DBSCAN

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
df = pd.read_csv("E:\Mall_customers.csv")
```

In [3]:

```
df
```

Out[3]:

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

In [4]:

```
df.head()
```

Out[4]:

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

In [5]:

```
df.tail()
```

Out[5]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

In [8]:

```
from sklearn.preprocessing import LabelEncoder
Le=LabelEncoder()

df['Gender']=Le.fit_transform(df['Gender'])
df['Age']=Le.fit_transform(df['Age'])
df['Annual Income (k$)']=Le.fit_transform(df['Annual Income (k$)'])
df['Spending Score (1-100)']=Le.fit_transform(df['Spending Score (1-100)'])
```

In [15]:

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

Out[15]:

```
0
```

In [16]:

```
df1= df.iloc[:, [3,4]].values
```

In [17]:

```
df1
```

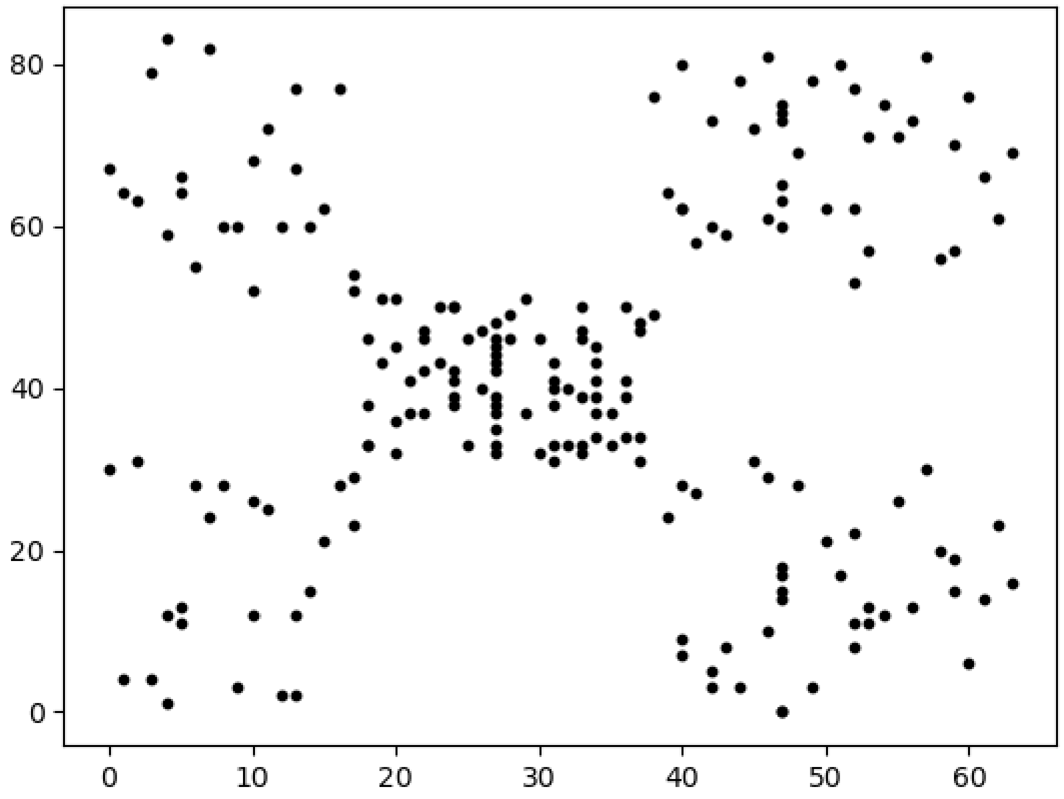
```
Out[17]: array([[ 0, 30],
                [ 0, 67],
                [ 1,  4],
                [ 1, 64],
                [ 2, 31],
                [ 2, 63],
                [ 3,  4],
                [ 3, 79],
                [ 4,  1],
                [ 4, 59],
                [ 4, 12],
                [ 4, 83],
                [ 5, 13],
                [ 5, 64],
                [ 5, 11],
                [ 5, 66],
                [ 6, 28],
                [ 6, 55],
                [ 7, 24],
                [ 7, 82],
                [ 8, 28],
                [ 8, 60],
                [ 9,  3],
                [ 9, 60],
                [10, 12],
                [10, 68],
                [10, 26],
                [10, 52],
                [11, 25],
                [11, 72],
                [12,  2],
                [12, 60],
                [13,  2],
                [13, 77],
                [13, 12],
                [13, 67],
                [14, 15],
                [14, 60],
                [15, 21],
                [15, 62],
                [16, 28],
                [16, 77],
                [17, 29],
                [17, 52],
                [17, 23],
                [17, 54],
                [18, 46],
                [18, 38],
                [18, 33],
                [18, 33],
                [19, 43],
                [19, 51],
                [20, 45],
                [20, 51],
                [20, 36],
                [20, 32],
                [21, 41],
                [21, 37],
                [22, 42],
                [22, 37],
                [22, 47],
                [22, 46],
                [23, 43],
                [23, 50],
            [24, 42],
            [24, 50],
            [24, 41],
            [24, 39],
            [24, 50],
            [24, 38],
            [25, 46],
            [25, 33],
            [26, 40],
            [26, 47],
            [27, 38],
            [27, 45],
            [27, 44],
            [27, 39],
            [27, 43],
            [27, 33],
            [27, 42],
            [27, 46],
            [27, 32],
            [27, 35],
            [27, 48],
            [27, 37],
            [28, 49],
            [28, 46],
            [29, 51],
            [29, 37],
            [30, 46],
            [30, 32],
            [31, 40],
            [31, 31],
            [31, 33],
            [31, 43],
            [31, 38],
            [31, 41],
            [32, 33],
            [32, 40],
            [33, 32],
            [33, 39],
            [33, 50],
            [33, 46],
            [33, 47],
            [33, 33],
            [34, 41],
            [34, 37],
            [34, 34],
            [34, 39],
            [34, 43],
            [34, 45],
            [35, 33],
            [35, 37],
            [36, 39],
            [36, 41],
            [36, 34],
            [36, 50],
            [37, 34],
            [37, 48],
            [37, 47],
            [37, 31],
            [38, 49],
            [38, 76],
            [39, 24],
            [39, 64],
            [40, 28],
            [40, 80],
[40,  9],
[40, 62],
[40,  7],
[40, 62],
[41, 27],
[41, 58],
[42,  3],
[42, 73],
[42,  5],
[42, 60],
[43,  8],
[43, 59],
[44,  3],
[44, 78],
[45, 31],
[45, 72],
[46, 10],
[46, 81],
[46, 29],
[46, 61],
[47, 18],
[47, 75],
[47, 15],
[47, 73],
[47, 17],
[47, 63],
[47, 14],
[47, 74],
[47,  0],
[47, 65],
[47,  0],
[47, 60],
[48, 28],
[48, 69],
[49,  3],
[49, 78],
[50, 21],
[50, 62],
[51, 17],
[51, 80],
[52, 22],
[52, 53],
[52, 11],
[52, 62],
[52,  8],
[52, 77],
[53, 11],
[53, 71],
[53, 13],
[53, 57],
[54, 12],
[54, 75],
[55, 26],
[55, 71],
[56, 13],
[56, 73],
[57, 30],
[57, 81],
[58, 20],
[58, 56],
[59, 15],
[59, 70],
[59, 19],
[59, 57],
       [60,  6],
       [60, 76],
       [61, 14],
       [61, 66],
       [62, 23],
       [62, 61],
       [63, 16],
       [63, 69]], dtype=int64)
```

In [11]:

```
plt.scatter(df1[:,0], df1[:,1], s=10, c= "black")
```

Out[11]:

```
<matplotlib.collections.PathCollection at 0x2703c6ed5e0>
```



In [12]:

```
from sklearn.cluster import KMeans
```

In [18]:

```
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters= i,
    init = 'k-means++', max_iter= 300, n_init= 10)
    kmeans.fit(df1)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11), wcss)
plt.title("The Elbow Method")
plt.xlabel("Number of clusters")
plt.ylabel("WCSS")
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_360\598975377.py in <module>
      3     kmeans = KMeans(n_clusters= i,
      4     init = 'k-means++', max_iter= 300, n_init= 10)
----> 5     kmeans.fit(df1)
      6     wcss.append(kmeans.inertia_)
      7 plt.plot(range(1,11), wcss)

~\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py in fit(self, X, y,
 sample_weight)
   1169         if self._algorithm == "full":
   1170             kmeans_single = _kmeans_single_lloyd
-> 1171             self._check_mkl_vcomp(X, X.shape[0])
   1172         else:
   1173             kmeans_single = _kmeans_single_elkan

~\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py in _check_mkl_vcomp
(self, X, n_samples)
   1026         active_threads = int(np.ceil(n_samples / CHUNK_SIZE))
   1027         if active_threads < self._n_threads:
-> 1028             modules = threadpool_info()
   1029             has_vcomp = "vcomp" in [module["prefix"] for module in
 modules]
   1030             has_mkl = ("mkl", "intel") in [

~\anaconda3\lib\site-packages\sklearn\utils\fixes.py in threadpool_info()
    323         return controller.info()
    324     else:
--> 325         return threadpoolctl.threadpool_info()
    326
    327

~\anaconda3\lib\site-packages\threadpoolctl.py in threadpool_info()
    122     In addition, each module may contain internal_api specific entri
es.
    123     """
--> 124     return _ThreadpoolInfo(user_api=_ALL_USER_APIS).todicts()
    125
    126

~\anaconda3\lib\site-packages\threadpoolctl.py in __init__(self, user_api, p
refixes, modules)
    338
    339             self.modules = []
--> 340             self._load_modules()
    341             self._warn_if_incompatible_openmp()
    342         else:

~\anaconda3\lib\site-packages\threadpoolctl.py in _load_modules(self)
    371             self._find_modules_with_dyld()
    372         elif sys.platform == "win32":
--> 373             self._find_modules_with_enum_process_module_ex()
    374         else:
    375             self._find_modules_with_dl_iterate_phdr()

~\anaconda3\lib\site-packages\threadpoolctl.py in _find_modules_with_enum_pr
ocess_module_ex(self)
    483
    484                 # Store the module if it is supported and selected
--> 485                 self._make_module_from_path(filepath)
    486         finally:
    487             kernel_32.CloseHandle(h_process)
~\anaconda3\lib\site-packages\threadpoolctl.py in _make_module_from_path(sel
f, filepath)
    513             if prefix in self.prefixes or user_api in self.user_api:
    514                 module_class = globals()[module_class]
--> 515                 module = module_class(filepath, prefix, user_api, in
ternal_api)
    516                 self.modules.append(module)
    517

~\anaconda3\lib\site-packages\threadpoolctl.py in __init__(self, filepath, p
refix, user_api, internal_api)
    604         self.internal_api = internal_api
    605         self._dynlib = ctypes.CDLL(filepath, mode=_RTLD_NOLOAD)
--> 606         self.version = self.get_version()
    607         self.num_threads = self.get_num_threads()
    608         self._get_extra_info()

~\anaconda3\lib\site-packages\threadpoolctl.py in get_version(self)
    644                                 lambda: None)
    645         get_config.restype = ctypes.c_char_p
--> 646         config = get_config().split()
    647         if config[0] == b"OpenBLAS":
    648             return config[1].decode("utf-8")

AttributeError: 'NoneType' object has no attribute 'split'
```

In [19]:

```
from sklearn.cluster import DBSCAN
```

In [20]:

```
dbscan = DBSCAN(eps=5, min_samples=5)
```

In [21]:

```
labels = dbscan.fit_predict(df)
```

In [22]:

```
np.unique(labels)
```

Out[22]:

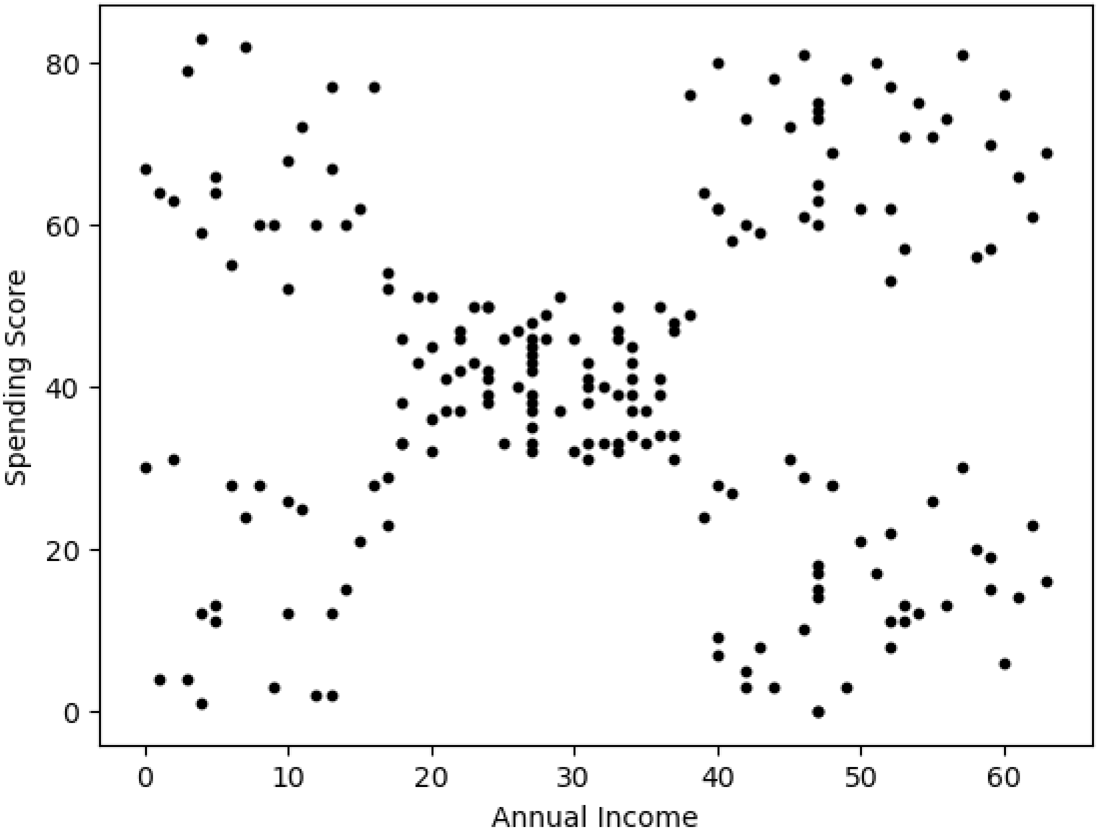
```
array([-1], dtype=int64)
```

In [25]:

```
# Visualising the clusters
plt.scatter(df1[labels == -1, 0], df1[labels == -1, 1], s = 10, c = 'black')

plt.scatter(df1[labels == 0, 0], df1[labels == 0, 1], s = 10, c = 'blue')
plt.scatter(df1[labels == 1, 0], df1[labels == 1, 1], s = 10, c = 'red')
plt.scatter(df1[labels == 2, 0], df1[labels == 2, 1], s = 10, c = 'green')
plt.scatter(df1[labels == 3, 0], df1[labels == 3, 1], s = 10, c = 'brown')
plt.scatter(df1[labels == 4, 0], df1[labels == 4, 1], s = 10, c = 'pink')
plt.scatter(df1[labels == 5, 0], df1[labels == 5, 1], s = 10, c = 'yellow')
plt.scatter(df1[labels == 6, 0], df1[labels == 6, 1], s = 10, c = 'silver')

plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.show()
```


80
60
40
20
0
Spending Score
0
10
20
30
40
50
60
Annual Income