5e-DBSCAN

November 13, 2024

1 DBSCAN

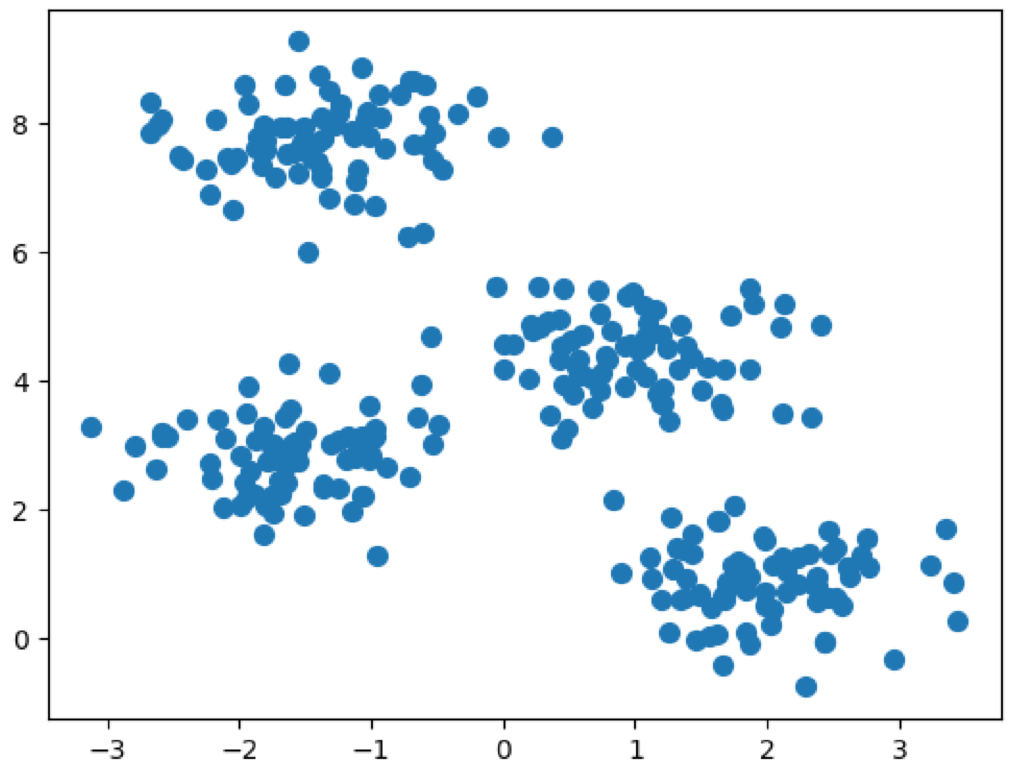
1.1 Imports

[1]:
```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_blobs
```

1.2 Blob Example

[2]:
```
X, y_true = make_blobs(n_samples=300, centers=4,cluster_std=0.60,␣
  ↪random_state=0)
plt.scatter(X[:, 0], X[:, 1], s=50)
;
```

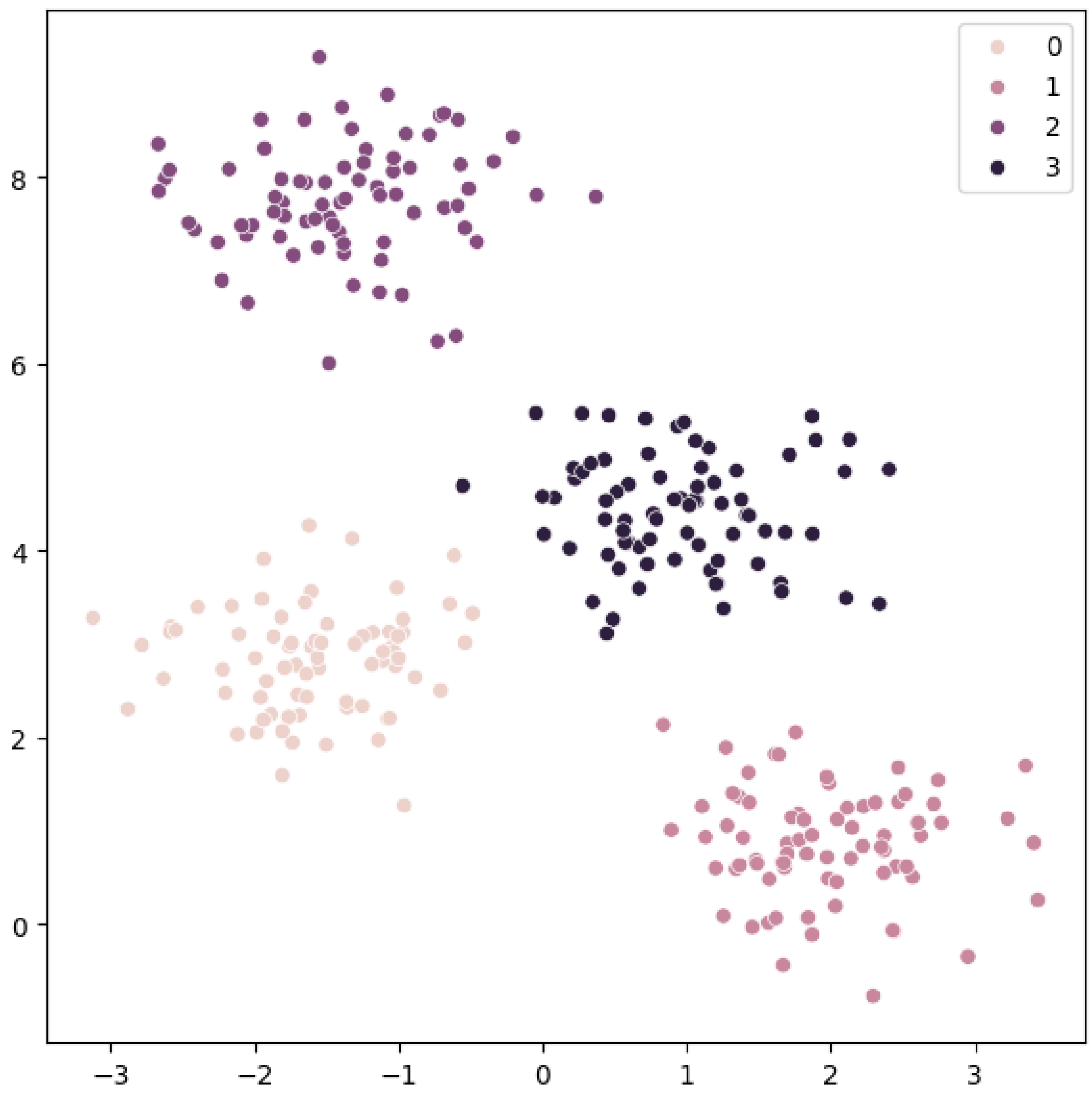
[2]:
```
''
```



1.2.1 K-Means

[3]:
```
kmeans = KMeans(n_clusters=4, n_init=10).fit(X)
```

[4]:
```
plt.figure(figsize = (7,7))
sns.scatterplot(x = X[:,0], y = X[:,1], hue = kmeans.labels_) ;
```

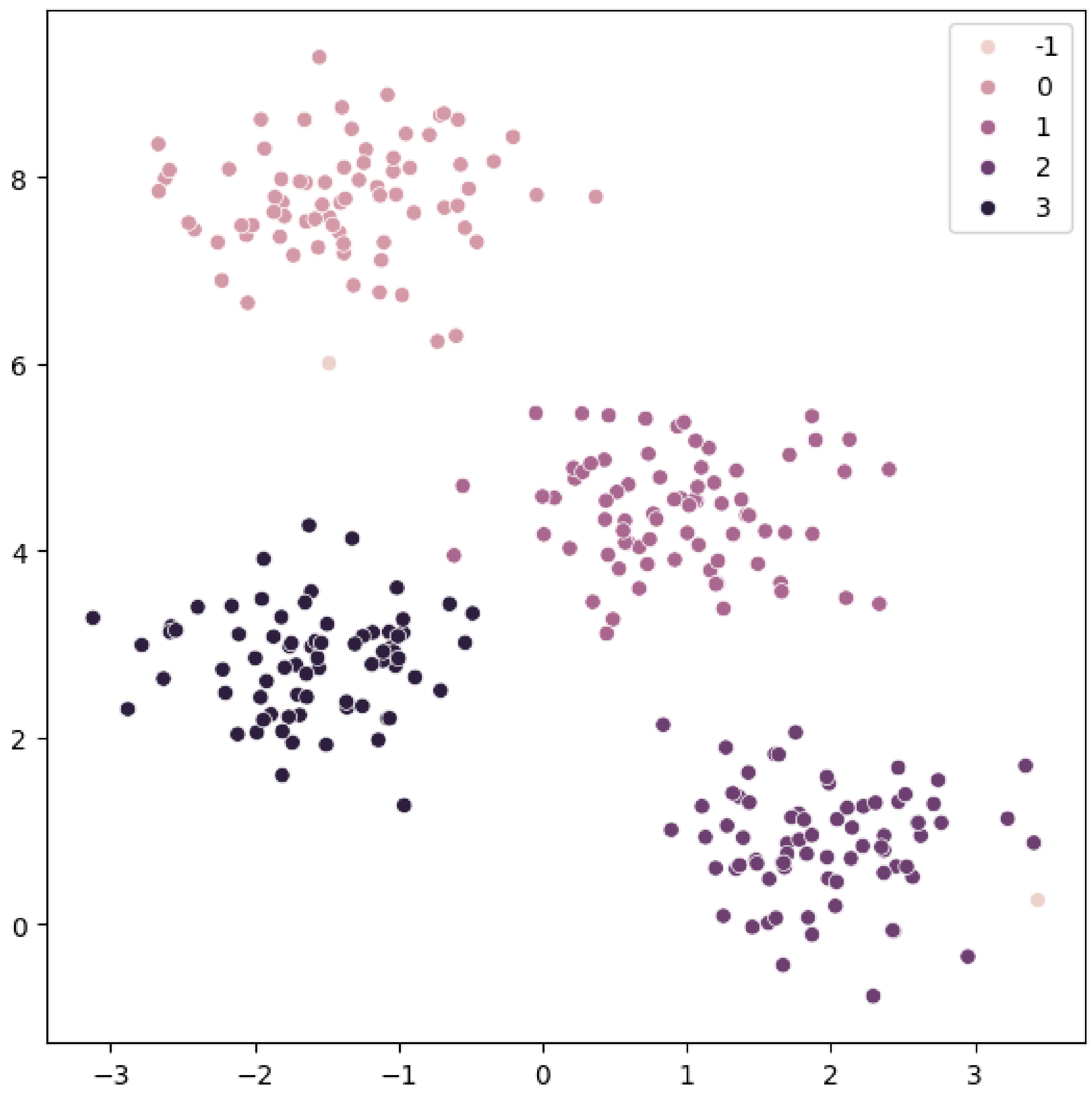


1.2.2 DBSCAN

[5]:
```
dbscan=DBSCAN(eps=.8,min_samples=9)
dbscan.fit(X)
;
```

[5]:
```
''
```

[6]:
```
plt.figure(figsize = (7,7))
sns.scatterplot(x = X[:,0], y = X[:,1], hue=dbscan.labels_) ;
```



```
[14]: dbscan.labels_
```

```
[14]: array([ 2,  0,  1,  0,  2, -1,  3,  1,  0,  0,  3,  0,  1,  0,  2,  1,  1,
             2,  3,  3,  2,  2,  1,  3,  3,  1,  2,  1,  3,  1,  0,  0,  1,  0,
             0,  0,  0,  0,  3,  2,  1,  3,  1,  1,  3,  3,  0,  3,  0,  2,  3,
             2,  0,  2,  2,  3,  0,  3,  0,  2,  0,  1,  0,  3,  3,  3,  0,  2,
             0,  3,  1,  3,  0,  3,  3,  0,  3,  1,  2,  0,  2,  1,  2,  2,  0,
             1,  2,  1,  0,  0,  1,  2,  0,  3,  3,  1,  2,  2,  1,  3,  0,  2,
             0,  2,  1,  2,  2,  1,  0,  1,  3,  3,  2,  0,  2,  1,  0,  2,  2,
             1,  3,  2,  3,  2,  2,  2,  2,  3,  2,  3,  0,  3,  3,  2,  0,  3,
             3,  0,  1,  0,  0,  3,  1,  3,  1,  3,  0,  1,  0,  0,  0,  1,  0,
             1,  2,  3,  0,  3,  2,  1,  0,  1,  1,  2,  1,  3,  3,  1,  2,  1,
             1,  0,  2,  1,  3,  0,  2,  2,  1,  3,  2,  1,  3,  3,  1,  1,  1,
             1,  2,  0,  1,  3,  1,  1,  3,  3,  3,  1,  3,  0,  1,  3,  2,  3,
             1, -1,  3,  0,  1,  0,  1,  3,  1,  1,  0,  3,  3,  2,  2,  1,  0,
             2,  2,  3,  2,  3,  1,  0,  0,  1,  1,  0,  1,  2,  3,  1,  2,  3,
             0,  3,  2,  1,  2,  0,  0,  0,  0,  3,  3,  0,  1,  3,  2,  1,  3,
             3,  1,  2,  2,  0,  1,  1,  3,  2,  0,  3,  1,  0,  1,  2,  2,  3,
             3,  1,  2,  2,  2,  1,  0,  0,  2,  2,  1,  2,  2,  2,  0,  3,  0,
             1,  2,  2,  0,  0,  0,  2,  2,  1,  0,  3])
```

[12]:
```
list(zip(*np.unique(dbscan.labels_, return_counts=True)))
```

[12]:
```
[(-1, 2), (0, 74), (1, 76), (2, 74), (3, 74)]
```

[17]:
```
dbscan.labels_
```

[17]:
```
5
```

1.3 Non-Blob Example

1.3.1 Create Random Data

[13]:
```
np.random.seed(100)

# Function for creating datapoints in the form of a circle
def PointsInCircum(r,n=100):
    '''This does math stuff'''
    return [ (math.cos(2*math.pi/n*x)*r+np.random.normal(-30,30),
              math.sin(2*math.pi/n*x)*r+np.random.normal(-30,30))
                for x in range(1,n+1)
           ]
```

[18]:
```
# Creating data points in the form of a circle
dfs = [ pd.DataFrame(PointsInCircum(500,1000)) ]
dfs[0].shape
```

[18]:
```
(1000, 2)
```

[19]:
```
# Add another circle inside
dfs += [ pd.DataFrame( PointsInCircum(300,700) ) ]
dfs[1].shape
```

[19]:
```
(700, 2)
```

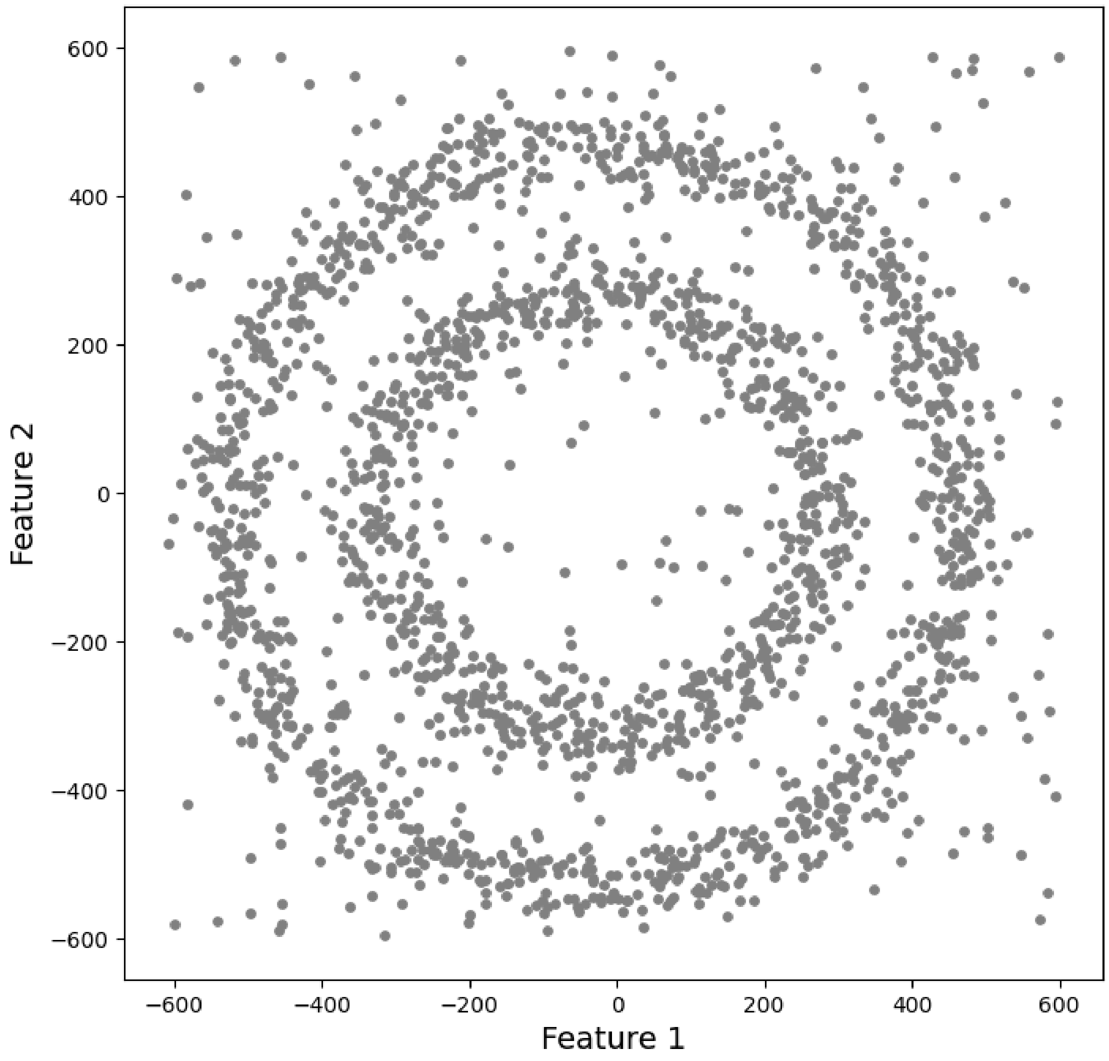
[20]:
```
# Adding noise to the dataset
dfs += [ pd.DataFrame( ( np.random.randint(-600,600), np.random.
 ↪randint(-600,600) ) for i in range(300) ) ]
dfs[2].shape
```

[20]:
```
(300, 2)
```

```
[21]: # Combine data sets
      df = pd.concat( dfs )
      df.shape
```

```
[21]: (2000, 2)
```

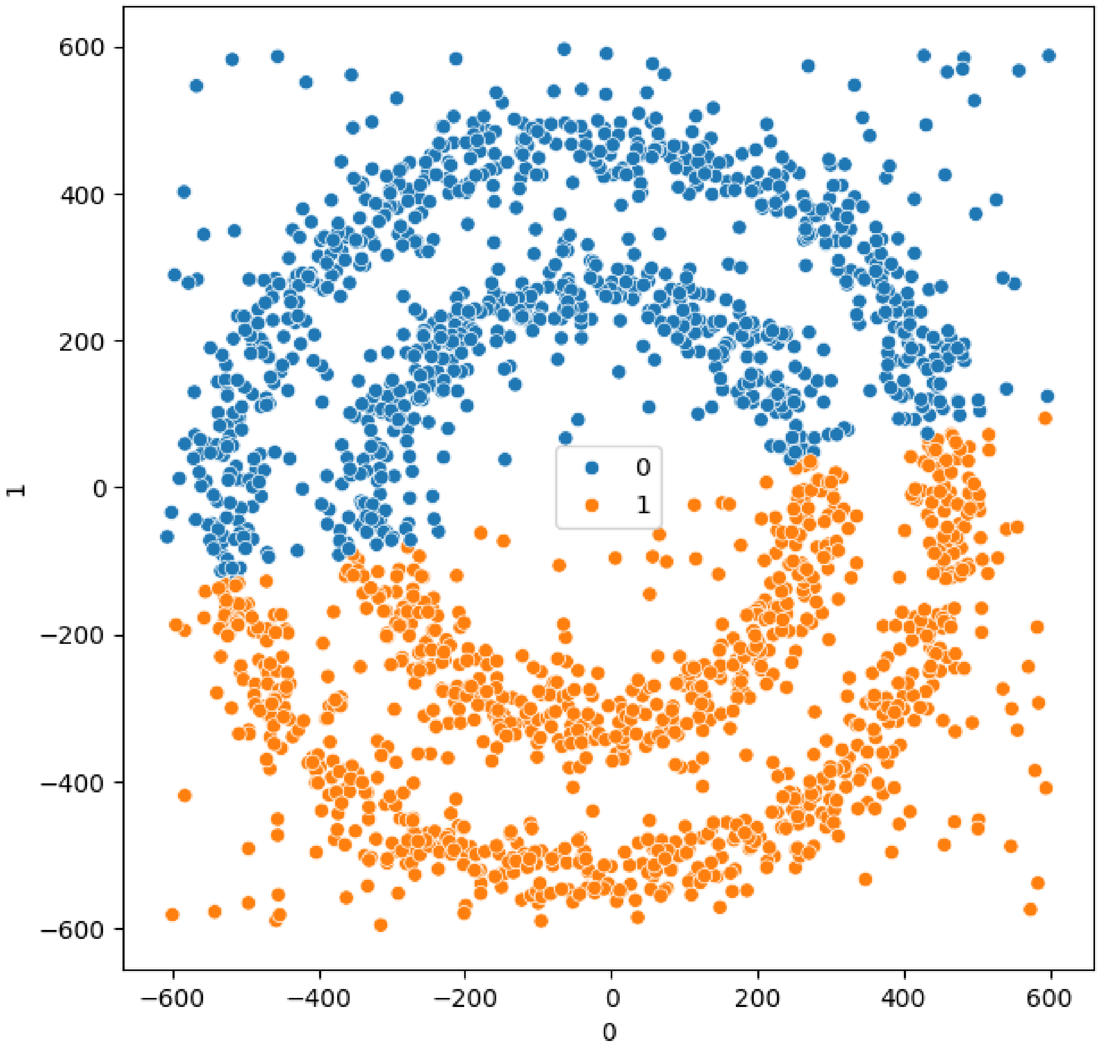
```
[22]: # Plotting data
      plt.figure(figsize=(8,8))
      plt.scatter(df[0],df[1],s=15,color='grey')
      plt.xlabel('Feature 1',fontsize=14)
      plt.ylabel('Feature 2',fontsize=14)
      plt.show() ;
```



1.3.2 K-means

[23]:
```
kmeans=KMeans(n_clusters=2, random_state=42, n_init=10).fit(df)
```

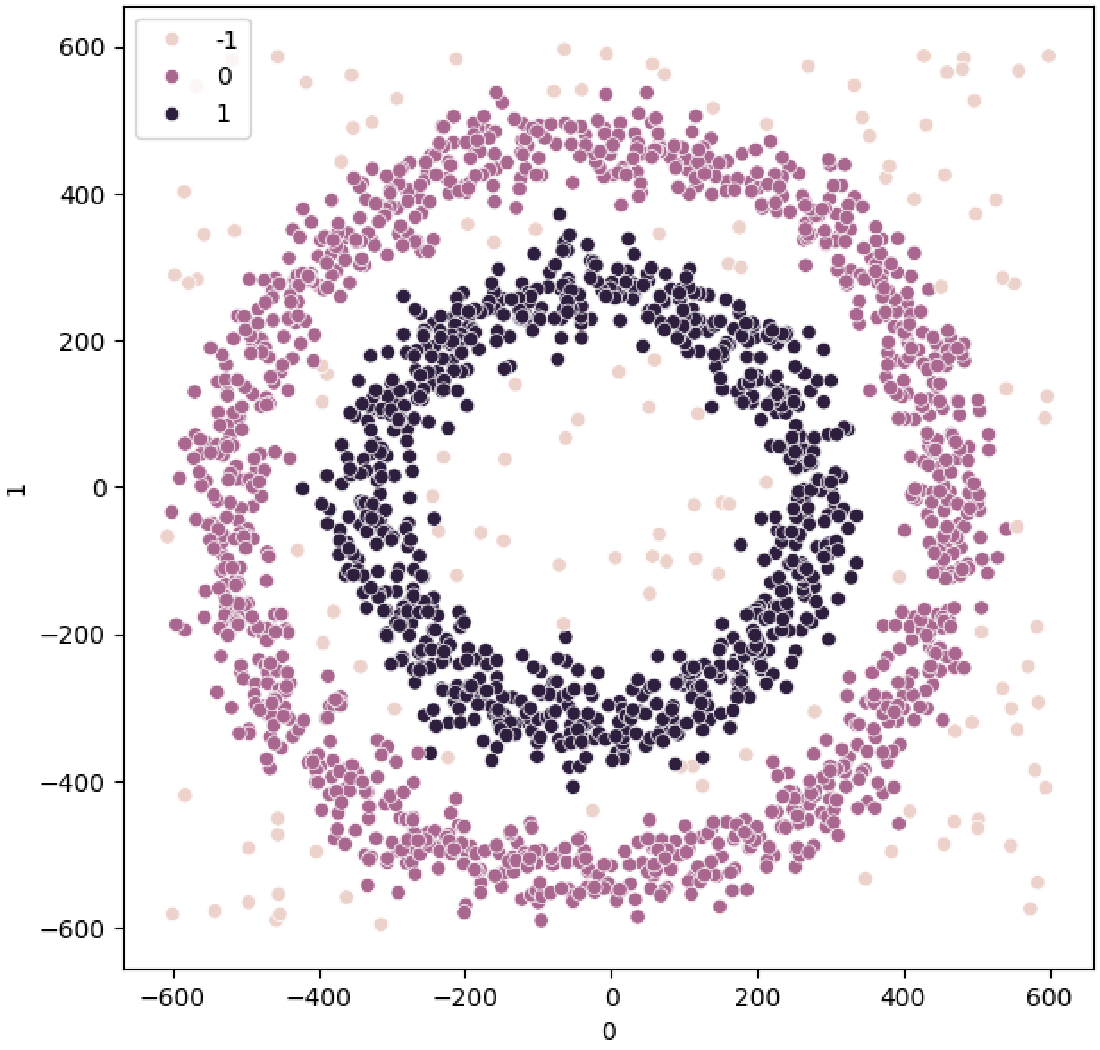
[24]:
```
plt.figure(figsize = (7,7))
sns.scatterplot(x = df[0], y = df[1], hue=kmeans.labels_) ;
```



1.3.3 DBSCAN

[25]:
```
dbscan=DBSCAN(eps=40, min_samples=7)
dbscan.fit(df) ;
```

[26]:
```
plt.figure(figsize = (7,7))
sns.scatterplot(x = df[0], y = df[1], hue=dbscan.labels_) ;
```


-1
0
1
600
400
200
0
−200
−400
−600
−600
−400
−200
0
200
400
600
0
1

```
[ ]:
```