

In [1]:

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

Bad key "text.kerning_factor" on line 4 in
C:\Users\91920\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib_classic_test_patch.mplstyle.
You probably need to get an updated matplotlibrc file from
<https://github.com/matplotlib/matplotlib/blob/v3.1.3/matplotlibrc.template>
(<https://github.com/matplotlib/matplotlib/blob/v3.1.3/matplotlibrc.template>
e)
or from the matplotlib source distribution

Here, I am creating a dataset with only two features so that we can visualize it easily. For creating the dataset I have created a function `PointsInCircum` which takes the radius and number of data points as arguments and returns an array of data points which when plotted forms a circle. We do this with the help of sin and cosine curves.

In [2]:

```
np.random.seed(42)

# Function for creating datapoints in the form of a circle
def PointsInCircum(r,n=100):
    return [(math.cos(2*math.pi/n*x)*r+np.random.normal(-30,30),math.sin(2*math.pi/n*x)*r+n
```

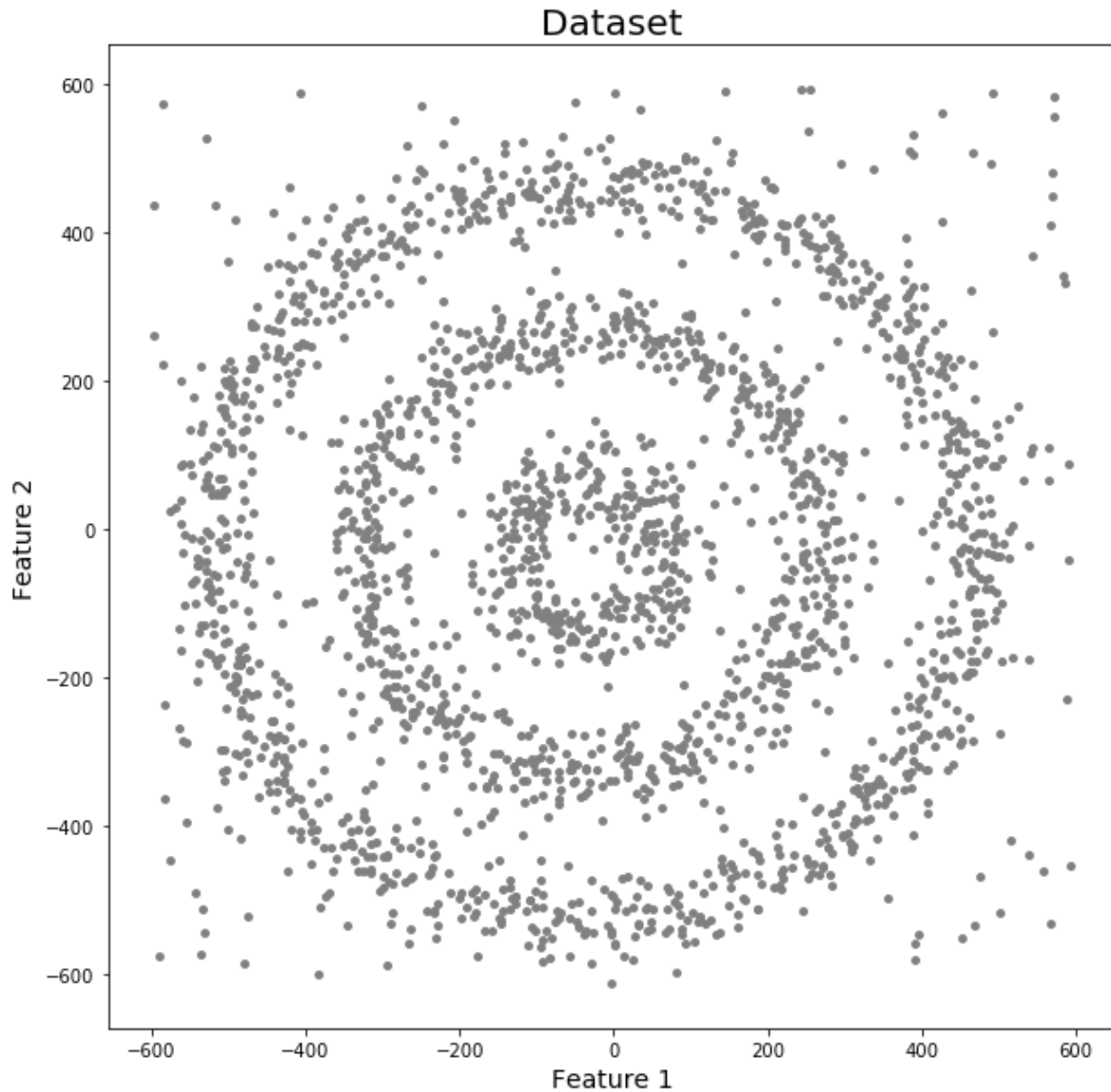
In [3]:

```
# Creating data points in the form of a circle
df=pd.DataFrame(PointsInCircum(500,1000))
df=df.append(PointsInCircum(300,700))
df=df.append(PointsInCircum(100,300))

# Adding noise to the dataset
df=df.append([(np.random.randint(-600,600),np.random.randint(-600,600)) for i in range(300)]
```

In [4]:

```
plt.figure(figsize=(10,10))
plt.scatter(df[0],df[1],s=15,color='grey')
plt.title('Dataset',fontsize=20)
plt.xlabel('Feature 1',fontsize=14)
plt.ylabel('Feature 2',fontsize=14)
plt.show()
```



In [5]:

```
from sklearn.cluster import KMeans
k_means=KMeans(n_clusters=4,random_state=42)
k_means.fit(df[[0,1]])
```

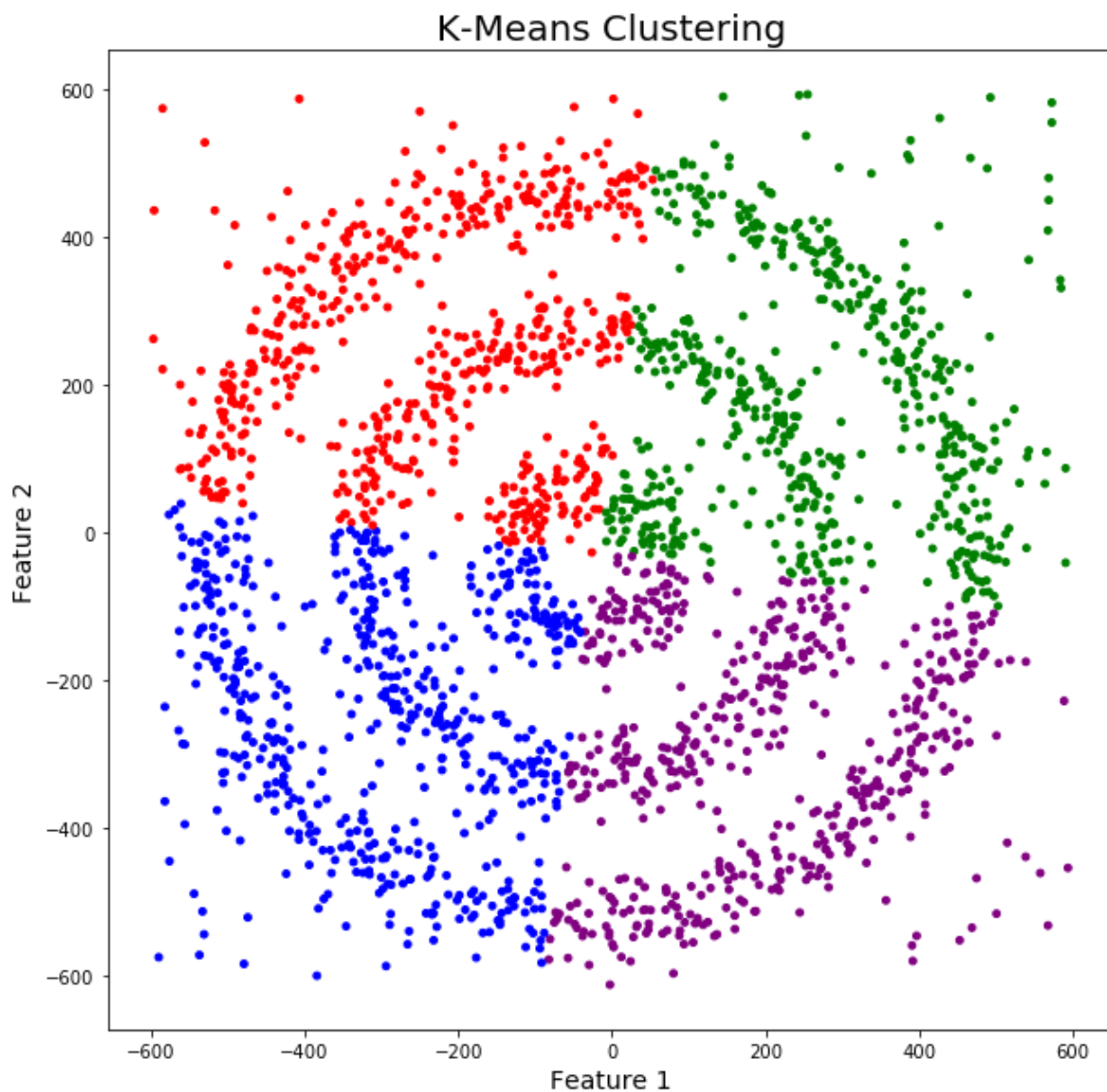
Out[5]:

KMeans(n_clusters=4, random_state=42)

In [6]:

```
df['KMeans_labels']=k_means.labels_

# Plotting resulting clusters
colors=['purple','red','blue','green']
plt.figure(figsize=(10,10))
plt.scatter(df[0],df[1],c=df['KMeans_labels'],cmap=matplotlib.colors.ListedColormap(colors))
plt.title('K-Means Clustering',fontsize=20)
plt.xlabel('Feature 1',fontsize=14)
plt.ylabel('Feature 2',fontsize=14)
plt.show()
```



In [7]:

```
from sklearn.cluster import AgglomerativeClustering
model = AgglomerativeClustering(n_clusters=4, affinity='euclidean')
model.fit(df[[0,1]])
```

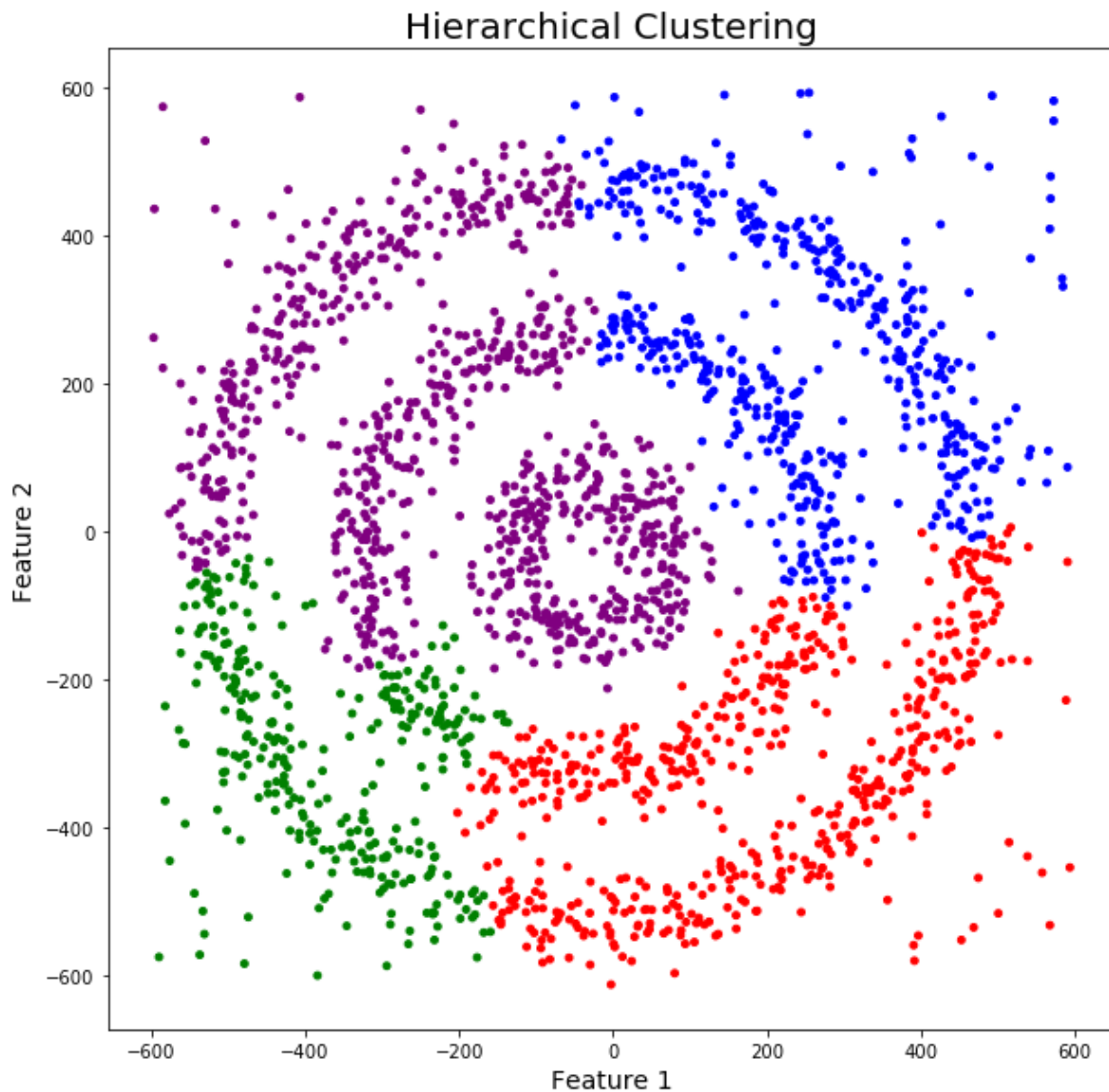
Out[7]:

AgglomerativeClustering(n_clusters=4)

In [8]:

```
df['HR_labels']=model.labels_

# Plotting resulting clusters
plt.figure(figsize=(10,10))
plt.scatter(df[0],df[1],c=df['HR_labels'],cmap=matplotlib.colors.ListedColormap(colors),s=1)
plt.title('Hierarchical Clustering',fontsize=20)
plt.xlabel('Feature 1',fontsize=14)
plt.ylabel('Feature 2',fontsize=14)
plt.show()
```



In [9]:

```
from sklearn.cluster import DBSCAN
dbscan=DBSCAN()
dbscan.fit(df[[0,1]])
```

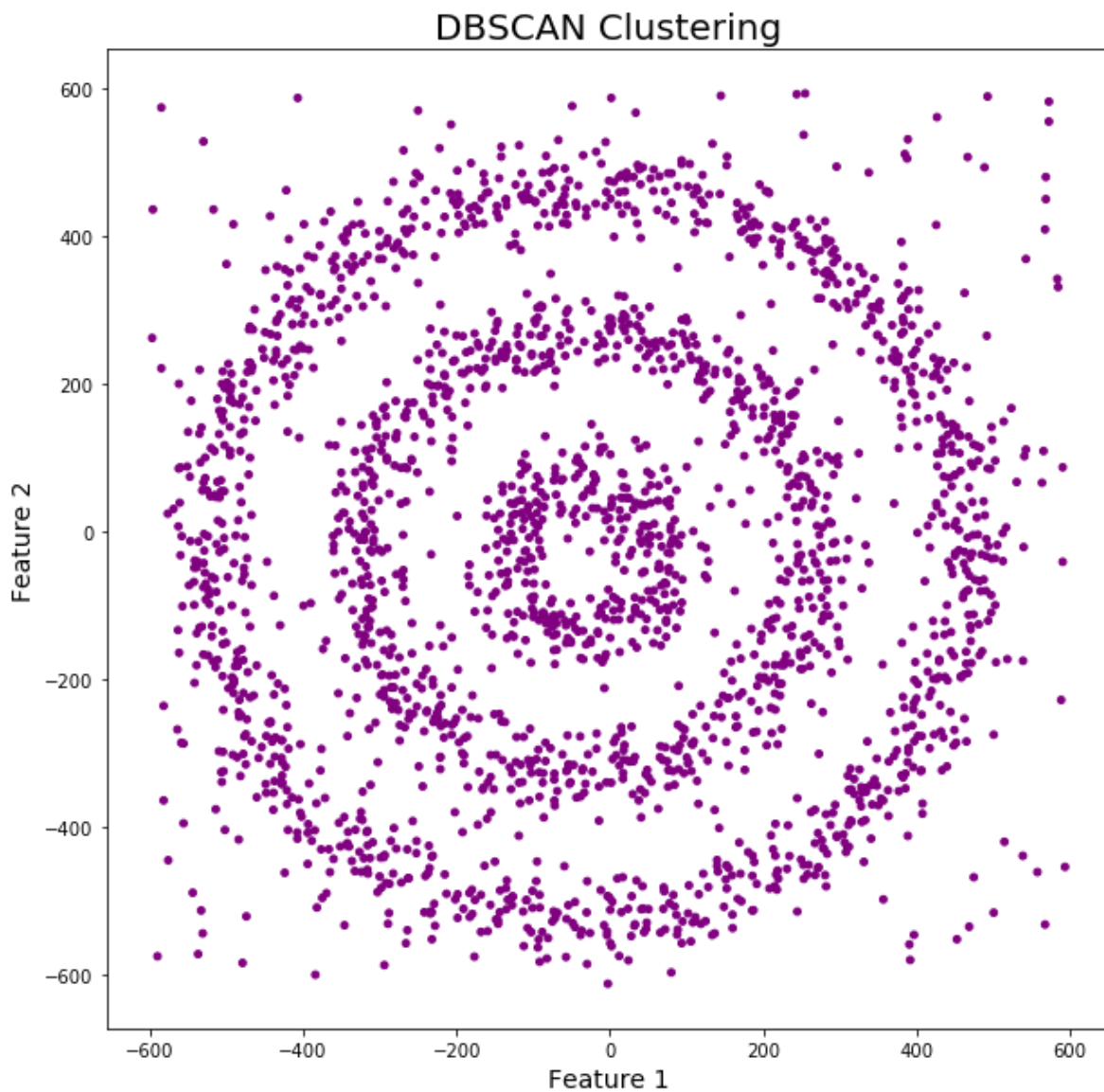
Out[9]:

DBSCAN()

In [10]:

```
df['DBSCAN_labels']=dbscan.labels_

# Plotting resulting clusters
plt.figure(figsize=(10,10))
plt.scatter(df[0],df[1],c=df['DBSCAN_labels'],cmap=matplotlib.colors.ListedColormap(colors))
plt.title('DBSCAN Clustering',fontsize=20)
plt.xlabel('Feature 1',fontsize=14)
plt.ylabel('Feature 2',fontsize=14)
plt.show()
```



In [11]:

```
from sklearn.neighbors import NearestNeighbors
neigh = NearestNeighbors(n_neighbors=2)
nbrs = neigh.fit(df[[0,1]])
distances, indices = nbrs.kneighbors(df[[0,1]])
```

In [14]:

```
#The distance variable contains an array of distances between
#a data point and its nearest data point for all data points in the dataset.
distances
```

Out[14]:

```
array([[ 0.          ,  6.91861074],
       [ 0.          , 10.80836526],
       [ 0.          ,  7.56030267],
       ...,
       [ 0.          , 25.77071121],
       [ 0.          , 23.34523506],
       [ 0.          , 37.73592453]])
```

In [13]:

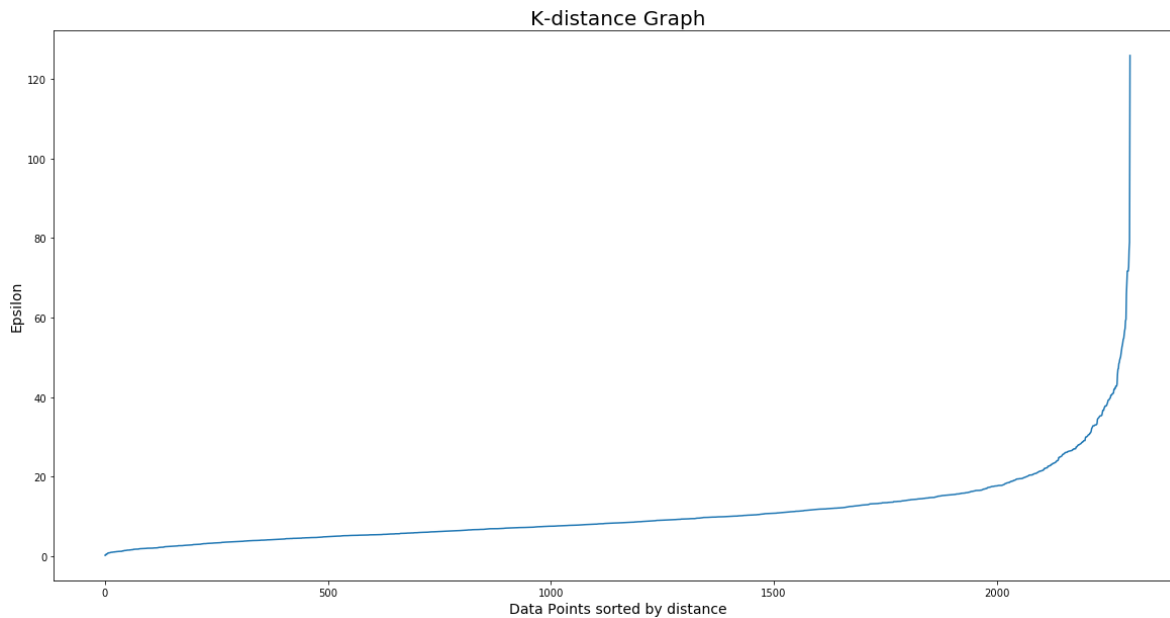
```
indices
```

Out[13]:

```
array([[ 0, 993],
       [ 1, 2237],
       [ 2,    5],
       ...,
       [2297, 210],
       [2298, 2139],
       [2299, 2059]], dtype=int64)
```

In [15]:

```
# Plotting K-distance Graph
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.figure(figsize=(20,10))
plt.plot(distances)
plt.title('K-distance Graph',fontsize=20)
plt.xlabel('Data Points sorted by distance',fontsize=14)
plt.ylabel('Epsilon',fontsize=14)
plt.show()
```



The optimum value of epsilon is at the point of maximum curvature in the K-Distance Graph, which is 30 in this case. Now, it's time to find the value of minPoints. The value of minPoints also depends on domain knowledge. This time I am taking minPoints as 6:

In [16]:

```
from sklearn.cluster import DBSCAN
dbscan_opt=DBSCAN(eps=30,min_samples=6)
dbscan_opt.fit(df[[0,1]])
```

Out[16]:

```
DBSCAN(eps=30, min_samples=6)
```

In [17]:

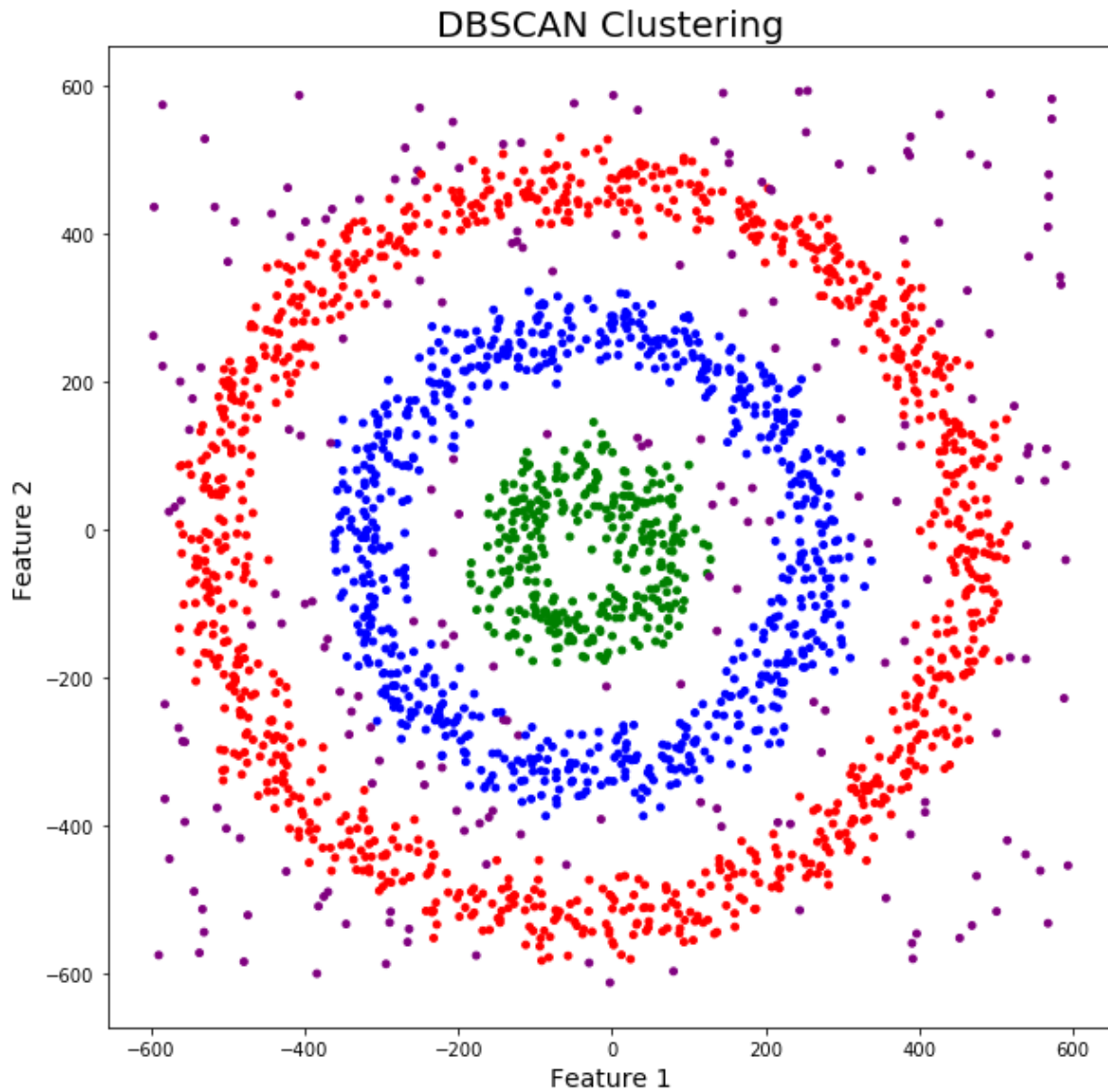
```
df['DBSCAN_opt_labels']=dbscan_opt.labels_
df['DBSCAN_opt_labels'].value_counts()
```

Out[17]:

```
0    1030
1     730
2     318
-1    222
Name: DBSCAN_opt_labels, dtype: int64
```


In [18]:

```
# Plotting the resulting clusters
plt.figure(figsize=(10,10))
plt.scatter(df[0],df[1],c=df['DBSCAN_opt_labels'],cmap=matplotlib.colors.ListedColormap(
plt.title('DBSCAN Clustering',fontsize=20)
plt.xlabel('Feature 1',fontsize=14)
plt.ylabel('Feature 2',fontsize=14)
plt.show()
```



In []: