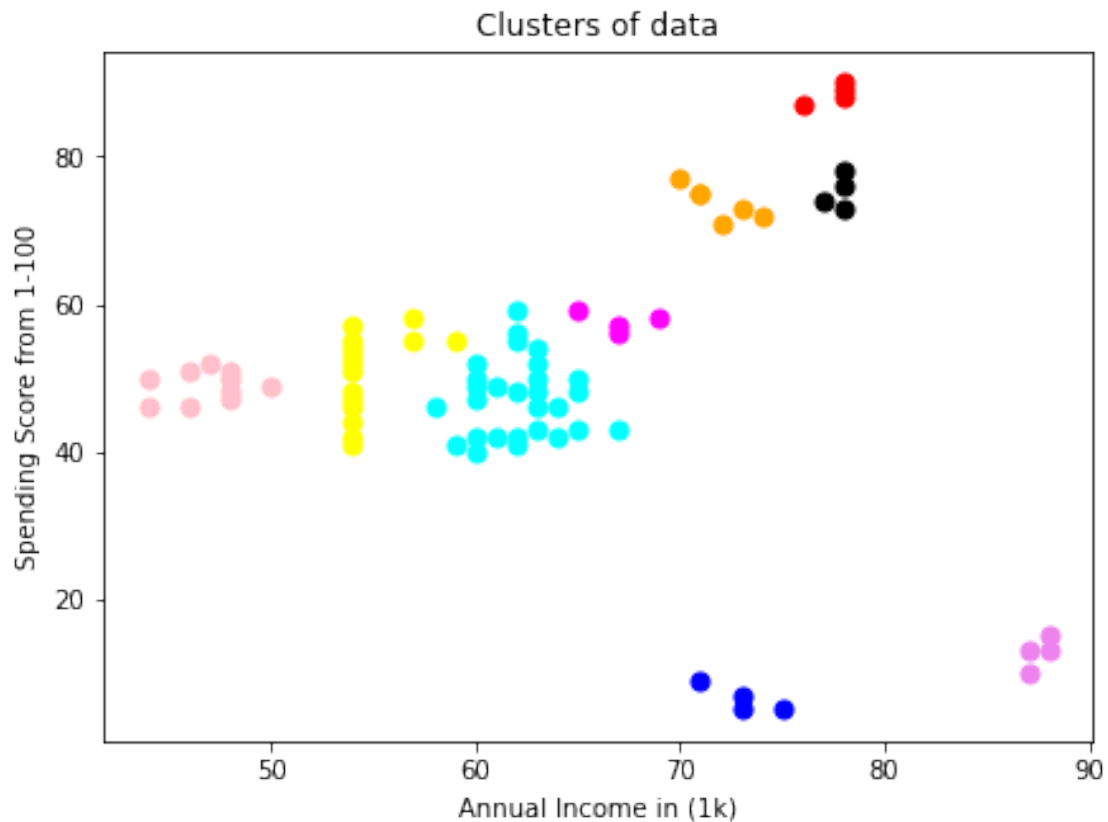
No of clusters: 9

```
[21]: y_means = db.fit_predict(x)  
      plt.figure(figsize=(7,5))  
      plt.scatter(x[y_means == 0, 0], x[y_means == 0, 1], s = 50, c = 'pink')  
      plt.scatter(x[y_means == 1, 0], x[y_means == 1, 1], s = 50, c = 'yellow')  
      plt.scatter(x[y_means == 2, 0], x[y_means == 2, 1], s = 50, c = 'cyan')  
      plt.scatter(x[y_means == 3, 0], x[y_means == 3, 1], s = 50, c = 'magenta')
```

```

plt.scatter(x[y_means == 4, 0], x[y_means == 4, 1], s = 50, c = 'orange')
plt.scatter(x[y_means == 5, 0], x[y_means == 5, 1], s = 50, c = 'blue')
plt.scatter(x[y_means == 6, 0], x[y_means == 6, 1], s = 50, c = 'red')
plt.scatter(x[y_means == 7, 0], x[y_means == 7, 1], s = 50, c = 'black')
plt.scatter(x[y_means == 8, 0], x[y_means == 8, 1], s = 50, c = 'violet')
plt.xlabel('Annual Income in (1k)')
plt.ylabel('Spending Score from 1-100')
plt.title('Clusters of data')
plt.show()

```

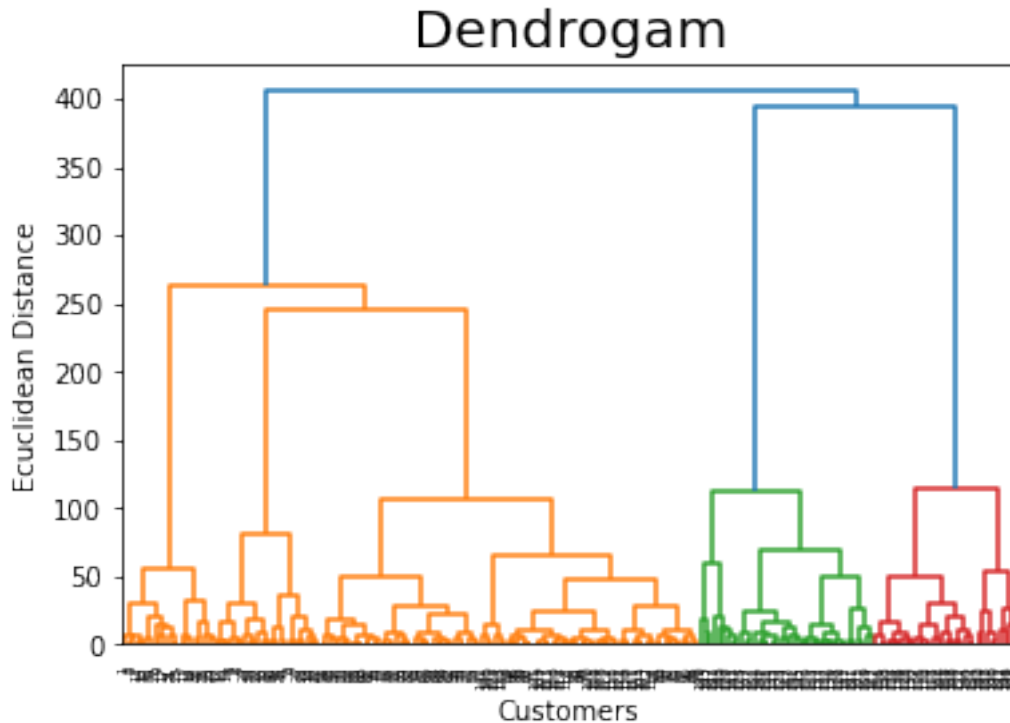


```

[22]: import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(x, method = 'ward'))
plt.title('Dendrogram', fontsize = 20)
plt.xlabel('Customers')
plt.ylabel('Euclidean Distance')
plt.show()

```

```
[24]: from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n_clusters = 9, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(x)

plt.scatter(x[y_hc == 0, 0], x[y_hc == 0, 1], s = 50, c = 'pink')
plt.scatter(x[y_hc == 1, 0], x[y_hc == 1, 1], s = 50, c = 'yellow')
plt.scatter(x[y_hc == 2, 0], x[y_hc == 2, 1], s = 50, c = 'cyan')
plt.scatter(x[y_hc == 3, 0], x[y_hc == 3, 1], s = 50, c = 'magenta')
plt.scatter(x[y_hc == 4, 0], x[y_hc == 4, 1], s = 50, c = 'orange')
plt.scatter(x[y_hc == 5, 0], x[y_hc == 5, 1], s = 50, c = 'blue')
plt.scatter(x[y_hc == 6, 0], x[y_hc == 6, 1], s = 50, c = 'red')
plt.scatter(x[y_hc == 7, 0], x[y_hc == 7, 1], s = 50, c = 'black')
plt.scatter(x[y_hc == 8, 0], x[y_hc == 8, 1], s = 50, c = 'violet')

plt.title('Hierarchical Clustering', fontsize = 20)
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.legend()
plt.grid()
plt.show()
```

WARNING:matplotlib.legend:No handles with labels found to put in legend.

