DensityBasedScan_DBScan

April 15, 2023

This particluar method is a clustering method and this helps in clustering our data based on density, its a classification method in machine learning.

Two major inputs for the algorithm

```
-   epsilon - This is just the radius created with the center point (our data point)
-   Min Point - Default value for this is 3 in the algorithm and this suggests that there shoul
-   As discussed above , if this minimum three point criteria is fulfilled here then the centra
```

Boundary & Noise point

```
-   Suppose a data point has only one core point as neighbor and not satisfying the requirement
-   If a data point remains completely aloof and it's radius circle doesn't include any other d
```

[12]:
```python
#Lets try to apply dbscan on a created dataset
#importing the libraries
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
import numpy as np
```

[13]:
```python
# creating a numpy array
X = np.array([[1, 2], [2, 2], [2, 3],[8, 7], [8, 8], [25, 80]])
```

[14]:
```python
#applying dbscan on the data
clustering = DBSCAN(eps=3, min_samples=2).fit(X)
clustering.labels_
```

[14]:
```
array([ 0,  0,  0,  1,  1, -1], dtype=int64)
```

The above cluster label signifies as below

```
-   The first three datapoints belong to cluster 0
-   The fourth and fifth data points belong to cluster 1
-   The sixth data point is a noise here hence it's output came in negative
```

Now we will see what can happen if we change the parameter value here.

[15]:
```python
clustering = DBSCAN(eps=3, min_samples=9).fit(X)
clustering.labels_
```

[15]:
```
array([-1, -1, -1, -1, -1, -1], dtype=int64)
```

We can see that by changing the minimum sample number as 9 every single data point went into noise. Let's understand what happened here.. our total data point here is 9. And by changing the sample number we are asking our algoritham to make one sample for eveyr single data point , hence every single item acted as outlier/Noise.

Now we will see what can happen if we change the epsilon value here

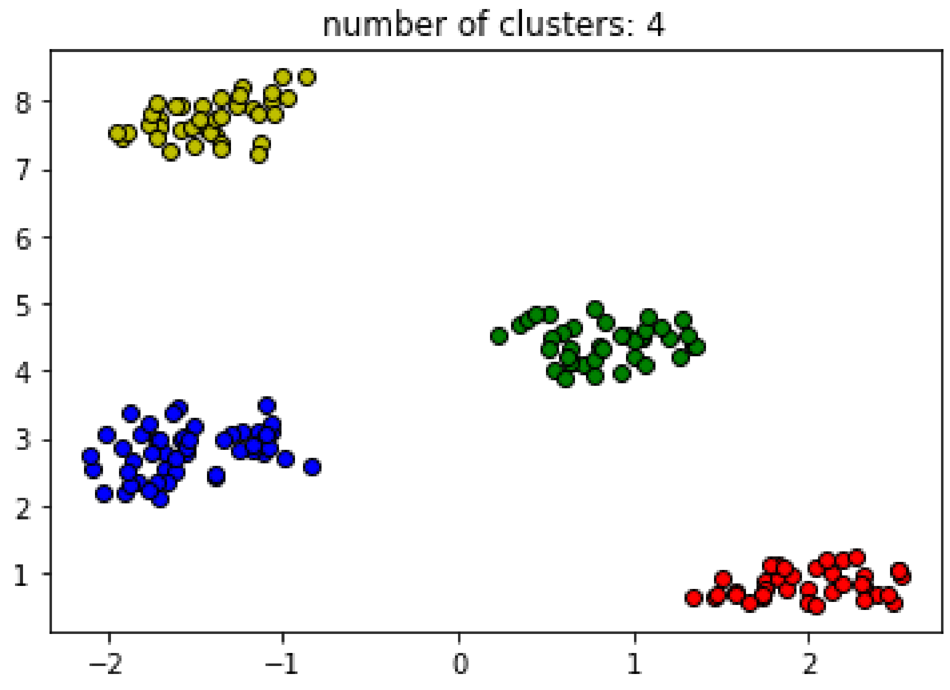
[16]:
```
clustering = DBSCAN(eps=100, min_samples=3).fit(X)
clustering.labels_
```

[16]:
```
array([0, 0, 0, 0, 0, 0], dtype=int64)
```

Now with the higher value of eps , the entire data point came into one cluster , which is cluster 0

[21]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler
from sklearn import datasets

# Load data in X
X, y_true = make_blobs(n_samples=300, centers=4,
                                        cluster_std=0.50, random_state=0)
db = DBSCAN(eps=0.3, min_samples=10).fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

print(labels)

# Plot result

# Black removed and is used for noise instead.
unique_labels = set(labels)
colors = ['y', 'b', 'g', 'r']
print(colors)
for k, col in zip(unique_labels, colors):
        if k == -1:
                # Black used for noise.
                col = 'k'

        class_member_mask = (labels == k)

        xy = X[class_member_mask & core_samples_mask]
        plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
                        markeredgecolor='k',
                        markersize=6)

        xy = X[class_member_mask & ~core_samples_mask]
        plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
                        markeredgecolor='k',
                        markersize=6)

plt.title('number of clusters: %d' % n_clusters_)
plt.show()
```

```
[-1  0 -1  0 -1 -1 -1  2 -1 -1  1 -1  2 -1 -1  2  2  3  1 -1 -1  3  2  1
  1 -1  3  2 -1 -1 -1  0 -1 -1  0 -1  0 -1  1  3 -1  1 -1 -1  1  1  0  1
  0 -1  1  3 -1  3 -1  1 -1 -1  0  3 -1  2 -1  1  1  1 -1  3 -1  1  2 -1
  0  1 -1  0  1  2  3  0 -1  2 -1  3  0 -1  3  2 -1  0  2  3 -1  1  1 -1
 -1  3 -1 -1 -1 -1 -1  3  2  3 -1  2 -1 -1 -1  1  3 -1  3 -1  0 -1 -1 -1
 -1  3  1  3 -1  3  3  1 -1  1 -1  1  1 -1  0 -1 -1  0 -1 -1 -1  1 -1 -1
 -1  1  0 -1  0  0  0  2 -1  2 -1  1  0  1  3  2 -1  2  2 -1  2 -1 -1 -1
 -1 -1  2  0  3 -1 -1  0 -1  3  2  1  3 -1  1  1  2  2 -1  2 -1  0 -1  1
  2  2  1  1 -1 -1  1  0 -1  1 -1  1 -1 -1  1 -1  2 -1  2  1 -1 -1  0  1
  1  3 -1  2  0  3  3 -1 -1 -1 -1 -1  0 -1  2 -1  2 -1  1  2  3  1 -1  1
 -1  2 -1  0  0  0  0  1  1 -1 -1  1  3  2  1 -1 -1 -1  3  0  2  2 -1  3
 -1  1 -1 -1 -1  3  3 -1  1 -1 -1 -1 -1 -1  0  0 -1  3 -1  3  3 -1  0 -1
 -1  2 -1  3  0 -1  0 -1 -1  2 -1  1]
['y', 'b', 'g', 'r']
```



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

[2]:
```
df = pd.read_csv('Mall_Customers.csv')
X_train = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
```

[3]:
```
clustering = DBSCAN(eps=12.5, min_samples=4).fit(X_train)
DBSCAN_dataset = X_train.copy()
DBSCAN_dataset.loc[:,'Cluster'] = clustering.labels_
```

[4]:
```
DBSCAN_dataset.Cluster.value_counts().to_frame()
```

[4]:
```
    Cluster
 0      112
 2       34
 3       24
-1       18
 1        8
 4        4
```

[ ]:
```

```