DBSCAN

June 14, 2021

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns

from matplotlib import pyplot as plt
from sklearn.cluster import KMeans, DBSCAN
from utils import get_data_train, get_columns
```

[10]:
```
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 tqdm import tqdm
```

[2]:
```
df = get_data_train()
chosen_cols = get_columns(df, n_cols=25) + ['activity', 'subject']
```

[4]:
```
X = df[chosen_cols].drop(['activity', 'subject'], axis=1)
y = df['activity']
```

[60]:
```
epss = [0.15, 0.2,0.25, 0.3,0.35]
min_samples = [30, 35, 40, 45, 50]
n_epss = len( epss)
n_min_samples = len( min_samples)

homogenities = np.ndarray((n_epss, n_min_samples),)
completenesses = np.ndarray((n_epss, n_min_samples),)
v_measures = np.ndarray((n_epss, n_min_samples),)
adjusted_rands = np.ndarray((n_epss, n_min_samples),)
adjusted_mutual_infos = np.ndarray((n_epss, n_min_samples),)
silhuettes = np.ndarray((n_epss, n_min_samples),)
```

[61]:
```
for i in tqdm(range( n_epss)):
    for j in range( n_min_samples):
        db = DBSCAN(eps=epss[i], min_samples=min_samples[j]).fit(X)
        labels = db.labels_
```

```
        homogenities[i,j] = metrics.homogeneity_score(y, labels)
        completenesses[i,j] = metrics.completeness_score(y, labels)
        v_measures[i,j] = metrics.v_measure_score(y, labels)
        adjusted_rands[i,j] = metrics.adjusted_rand_score(y, labels)
        adjusted_mutual_infos[i,j] = metrics.adjusted_mutual_info_score(y,␣
↪labels)
        silhuettes[i,j] = metrics.silhouette_score(X, labels)
```

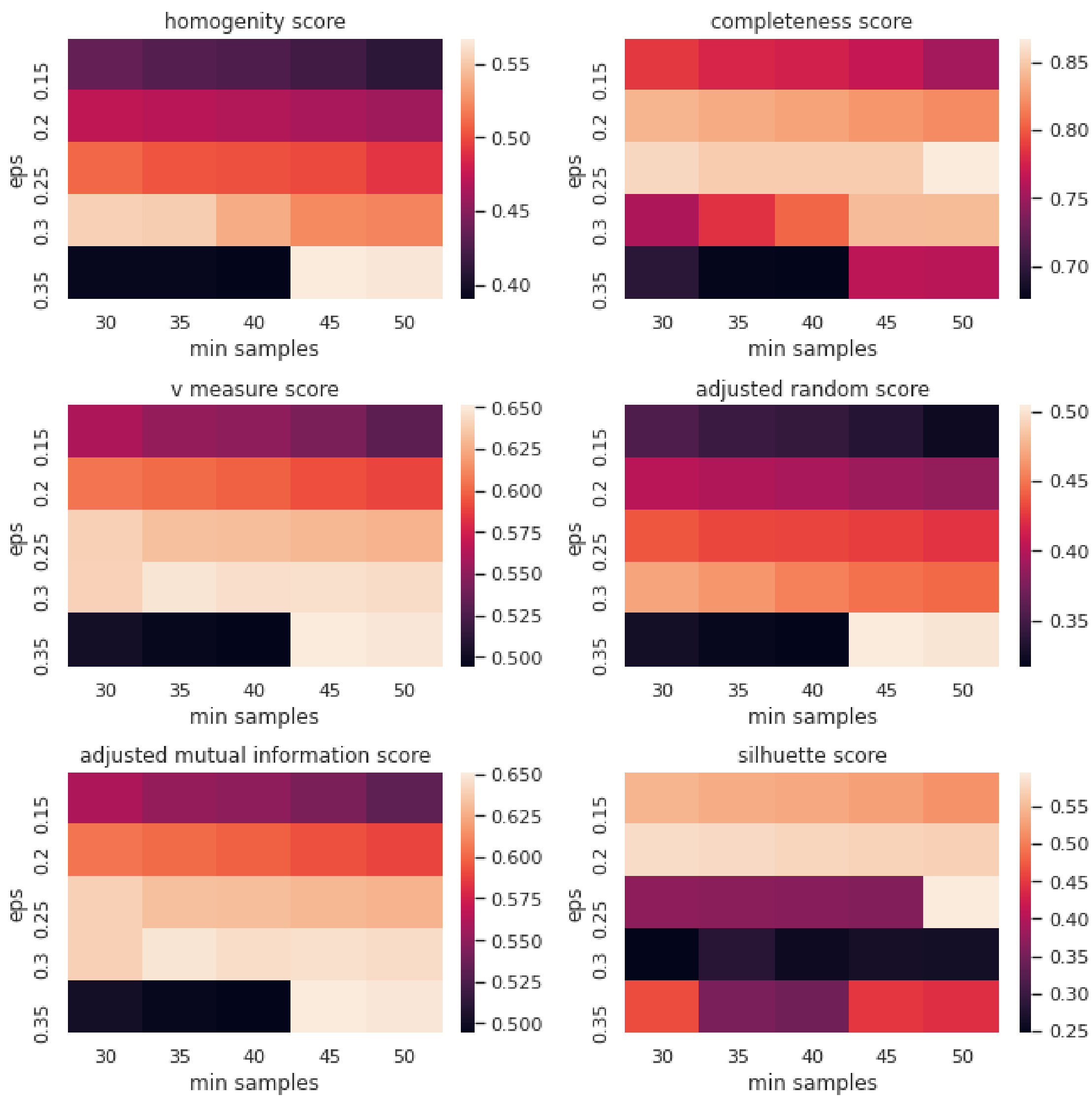
```
100%|      | 5/5 [00:34<00:00,  6.84s/it]
```

[57]:
```
import seaborn as sns; sns.set_theme()
import pandas as pd
```

[62]:
```
homogenities_df = pd.DataFrame( homogenities, columns=min_samples, index=epss )
completenesses_df = pd.DataFrame( completenesses, columns=min_samples,␣
↪index=epss )
v_measures_df = pd.DataFrame( v_measures, columns=min_samples, index=epss )
adjusted_rands_df = pd.DataFrame( adjusted_rands, columns=min_samples,␣
↪index=epss )
adjusted_mutual_infos_df = pd.DataFrame( adjusted_mutual_infos,␣
↪columns=min_samples, index=epss )
silhuettes_df = pd.DataFrame( silhuettes, columns=min_samples, index=epss )
```

[63]:
```
fig, axs = plt.subplots(3, 2,figsize=(9,9), constrained_layout=True)
#fig.tight_layout()
sns.heatmap( ax = axs[0,0], data = homogenities_df).set(title='homogenity␣
↪score', ylabel="eps", xlabel = "min samples")
sns.heatmap( ax = axs[0,1], data = completenesses_df).set(title='completeness␣
↪score', ylabel="eps", xlabel = "min samples")
sns.heatmap( ax = axs[1,0], data = v_measures_df).set(title='v measure score',␣
↪ylabel="eps", xlabel = "min samples")
sns.heatmap( ax = axs[1,1], data = adjusted_rands_df).set(title='adjusted␣
↪random score', ylabel="eps", xlabel = "min samples")
sns.heatmap( ax = axs[2,0], data = adjusted_mutual_infos_df).
↪set(title='adjusted mutual information score', ylabel="eps", xlabel = "min␣
↪samples")
sns.heatmap( ax = axs[2,1], data = silhuettes_df).set(title='silhuette score',␣
↪ylabel="eps", xlabel = "min samples")

plt.show()
```



Powyższe obrazy sugerują, ze parametry $\epsilon = 0.25$ oraz min_samples= 50 dają dobre rezultaty.

[47]:
```
import sklearn
from matplotlib.ticker import MaxNLocator
```

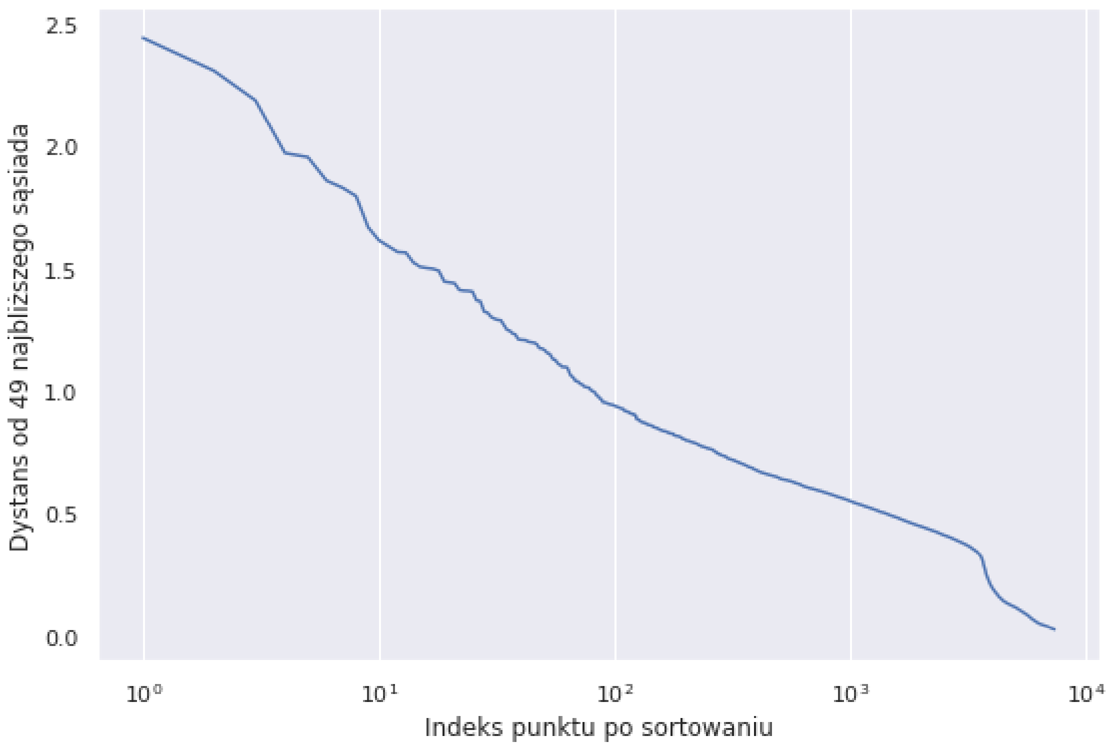
Dobieranie bardziej typowo:

[76]:
```
minPts = 2*25
nbrs = sklearn.neighbors.NearestNeighbors(n_neighbors=minPts).fit( X)
distances, indices = nbrs.kneighbors( X)
distanceDec = sorted(distances[:,minPts-1], reverse=True)
#fig = plt.figure(figsize=(9,6))
#ax1 = fig.add_subplot(111)
fig,axes=plt.subplots(1,1, figsize=(9,6))
```

```
axes.xaxis.set_major_locator(MaxNLocator(10))
plt.xlabel('Indeks punktu po sortowaniu')
plt.ylabel('Dystans od 49 najbliższego sąsiada')
plt.plot(list(range(1, X.shape[0]+1)), distanceDec)

plt.xscale('log')
plt.grid(axis='y')

plt.show()
```



zdaje się, że wartość $\epsilon = 0.25$ jest optymalna

[64]:
```
db = DBSCAN(eps=0.25, min_samples=50).fit(X)
```

[68]:
```
np.unique( db.labels_)
```

[68]:
```
array([-1,  0,  1])
```