dbscan

May 15, 2023

[1]:
```
from sklearn.datasets import make_moons
import numpy as np
import matplotlib.pyplot as plt
```

0.1 Create Dataset

Menggunakan make moons disini mengcreate dataset 100

[2]:
```
X, y = make_moons(n_samples=1000, noise=0.05, random_state=42)
```

[3]:
```
from sklearn.cluster import DBSCAN
```

Fitting dataset dengan DBSCAN dengan nilai epsilon (luas jangkauan core adalah 0.05 dan minimal sampe disekitar instance 5

[4]:
```
dbscan = DBSCAN(eps=0.05, min_samples=5)
dbscan.fit(X)
```

[4]:
```
DBSCAN(algorithm='auto', eps=0.05, leaf_size=30, metric='euclidean',
       metric_params=None, min_samples=5, n_jobs=None, p=None)
```

[5]:
```
dbscan.labels_[:10]
```

[5]:
```
array([ 0,  2, -1, -1,  1,  0,  0,  0,  2,  5])
```

[6]:
```
len(dbscan.core_sample_indices_)
```

[6]:
```
808
```

[7]:
```
dbscan.core_sample_indices_[:10] # hanya 10 data pertama
```

[7]:
```
array([ 0,  4,  5,  6,  7,  8, 10, 11, 12, 13])
```

[8]:
```
dbscan.components_[:3] # hanya 3 data pertama
```

[8]:
```
array([[-0.02137124,  0.40618608],
       [-0.84192557,  0.53058695],
       [ 0.58930337, -0.32137599]])
```

[9]:
```
np.unique(dbscan.labels_)
```

[9]:
```
array([-1,  0,  1,  2,  3,  4,  5,  6])
```

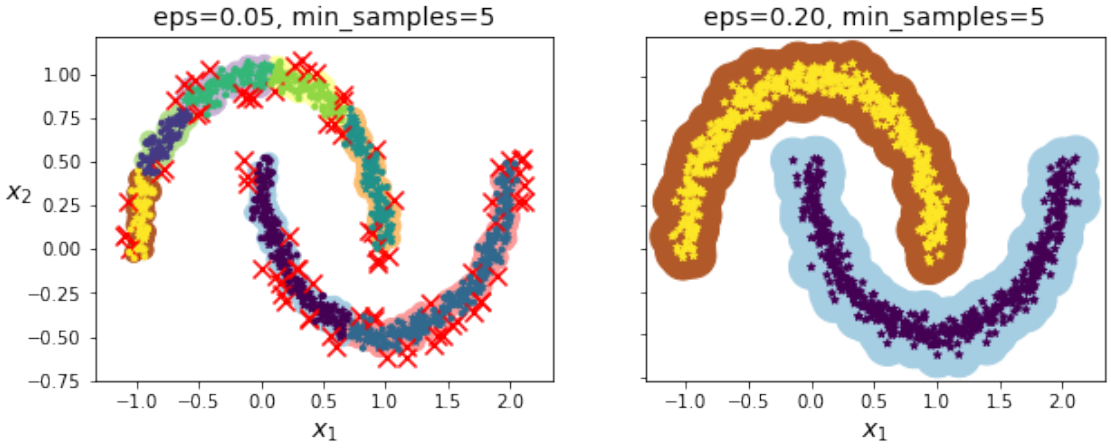
Fitting dataset dengan DBSCAN dengan nilai epsilon (luas jangkauan core adalah 0.2 dan minimal sampe disekitar instance 5

[10]:
```
dbscan2 = DBSCAN(eps=0.2)
dbscan2.fit(X)
```

[10]:
```
DBSCAN(algorithm='auto', eps=0.2, leaf_size=30, metric='euclidean',
       metric_params=None, min_samples=5, n_jobs=None, p=None)
```

[11]:
```
def plot_dbscan(dbscan, X, size, show_xlabels=True, show_ylabels=True):
    core_mask = np.zeros_like(dbscan.labels_, dtype=bool)
    core_mask[dbscan.core_sample_indices_] = True
    anomalies_mask = dbscan.labels_ == -1
    non_core_mask = ~(core_mask | anomalies_mask)
    cores = dbscan.components_
    anomalies = X[anomalies_mask]
    non_cores = X[non_core_mask]
    plt.scatter(cores[:, 0], cores[:, 1],
    c=dbscan.labels_[core_mask], marker='o', s=size, cmap="Paired")
    plt.scatter(cores[:, 0], cores[:, 1], marker='*', s=20, c=dbscan.
 ↪labels_[core_mask])
    plt.scatter(anomalies[:, 0], anomalies[:, 1],
    c="r", marker="x", s=100)
    plt.scatter(non_cores[:, 0], non_cores[:, 1], c=dbscan.
 ↪labels_[non_core_mask], marker=".")
    if show_xlabels:
        plt.xlabel("$x_1$", fontsize=14)
    else:
        plt.tick_params(labelbottom=False)
    if show_ylabels:
        plt.ylabel("$x_2$", fontsize=14, rotation=0)
    else:
        plt.tick_params(labelleft=False)
    plt.title("eps={:.2f}, min_samples={}".format(dbscan.eps, dbscan.
 ↪min_samples), fontsize=14)
```

[12]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 3.5))
plt.subplot(121)
plot_dbscan(dbscan, X, size=100)
plt.subplot(122)
plot_dbscan(dbscan2, X, size=600, show_ylabels=False)
plt.show()
```



Gambar sebelah kiri di atas menunjukan bahwa terdapat banyak anomali ditambah dengan 7 cluster yang berbeda, ketika harga = 0.05. Tetapi kita dapat memperlebar jarak tetangga dari sebuah instance dengan cara merubah menjadi 0.2. Sehingga akan diperoleh gambar sebelah kanan. Selanjutnya kita akan memakai model dengan = 0.2.

[13]:
```
dbscan = dbscan2
```

[14]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[15]:
```
knn = KNeighborsClassifier(n_neighbors=50)
knn.fit(dbscan.components_, dbscan.labels_[dbscan.core_sample_indices_])
```

[15]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=50, p=2,
                     weights='uniform')
```

[16]:
```
X_new = np.array([[-0.5, 0], [0, 0.5], [1, -0.1], [2, 1]])
knn.predict(X_new)
```

[16]:
```
array([1, 0, 1, 0])
```

[17]:
```
 knn.predict_proba(X_new)
```

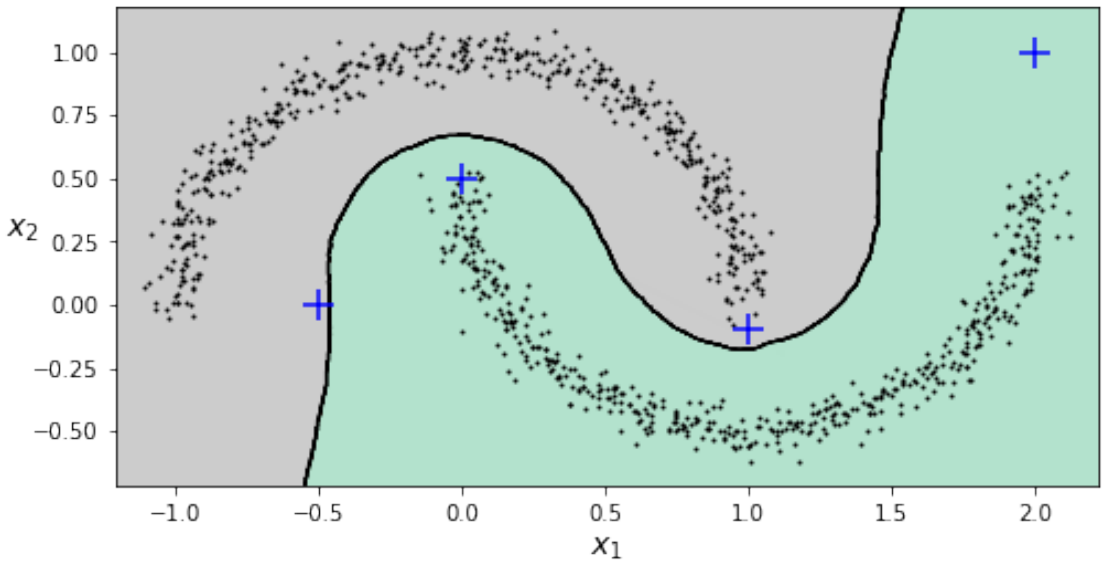
[17]:
```
array([[0.18, 0.82],
       [1.  , 0.  ],
       [0.12, 0.88],
       [1.  , 0.  ]])
```

[18]:
```
def plot_data(X):
    plt.plot(X[:, 0], X[:, 1], 'k.', markersize=2)
```

[19]:
```
def plot_centroids(centroids, weights=None, circle_color='w', cross_color='k'):
    if weights is not None:
        centroids = centroids[weights > weights.max() / 10]
    plt.scatter(centroids[:, 0], centroids[:, 1],
                marker='o', s=30, linewidths=8,
                color=circle_color, zorder=10, alpha=0.9)
    plt.scatter(centroids[:, 0], centroids[:, 1],
                marker='x', s=50, linewidths=50,
                color=cross_color, zorder=11, alpha=1)
```

[20]:
```
def plot_decision_boundaries(clusterer, X, resolution=1000,␣
↪show_centroids=True,show_xlabels=True, show_ylabels=True):
    mins = X.min(axis=0) - 0.1
    maxs = X.max(axis=0) + 0.1
    xx, yy = np.meshgrid(np.linspace(mins[0], maxs[0], resolution),
                         np.linspace(mins[1], maxs[1], resolution))
    Z = clusterer.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]),cmap="Pastel2")
    plt.contour(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]),
            linewidths=1, colors='k')
    plot_data(X)
    if show_centroids:
        plot_centroids(clusterer.cluster_centers_)
    if show_xlabels:
        plt.xlabel("$x_1$", fontsize=14)
    else:
        plt.tick_params(labelbottom=False)
    if show_ylabels:
        plt.ylabel("$x_2$", fontsize=14, rotation=0)
    else:
        plt.tick_params(labelleft=False)
```

[21]:
```
plt.figure(figsize=(8, 4))
plot_decision_boundaries(knn, X, show_centroids=False)
plt.scatter(X_new[:, 0], X_new[:, 1], c="b", marker="+", s=200,zorder=10)
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
x_2
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x_1

[ ]:

[ ]: