Lab4

October 13, 2021

[ ]:
```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics.cluster import contingency_matrix
from sklearn.model_selection import cross_val_predict
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

1 Assignment 1

[ ]:
```
cluster_a = make_blobs(300, n_features=2, centers=4, cluster_std=0.6,␣
↪center_box = (-10.0, 10.0))
cluster_b = make_blobs(300, n_features=2, centers=4, cluster_std=0.1,␣
↪center_box = (-10.0, 10.0))
cluster_c = make_blobs(300, n_features=2, centers=4, cluster_std=2.5,␣
↪center_box = (-10.0, 10.0))
```

[ ]:
```
def get_clustering(cluster, std, random_state=None):
    X, y = cluster
    fig, axs = plt.subplots(1, 11, figsize=(40,40))
    fig.suptitle(f'cluster for standard deviation = {std}, random state =␣
↪{random_state}')

    sse = []
    for i in range(1, 11):
        kmean = KMeans(n_clusters=i, random_state=random_state)
        y_pred = kmean.fit_predict(X,y)
        centroids = kmean.cluster_centers_
        sse.append(kmean.inertia_)
        axs[i-1].scatter(X[:,0], X[:,1], c= kmean.labels_)
        axs[i-1].scatter(centroids[:,0], centroids[:,1], c=np.unique(kmean.
↪labels_), edgecolors='red')
        axs[i-1].set(aspect = 'equal', adjustable = 'box', title=f'nr. of␣
↪clusters: {i}')
        print(f"contingency matrix for cluster with standard deviation = {std},␣
↪random state = {random_state}, k = {i}: ")
```

```
        print(contingency_matrix(y, y_pred))

    axs[10].plot(np.arange(1,11), sse)
    asp = np.diff(axs[10].get_xlim())[0] / np.diff(axs[10].get_ylim())[0]

    axs[10].set(aspect=asp, title=f'SSE for different k')
    axs[10].set_xticks(np.arange(1,11))
    axs[10].yaxis.tick_right()
    fig.tight_layout()
    fig.set_size_inches(17, 3)
```

[ ]:
```
for cluster, std in zip([cluster_a, cluster_b, cluster_c], [0.6, 0.1, 2.5]):

    get_clustering(cluster, std)
```

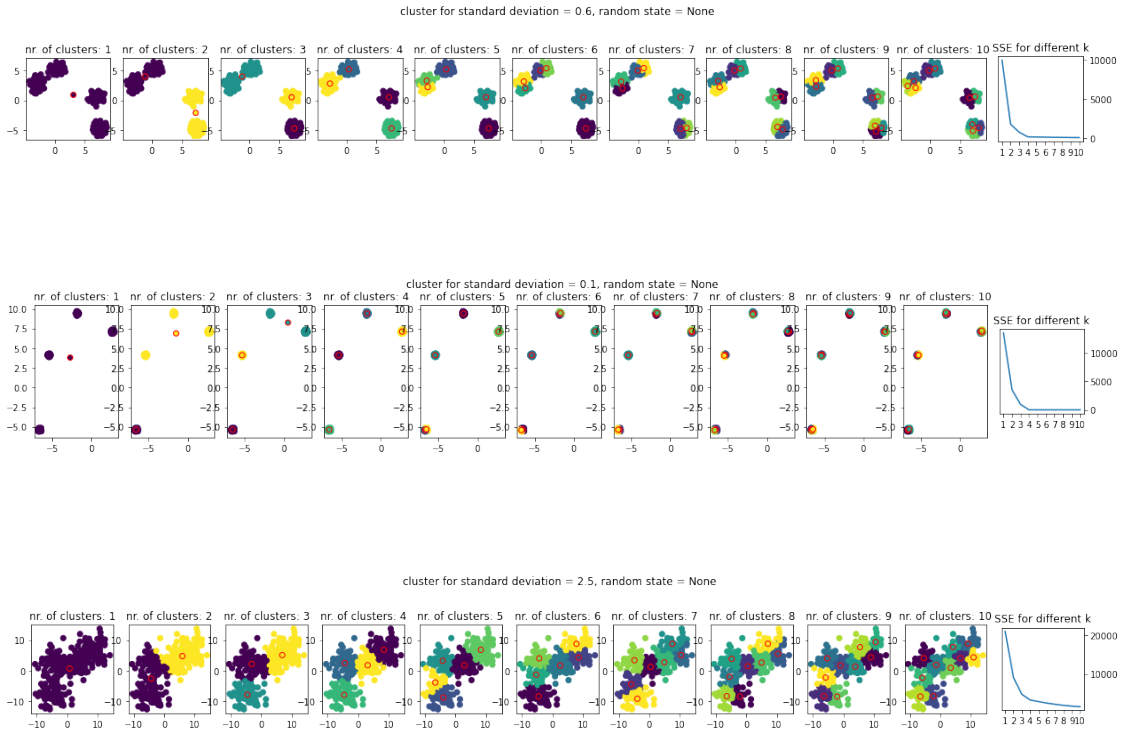
```
contingency matrix for cluster with standard deviation = 0.6, random state =
None, k = 1:
[[75]
 [75]
 [75]
 [75]]
contingency matrix for cluster with standard deviation = 0.6, random state =
None, k = 2:
[[75  0]
 [75  0]
 [ 0 75]
 [ 0 75]]
contingency matrix for cluster with standard deviation = 0.6, random state =
None, k = 3:
[[ 0 75  0]
 [ 0 75  0]
 [ 0  0 75]
 [75  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state =
None, k = 4:
[[ 0 75  0  0]
 [ 0  0  0 75]
 [75  0  0  0]
 [ 0  0 75  0]]
contingency matrix for cluster with standard deviation = 0.6, random state =
None, k = 5:
[[ 0 75  0  0  0]
 [ 0  0  0 42 33]
 [ 0  0 75  0  0]
 [75  0  0  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state =
None, k = 6:
[[ 0 39  0  0 36  0]
```

```
 [ 0  1  0 24  0 50]
 [ 0  0 75  0  0  0]
 [75  0  0  0  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state =
None, k = 7:
[[ 0  0 41  0  0  0 34]
 [23  0  1  0 51  0  0]
 [ 0  0  0 75  0  0  0]
 [ 0 44  0  0  0 31  0]]
contingency matrix for cluster with standard deviation = 0.6, random state =
None, k = 8:
[[ 0 35  0  0 40  0  0  0]
 [ 0  1  0 43  0  0  0 31]
 [35  0  0  0  0 40  0  0]
 [ 0  0 43  0  0  0 32  0]]
contingency matrix for cluster with standard deviation = 0.6, random state =
None, k = 9:
[[ 0 35  0  0  0 40  0  0  0]
 [ 0  1  0 40  0  0  0  0 34]
 [ 0  0 38  0  0  0 37  0  0]
 [24  0  0  0 27  0  0 24  0]]
contingency matrix for cluster with standard deviation = 0.6, random state =
None, k = 10:
[[ 0 35  0  0 40  0  0  0  0  0]
 [ 0  1  0 44  0  0  0  0 11 19]
 [38  0  0  0  0  0 37  0  0  0]
 [ 0  0 28  0  0 16  0 31  0  0]]
contingency matrix for cluster with standard deviation = 0.1, random state =
None, k = 1:
[[75]
 [75]
 [75]
 [75]]
contingency matrix for cluster with standard deviation = 0.1, random state =
None, k = 2:
[[ 0 75]
 [ 0 75]
 [ 0 75]
 [75  0]]
contingency matrix for cluster with standard deviation = 0.1, random state =
None, k = 3:
[[ 0  0 75]
 [ 0 75  0]
 [ 0 75  0]
 [75  0  0]]
contingency matrix for cluster with standard deviation = 0.1, random state =
None, k = 4:
[[75  0  0  0]
```

```
 [ 0  0  0 75]
 [ 0 75  0  0]
 [ 0  0 75  0]]
contingency matrix for cluster with standard deviation = 0.1, random state =
None, k = 5:
[[ 0  0 75  0  0]
 [ 0  0  0 75  0]
 [75  0  0  0  0]
 [ 0 36  0  0 39]]
contingency matrix for cluster with standard deviation = 0.1, random state =
None, k = 6:
[[ 0  0 75  0  0  0]
 [ 0  0  0 75  0  0]
 [ 0 35  0  0 40  0]
 [44  0  0  0  0 31]]
contingency matrix for cluster with standard deviation = 0.1, random state =
None, k = 7:
[[ 0  0  0 75  0  0  0]
 [ 0  0 31  0  0 44  0]
 [38  0  0  0 37  0  0]
 [ 0 33  0  0  0  0 42]]
contingency matrix for cluster with standard deviation = 0.1, random state =
None, k = 8:
[[ 0  0 42  0  0  0  0 33]
 [43  0  0  0 32  0  0  0]
 [ 0  0  0 46  0 29  0  0]
 [ 0 40  0  0  0  0 35  0]]
contingency matrix for cluster with standard deviation = 0.1, random state =
None, k = 9:
[[ 0  0 33  0  0 42  0  0  0]
 [ 0  0  0 32  0  0 43  0  0]
 [28  0  0  0 47  0  0  0  0]
 [ 0 29  0  0  0  0  0 20 26]]
contingency matrix for cluster with standard deviation = 0.1, random state =
None, k = 10:
[[ 0  0 34  0  0  0  0  0  0 41]
 [ 0  0  0 29  0  0  0 21 25  0]
 [ 0 24  0  0 25  0 26  0  0  0]
 [37  0  0  0  0 38  0  0  0  0]]
contingency matrix for cluster with standard deviation = 2.5, random state =
None, k = 1:
[[75]
 [75]
 [75]
 [75]]
contingency matrix for cluster with standard deviation = 2.5, random state =
None, k = 2:
[[75  0]
```

```
 [70  5]
 [13 62]
 [ 0 75]]
contingency matrix for cluster with standard deviation = 2.5, random state =
None, k = 3:
[[ 3 72  0]
 [74  1  0]
 [28  1 46]
 [ 0  0 75]]
contingency matrix for cluster with standard deviation = 2.5, random state =
None, k = 4:
[[ 0  3 72  0]
 [ 0 70  1  4]
 [ 8  7  0 60]
 [68  0  0  7]]
contingency matrix for cluster with standard deviation = 2.5, random state =
None, k = 5:
[[ 0 54  0  0 21]
 [ 4  0 61  0 10]
 [61  0  6  8  0]
 [ 7  0  0 68  0]]
contingency matrix for cluster with standard deviation = 2.5, random state =
None, k = 6:
[[65  0  0 10  0  0]
 [ 0  0  6 23 46  0]
 [ 0  7 59  0  5  4]
 [ 0 33  5  0  0 37]]
contingency matrix for cluster with standard deviation = 2.5, random state =
None, k = 7:
[[ 0 27  0  0  0  0 48]
 [12  8  0  0  0 55  0]
 [30  0  2  4 37  2  0]
 [ 0  0 24 34 17  0  0]]
contingency matrix for cluster with standard deviation = 2.5, random state =
None, k = 8:
[[29  0  0 11  0  0 35  0]
 [ 0 11  0 16 48  0  0  0]
 [ 1 29  2  0  2 37  0  4]
 [ 0  0 28  0  0 16  0 31]]
contingency matrix for cluster with standard deviation = 2.5, random state =
None, k = 9:
[[ 0 36  0  0  0  0 28  0 11]
 [ 0  0 23 38  0  0  0  1 13]
 [ 3  0 19  2 40  1  1  9  0]
 [33  0  0  0  2 14  0 26  0]]
contingency matrix for cluster with standard deviation = 2.5, random state =
None, k = 10:
[[ 0  0 31  0  0  0 11  0 33  0]
```

```
[36  0  0  0 26  1 12  0  0  0]
[ 2  8  1  1 14  8  0 39  0  2]
[ 0 19  0 28  0 11  0  2  0 15]]
```



We can see that the clusters are differently spread out, depending on the standard deviation. It can be stated that the bigger the deviation the bigger the blobs. At standard deviation 2.5 it's actually hard to see the different blops. Consequently we can see how the SSE behaves differently for the different standard deviations. For deviation = 0.1 the sse curve indicates that there are actually 4 different clusters. An observation that is easily backed up when looking at the plots. Deviation = 0.6 behaves not as clearly. I've run the experiment a couple of times and there are blop distributions that are already linked to a very low SSE for k=3. Contrary to this Deviation = 2.5 behaves differently in the sense that 10 clusters perform significantly better than 9 clusters. This indicates that for blobs spread out over a large area we'd need higher number of clusters to minimize the SSE. An insight that is very trivial.

[ ]:
```
for cluster, std in zip([cluster_a, cluster_b, cluster_c], [0.6, 0.1, 2.5]):
    for r in [1,5,10]:
        get_clustering(cluster, std, r)
```

```
contingency matrix for cluster with standard deviation = 0.6, random state = 1,
k = 1:
[[75]
 [75]
 [75]
 [75]]
```

```
contingency matrix for cluster with standard deviation = 0.6, random state = 1,
k = 2:
[[75  0]
 [75  0]
 [ 0 75]
 [ 0 75]]
contingency matrix for cluster with standard deviation = 0.6, random state = 1,
k = 3:
[[75  0  0]
 [75  0  0]
 [ 0  0 75]
 [ 0 75  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 1,
k = 4:
[[75  0  0  0]
 [ 0  0  0 75]
 [ 0  0 75  0]
 [ 0 75  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 1,
k = 5:
[[ 0  0 75  0  0]
 [23  0  0  0 52]
 [ 0  0  0 75  0]
 [ 0 75  0  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 1,
k = 6:
[[ 0 41  0  0 34  0]
 [ 0  1  0 40  0 34]
 [ 0  0 75  0  0  0]
 [75  0  0  0  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 1,
k = 7:
[[ 0  0  0 39  0  0 36]
 [ 0 40  0  1  0 34  0]
 [75  0  0  0  0  0  0]
 [ 0  0 32  0 43  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 1,
k = 8:
[[39  0  0  0  0 36  0  0]
 [ 1  0  0 40  0  0 34  0]
 [ 0  0 40  0  0  0  0 35]
 [ 0 32  0  0 43  0  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 1,
k = 9:
[[ 0 41  0  0  0  0 34  0  0]
 [ 0  1  0 51  0 23  0  0  0]
 [ 0  0 39  0  0  0  0 36  0]
 [28  0  0  0 16  0  0  0 31]]
```

```
contingency matrix for cluster with standard deviation = 0.6, random state = 1,
k = 10:
[[ 0  0  0 36  0  0  0 39  0  0]
 [ 0 19  0  0 36  0 20  0  0  0]
 [ 0  0 39  0  0  0  0  0  0 36]
 [21  0  0  0  0 27  0  0 27  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 5,
k = 1:
[[75]
 [75]
 [75]
 [75]]
contingency matrix for cluster with standard deviation = 0.6, random state = 5,
k = 2:
[[75  0]
 [75  0]
 [ 0 75]
 [ 0 75]]
contingency matrix for cluster with standard deviation = 0.6, random state = 5,
k = 3:
[[75  0  0]
 [75  0  0]
 [ 0 75  0]
 [ 0  0 75]]
contingency matrix for cluster with standard deviation = 0.6, random state = 5,
k = 4:
[[ 0  0  0 75]
 [75  0  0  0]
 [ 0 75  0  0]
 [ 0  0 75  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 5,
k = 5:
[[ 0  0  0 75  0]
 [23  0  0  0 52]
 [ 0 75  0  0  0]
 [ 0  0 75  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 5,
k = 6:
[[40  0  0  0 35  0]
 [ 1  0  0 51  0 23]
 [ 0 75  0  0  0  0]
 [ 0  0 75  0  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 5,
k = 7:
[[ 0 41  0  0  0 34  0]
 [ 0  1  0 40  0  0 34]
 [ 0  0 75  0  0  0  0]
 [43  0  0  0 32  0  0]]
```

```
contingency matrix for cluster with standard deviation = 0.6, random state = 5,
k = 8:
[[39  0  0  0  0  0 36  0]
 [ 1  0  0 51  0 23  0  0]
 [ 0 35  0  0 40  0  0  0]
 [ 0  0 44  0  0  0  0 31]]
contingency matrix for cluster with standard deviation = 0.6, random state = 5,
k = 9:
[[ 0  0  0 40 35  0  0  0  0]
 [30  0  0  0  0  0  0 20 25]
 [ 0  0 39  0  0  0 36  0  0]
 [ 0 31  0  0  0 44  0  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 5,
k = 10:
[[ 0  0  0 40  0  0 35  0  0  0]
 [31  0  0  0 24 20  0  0  0  0]
 [ 0 39  0  0  0  0  0  0  0 36]
 [ 0  0 23  0  0  0  0 25 27  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 10,
k = 1:
[[75]
 [75]
 [75]
 [75]]
contingency matrix for cluster with standard deviation = 0.6, random state = 10,
k = 2:
[[ 0 75]
 [ 0 75]
 [75  0]
 [75  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 10,
k = 3:
[[ 0 75  0]
 [ 0 75  0]
 [75  0  0]
 [ 0  0 75]]
contingency matrix for cluster with standard deviation = 0.6, random state = 10,
k = 4:
[[ 0 75  0  0]
 [ 0  0  0 75]
 [75  0  0  0]
 [ 0  0 75  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 10,
k = 5:
[[ 0  0  0 75  0]
 [ 0 52  0  0 23]
 [75  0  0  0  0]
 [ 0  0 75  0  0]]
```

```
contingency matrix for cluster with standard deviation = 0.6, random state = 10,
k = 6:
[[ 0  0  0 40 35  0]
 [50  0  0  0  1 24]
 [ 0 75  0  0  0  0]
 [ 0  0 75  0  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 10,
k = 7:
[[ 0 40  0  0  0  0 35]
 [ 0  0  0 44  0 30  1]
 [ 0  0 75  0  0  0  0]
 [30  0  0  0 45  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 10,
k = 8:
[[ 0  0  0 41  0  0 34  0]
 [ 0 45  0  1 29  0  0  0]
 [36  0  0  0  0 39  0  0]
 [ 0  0 43  0  0  0  0 32]]
contingency matrix for cluster with standard deviation = 0.6, random state = 10,
k = 9:
[[ 0 39  0  0  0 36  0  0  0]
 [ 0  0  0 33  0  0 20 22  0]
 [38  0  0  0  0  0  0  0 37]
 [ 0  0 44  0 31  0  0  0  0]]
contingency matrix for cluster with standard deviation = 0.6, random state = 10,
k = 10:
[[ 0 39  0  0  0 36  0  0  0  0]
 [ 0  0  0 33  0  0 20 22  0  0]
 [38  0  0  0  0  0  0  0 37  0]
 [ 0  0 31  0 28  0  0  0  0 16]]
contingency matrix for cluster with standard deviation = 0.1, random state = 1,
k = 1:
[[75]
 [75]
 [75]
 [75]]
contingency matrix for cluster with standard deviation = 0.1, random state = 1,
k = 2:
[[75  0]
 [75  0]
 [75  0]
 [ 0 75]]
contingency matrix for cluster with standard deviation = 0.1, random state = 1,
k = 3:
[[ 0  0 75]
 [75  0  0]
 [75  0  0]
 [ 0 75  0]]
```

```
contingency matrix for cluster with standard deviation = 0.1, random state = 1,
k = 4:
[[ 0  0 75  0]
 [75  0  0  0]
 [ 0  0  0 75]
 [ 0 75  0  0]]
contingency matrix for cluster with standard deviation = 0.1, random state = 1,
k = 5:
[[ 0  0 75  0  0]
 [ 0 75  0  0  0]
 [ 0  0  0 75  0]
 [43  0  0  0 32]]
contingency matrix for cluster with standard deviation = 0.1, random state = 1,
k = 6:
[[ 0  0 75  0  0  0]
 [75  0  0  0  0  0]
 [ 0  0  0 32  0 43]
 [ 0 51  0  0 24  0]]
contingency matrix for cluster with standard deviation = 0.1, random state = 1,
k = 7:
[[75  0  0  0  0  0  0]
 [ 0  0 41  0 34  0  0]
 [ 0  0  0 43  0 32  0]
 [ 0 46  0  0  0  0 29]]
contingency matrix for cluster with standard deviation = 0.1, random state = 1,
k = 8:
[[ 0  0 37  0 38  0  0  0]
 [ 0  0  0 52  0  0 23  0]
 [47  0  0  0  0 28  0  0]
 [ 0 44  0  0  0  0  0 31]]
contingency matrix for cluster with standard deviation = 0.1, random state = 1,
k = 9:
[[ 0  0 28  0  0  0  0 47  0]
 [ 0  0  0 41  0  0  0  0 34]
 [ 0 36  0  0  0  0 39  0  0]
 [22  0  0  0 18 35  0  0  0]]
contingency matrix for cluster with standard deviation = 0.1, random state = 1,
k = 10:
[[29  0  0  0 46  0  0  0  0  0]
 [ 0  0 31  0  0  0 22 22  0  0]
 [ 0  0  0 41  0  0  0  0 34  0]
 [ 0 23  0  0  0 33  0  0  0 19]]
contingency matrix for cluster with standard deviation = 0.1, random state = 5,
k = 1:
[[75]
 [75]
 [75]
 [75]]
```

```
contingency matrix for cluster with standard deviation = 0.1, random state = 5,
k = 2:
[[75  0]
 [75  0]
 [75  0]
 [ 0 75]]
contingency matrix for cluster with standard deviation = 0.1, random state = 5,
k = 3:
[[ 0  0 75]
 [75  0  0]
 [75  0  0]
 [ 0 75  0]]
contingency matrix for cluster with standard deviation = 0.1, random state = 5,
k = 4:
[[ 0  0 75  0]
 [ 0  0  0 75]
 [75  0  0  0]
 [ 0 75  0  0]]
contingency matrix for cluster with standard deviation = 0.1, random state = 5,
k = 5:
[[75  0  0  0  0]
 [ 0 75  0  0  0]
 [ 0  0  0 75  0]
 [ 0  0 44  0 31]]
contingency matrix for cluster with standard deviation = 0.1, random state = 5,
k = 6:
[[75  0  0  0  0  0]
 [ 0  0  0 75  0  0]
 [ 0 36  0  0 39  0]
 [ 0  0 36  0  0 39]]
contingency matrix for cluster with standard deviation = 0.1, random state = 5,
k = 7:
[[ 0  0 75  0  0  0  0]
 [43  0  0  0  0 32  0]
 [ 0  0  0 36 39  0  0]
 [ 0 43  0  0  0  0 32]]
contingency matrix for cluster with standard deviation = 0.1, random state = 5,
k = 8:
[[ 0  0 75  0  0  0  0  0]
 [ 0  0  0 26  0 24 25  0]
 [32  0  0  0 43  0  0  0]
 [ 0 43  0  0  0  0  0 32]]
contingency matrix for cluster with standard deviation = 0.1, random state = 5,
k = 9:
[[30  0  0  0 45  0  0  0  0]
 [ 0 31  0  0  0 44  0  0  0]
 [ 0  0  0 37  0  0  0 38  0]
 [ 0  0 20  0  0  0 24  0 31]]
```

```
contingency matrix for cluster with standard deviation = 0.1, random state = 5,
k = 10:
[[ 0  0 39  0  0  0 36  0  0  0]
 [ 0  0  0 44  0  0  0 31  0  0]
 [ 0 23  0  0 26  0  0  0 26  0]
 [21  0  0  0  0 34  0  0  0 20]]
contingency matrix for cluster with standard deviation = 0.1, random state = 10,
k = 1:
[[75]
 [75]
 [75]
 [75]]
contingency matrix for cluster with standard deviation = 0.1, random state = 10,
k = 2:
[[ 0 75]
 [ 0 75]
 [ 0 75]
 [75  0]]
contingency matrix for cluster with standard deviation = 0.1, random state = 10,
k = 3:
[[ 0  0 75]
 [ 0 75  0]
 [ 0 75  0]
 [75  0  0]]
contingency matrix for cluster with standard deviation = 0.1, random state = 10,
k = 4:
[[ 0  0 75  0]
 [ 0  0  0 75]
 [ 0 75  0  0]
 [75  0  0  0]]
contingency matrix for cluster with standard deviation = 0.1, random state = 10,
k = 5:
[[ 0  0 75  0  0]
 [75  0  0  0  0]
 [ 0  0  0 75  0]
 [ 0 34  0  0 41]]
contingency matrix for cluster with standard deviation = 0.1, random state = 10,
k = 6:
[[75  0  0  0  0  0]
 [ 0  0 75  0  0  0]
 [ 0 32  0  0 43  0]
 [ 0  0  0 46  0 29]]
contingency matrix for cluster with standard deviation = 0.1, random state = 10,
k = 7:
[[ 0  0 75  0  0  0  0]
 [ 0  0  0 25 50  0  0]
 [ 0 34  0  0  0  0 41]
 [31  0  0  0  0 44  0]]
```

```
contingency matrix for cluster with standard deviation = 0.1, random state = 10,
k = 8:
[[ 0  0 75  0  0  0  0  0]
 [21  0  0  0  0 28  0 26]
 [ 0  0  0 29 46  0  0  0]
 [ 0 47  0  0  0  0 28  0]]
contingency matrix for cluster with standard deviation = 0.1, random state = 10,
k = 9:
[[ 0  0 75  0  0  0  0  0  0]
 [20  0  0  0 27  0  0  0 28]
 [ 0  0  0 38  0  0 37  0  0]
 [ 0 23  0  0  0 33  0 19  0]]
contingency matrix for cluster with standard deviation = 0.1, random state = 10,
k = 10:
[[75  0  0  0  0  0  0  0  0  0]
 [ 0  0 25  0 24 26  0  0  0  0]
 [ 0  0  0 28  0  0 22  0 25  0]
 [ 0 26  0  0  0  0  0 22  0 27]]
contingency matrix for cluster with standard deviation = 2.5, random state = 1,
k = 1:
[[75]
 [75]
 [75]
 [75]]
contingency matrix for cluster with standard deviation = 2.5, random state = 1,
k = 2:
[[ 0 75]
 [ 5 70]
 [63 12]
 [75  0]]
contingency matrix for cluster with standard deviation = 2.5, random state = 1,
k = 3:
[[ 3 72  0]
 [74  1  0]
 [28  1 46]
 [ 0  0 75]]
contingency matrix for cluster with standard deviation = 2.5, random state = 1,
k = 4:
[[ 0  3 72  0]
 [ 0 70  1  4]
 [ 8  7  0 60]
 [68  0  0  7]]
contingency matrix for cluster with standard deviation = 2.5, random state = 1,
k = 5:
[[ 0 63  0 12  0]
 [ 4  0 51 20  0]
 [61  0  6  0  8]
 [ 7  0  0  0 68]]
```

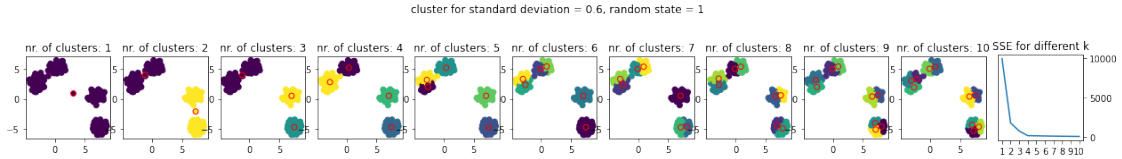
```
contingency matrix for cluster with standard deviation = 2.5, random state = 1,
k = 6:
[[ 0  0 54  0  0 21]
 [ 6 58  0  0  0 11]
 [59  5  0  7  4  0]
 [ 4  0  0 32 39  0]]
contingency matrix for cluster with standard deviation = 2.5, random state = 1,
k = 7:
[[25  0  0  0 50  0  0]
 [ 9  0 26  0  0 40  0]
 [ 0  4 18 47  0  2  4]
 [ 0 37  0  6  0  0 32]]
contingency matrix for cluster with standard deviation = 2.5, random state = 1,
k = 8:
[[ 0 27  0  0 12  0 36  0]
 [ 0  0 44  0 14  1  0 16]
 [38  1  2  3  0  5  0 26]
 [16  0  0 40  0 19  0  0]]
contingency matrix for cluster with standard deviation = 2.5, random state = 1,
k = 9:
[[ 0  0 36  0 12  0  0 27  0]
 [35  0  0  0 12  0 28  0  0]
 [ 2 13  0 42  0  1 14  1  2]
 [ 0 21  0  2  0 33  0  0 19]]
contingency matrix for cluster with standard deviation = 2.5, random state = 1,
k = 10:
[[ 0 12  0  0  0 29 34  0  0  0]
 [ 0 12 42  0  0  0  0 21  0  0]
 [17  0  2  5  3  1  0 19 28  0]
 [ 5  0  0 30 25  0  0  0  1 14]]
contingency matrix for cluster with standard deviation = 2.5, random state = 5,
k = 1:
[[75]
 [75]
 [75]
 [75]]
contingency matrix for cluster with standard deviation = 2.5, random state = 5,
k = 2:
[[75  0]
 [70  5]
 [13 62]
 [ 0 75]]
contingency matrix for cluster with standard deviation = 2.5, random state = 5,
k = 3:
[[ 3  0 72]
 [74  0  1]
 [28 46  1]
 [ 0 75  0]]
```

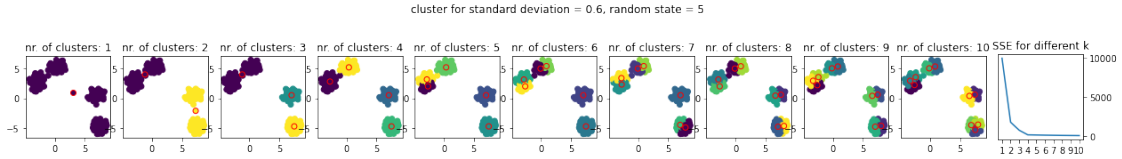
```
contingency matrix for cluster with standard deviation = 2.5, random state = 5,
k = 4:
[[ 3  0 72  0]
 [70  0  1  4]
 [ 7  8  0 60]
 [ 0 68  0  7]]
contingency matrix for cluster with standard deviation = 2.5, random state = 5,
k = 5:
[[63  0  0  0 12]
 [ 0 51  0  4 20]
 [ 0  6  8 61  0]
 [ 0  0 68  7  0]]
contingency matrix for cluster with standard deviation = 2.5, random state = 5,
k = 6:
[[ 0  0 63  0 12  0]
 [52  0  0  5 18  0]
 [ 6  4  0 60  0  5]
 [ 0 37  0  5  0 33]]
contingency matrix for cluster with standard deviation = 2.5, random state = 5,
k = 7:
[[ 0 27  0  0 48  0  0]
 [42  8  0 25  0  0  0]
 [ 2  0 11 20  0 40  2]
 [ 0  0 36  0  0  7 32]]
contingency matrix for cluster with standard deviation = 2.5, random state = 5,
k = 8:
[[ 7  0 40  0  0  0 28  0]
 [22  0  0  0 39  0  0 14]
 [ 0  4  0 40  2  4  1 24]
 [ 0 37  0  5  0 32  0  1]]
contingency matrix for cluster with standard deviation = 2.5, random state = 5,
k = 9:
[[ 0 11  0 31  0  0  0 33  0]
 [ 0 18  0  0 15  0 42  0  0]
 [36  0  1  1 22 11  2  0  2]
 [ 4  0 24  0  0 28  0  0 19]]
contingency matrix for cluster with standard deviation = 2.5, random state = 5,
k = 10:
[[ 0 11  0 33  0  0  0 31  0  0]
 [ 0 18  0  0 15  0 42  0  0  0]
 [30  0  0  0 19 18  2  0  2  4]
 [ 1  0 11  0  0 19  0  0 18 26]]
contingency matrix for cluster with standard deviation = 2.5, random state = 10,
k = 1:
[[75]
 [75]
 [75]
 [75]]
```

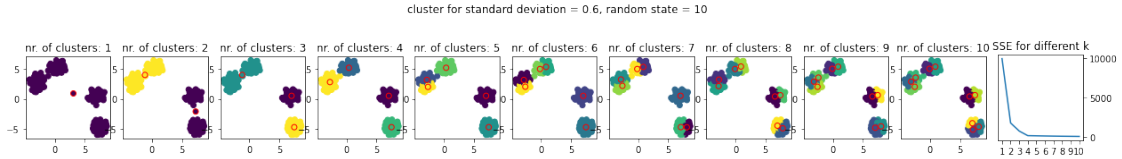
```
contingency matrix for cluster with standard deviation = 2.5, random state = 10,
k = 2:
[[ 0 75]
 [ 5 70]
 [63 12]
 [75  0]]
contingency matrix for cluster with standard deviation = 2.5, random state = 10,
k = 3:
[[ 3  0 72]
 [74  0  1]
 [28 46  1]
 [ 0 75  0]]
contingency matrix for cluster with standard deviation = 2.5, random state = 10,
k = 4:
[[ 3  0 72  0]
 [70  0  1  4]
 [ 7  8  0 60]
 [ 0 68  0  7]]
contingency matrix for cluster with standard deviation = 2.5, random state = 10,
k = 5:
[[26  0  0 49  0]
 [ 9  0  4  0 62]
 [ 0  8 61  0  6]
 [ 0 68  7  0  0]]
contingency matrix for cluster with standard deviation = 2.5, random state = 10,
k = 6:
[[ 0 63  0 12  0  0]
 [ 0  0  5 20  0 50]
 [ 4  0 59  0  6  6]
 [37  0  5  0 33  0]]
contingency matrix for cluster with standard deviation = 2.5, random state = 10,
k = 7:
[[52  0  0  0  0 23  0]
 [ 0 48  0  0  0  9 18]
 [ 0  2  2 39  5  0 27]
 [ 0  0 25 16 34  0  0]]
contingency matrix for cluster with standard deviation = 2.5, random state = 10,
k = 8:
[[ 0 11  0 29  0  0 35  0]
 [ 0 16 14  0 45  0  0  0]
 [ 4  0 27  1  2 39  0  2]
 [34  0  1  0  0 16  0 24]]
contingency matrix for cluster with standard deviation = 2.5, random state = 10,
k = 9:
[[ 0  0  0 37  0  0 11  0 27]
 [44  0  0  0  0  0 16 15  0]
 [ 2 13 34  0  2  1  0 22  1]
 [ 0 22  3  0 23 27  0  0  0]]
```

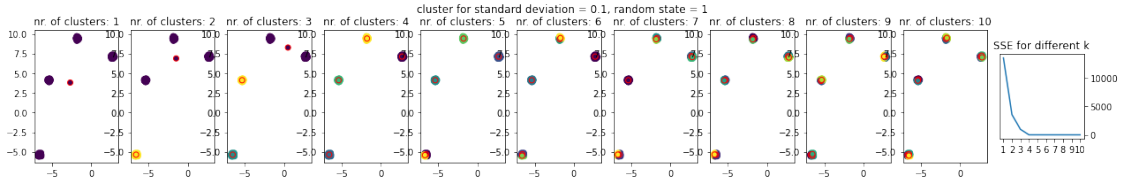
contingency matrix for cluster with standard deviation = 2.5, random state = 10, k = 10:

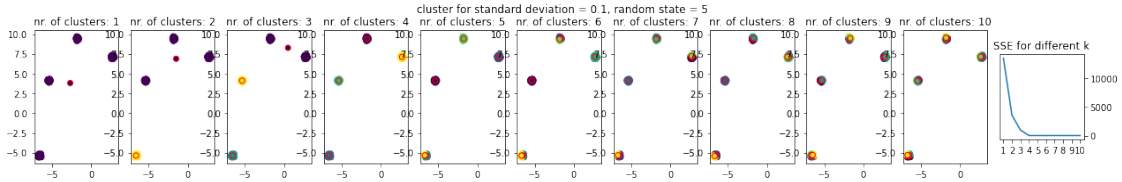
```
[[ 0  0 11  0 29  0  0 35  0  0]
 [ 0 27 12  0  0 36  0  0  0  0]
 [ 2 14  0  3  0  2 35  0 19  0]
 [22  0  0 25  0  0  1  0 16 11]]
```

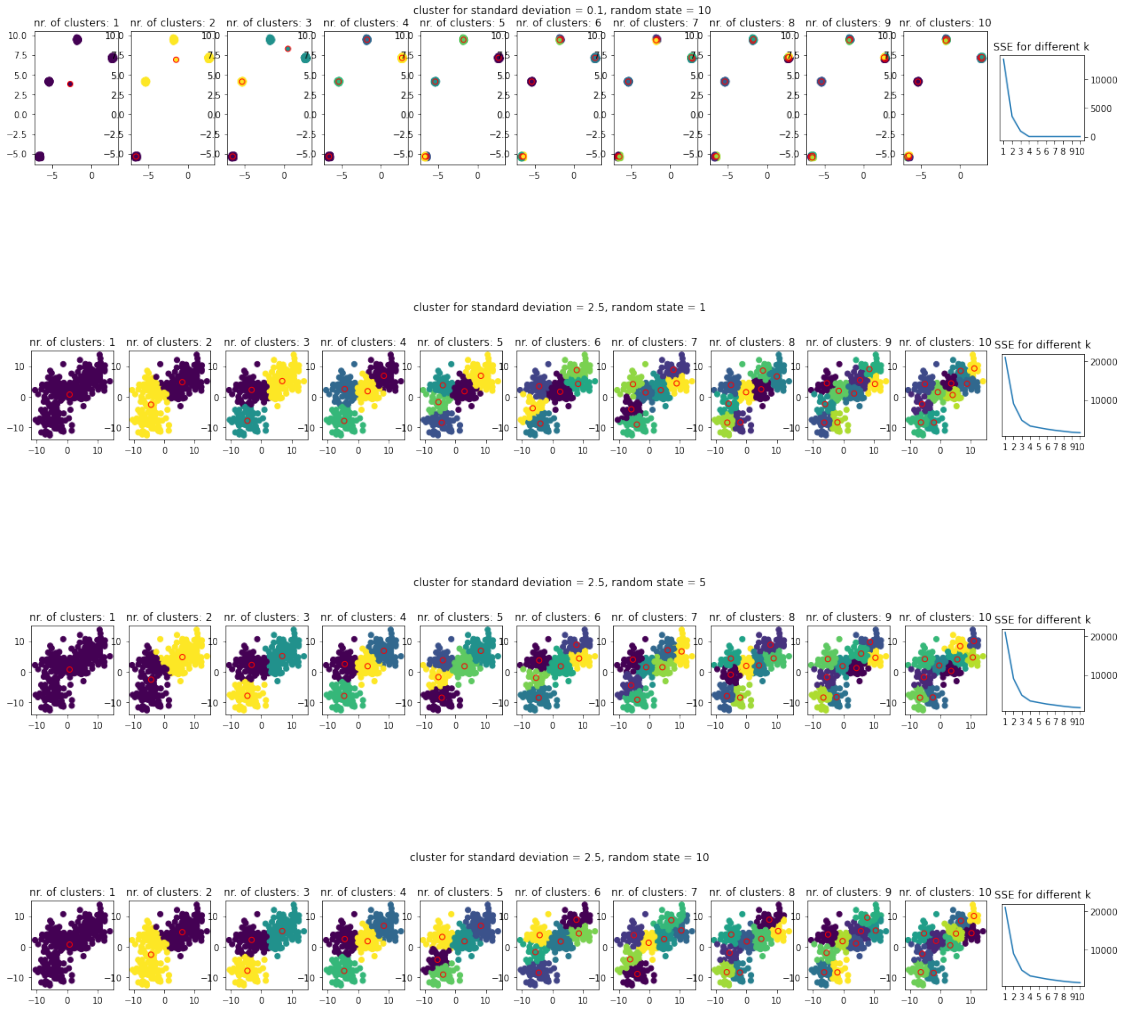












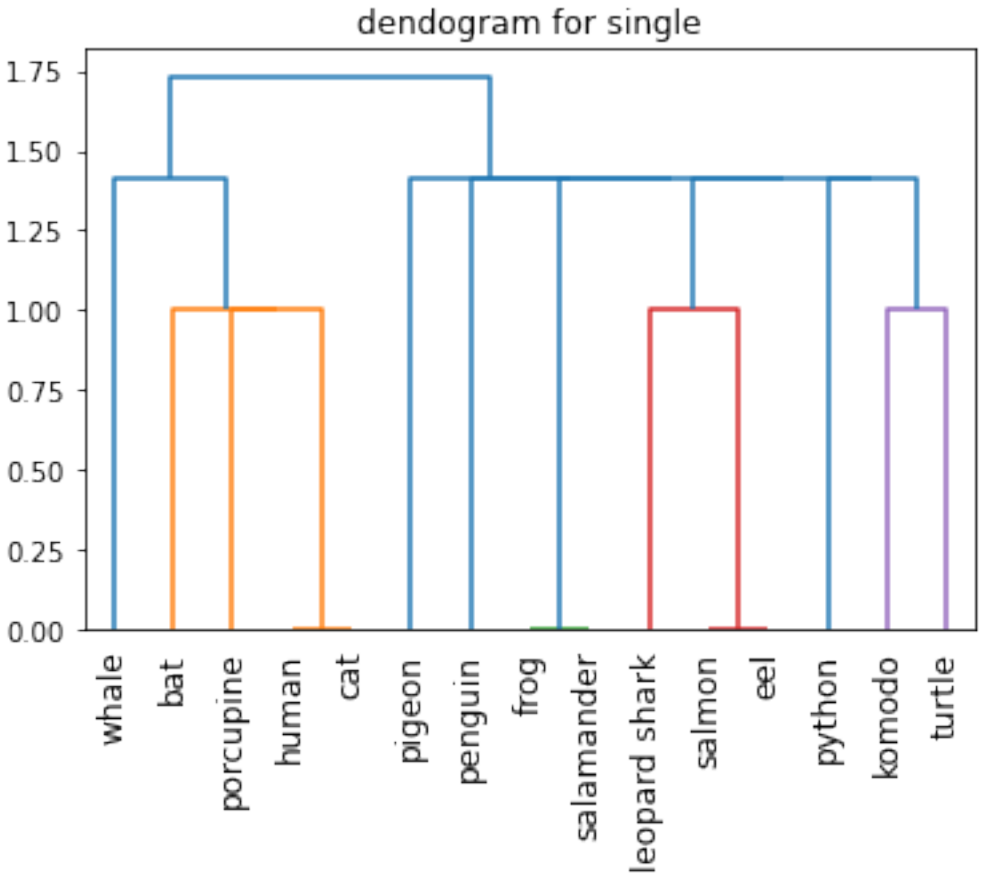
The implementation of k-means relies on randomness in order to find the first set of centroids. Accordingly each different random state produces a different result. However, these differences are only appearent after the actual number of clusters is reached. Meaning that the algorithm produces similar results for k <= 4. with k > 4 the solutions start to differgate. It's not supprising that this is the case, given that the data is distributed into 4 blobs. However these changes do not influence the SSE, which seems to be constant for the different random states. One possible way to chane that would be to have the algorithm start at k fixed locations in the grid provided by the min-max values on all axes. This would ensure that the results stay the same.

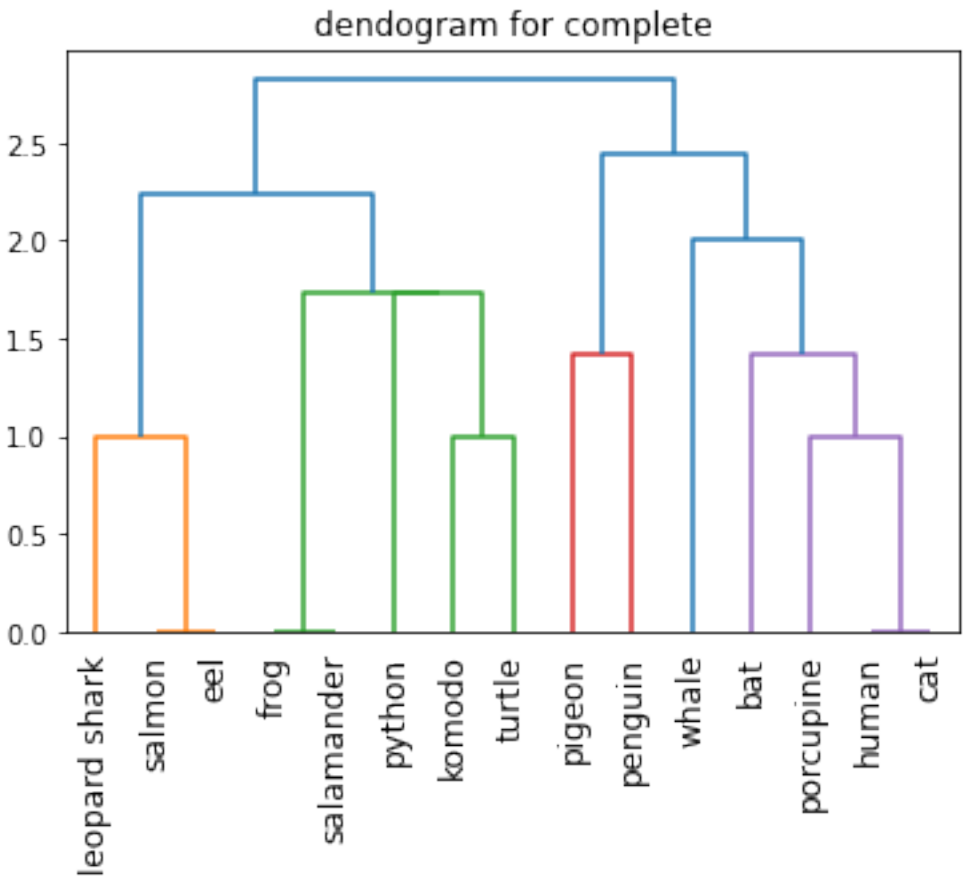
2 Assignment 2

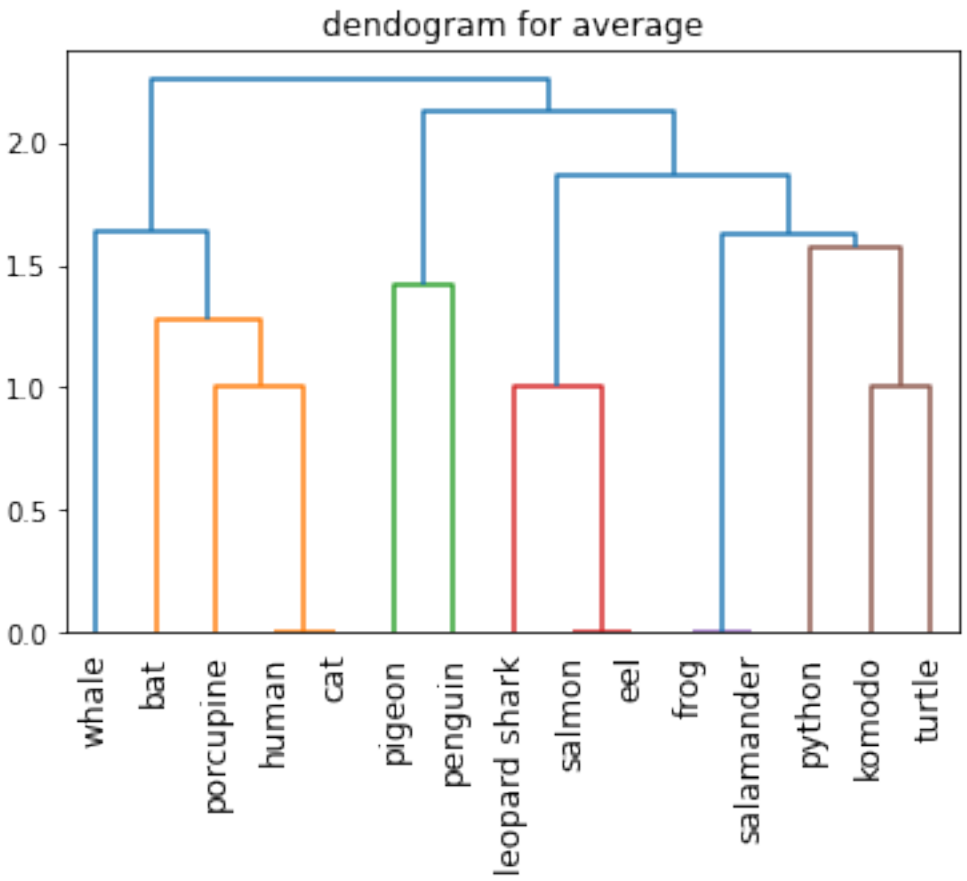
[ ]:
```
data = pd.read_csv('vertebrate.csv')
data = data.convert_dtypes()
data['Class'] = data['Class'].astype(object)
data = pd.get_dummies(data)
```

[ ]:
```
from scipy.cluster.hierarchy import dendrogram, single, complete, average
from scipy.spatial.distance import pdist
```

[ ]:
```
distances = pdist(data.iloc[:,1:-1].astype(int).values)
for i in [single, complete, average]:
    Z = i(distances)
    dendrogram(Z, labels= data.iloc[:,0].values)
    plt.xticks(rotation=90)
    plt.title(f'dendogram for {i.__name__}')
    plt.show()
```




dendogram for complete
2.5
2.0
1.5
1.0
0.5
0.0
leopard shark
salmon
eel
frog
salamander
python
komodo
turtle
pigeon
penguin
whale
bat
porcupine
human
cat



I'd argue that the representation provided by 'average' is the best, simply because the distinction between mammals and non-mammals is the most natural to me. This difference is also given in the 'single' dendogram. However, due to the fact that the rest is clustered into one major subclass is somewhat off in this solution. Comparing 'complete' and 'average' I'd still argue for 'average', simply because birds are closer to reptiles (given that both produce egs) than to mammals. Generally speaking it's obvious that the best (most natural) solution depends highly on the metric used to evaluate this.

3 Assignment 3

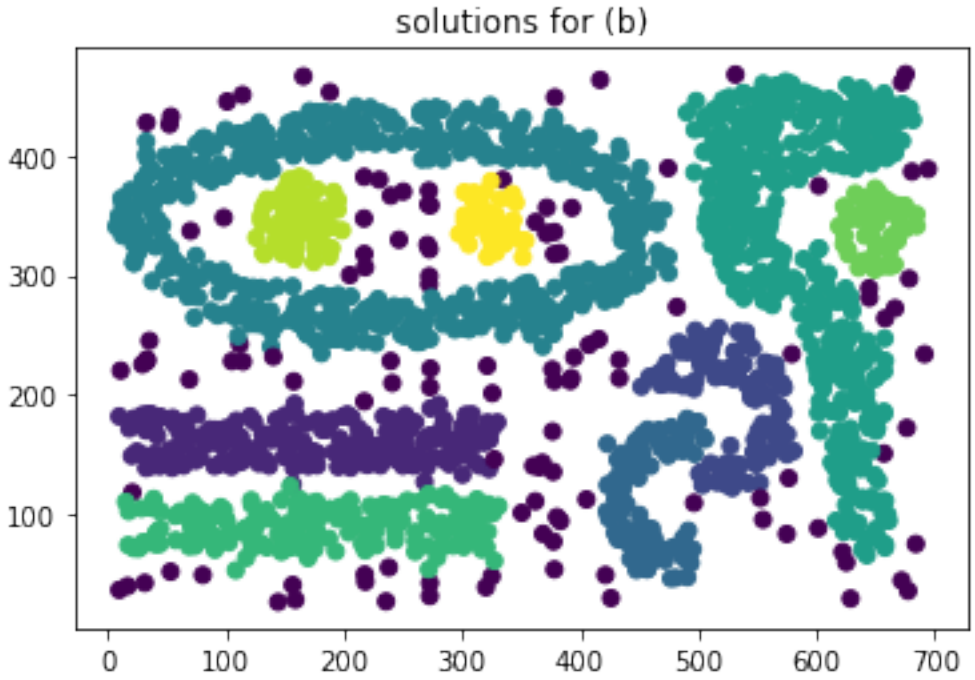
[ ]:
```
from sklearn.cluster import DBSCAN
```

[ ]:
```
data = pd.read_csv('chameleon.csv')
```

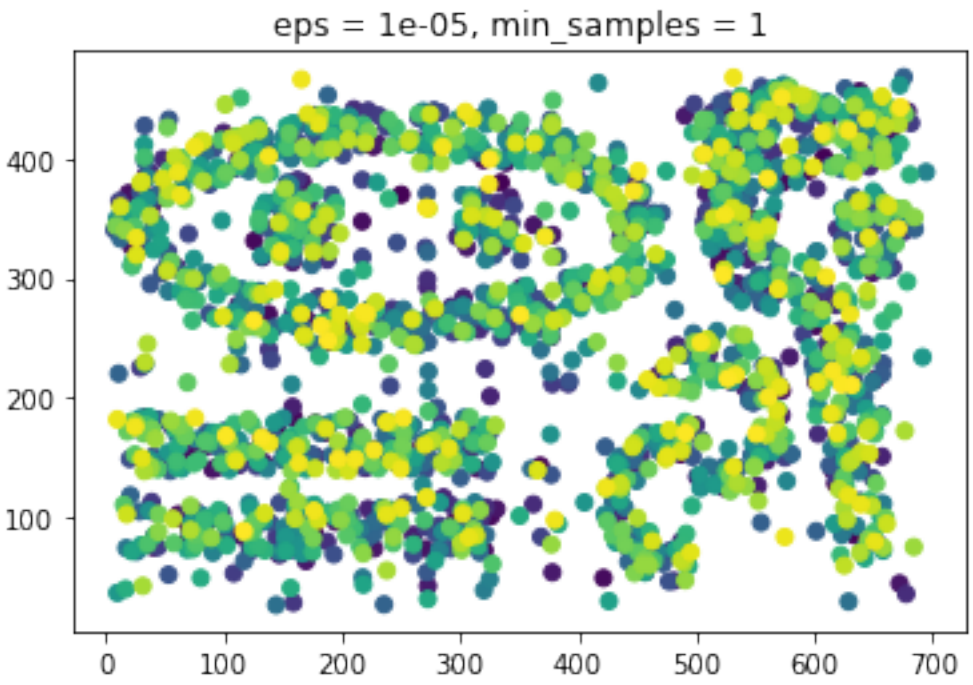
[ ]:
```
dbscan = DBSCAN(eps=15.5, min_samples=5)
class_pred = dbscan.fit_predict(data)
data['labels'] = class_pred
```

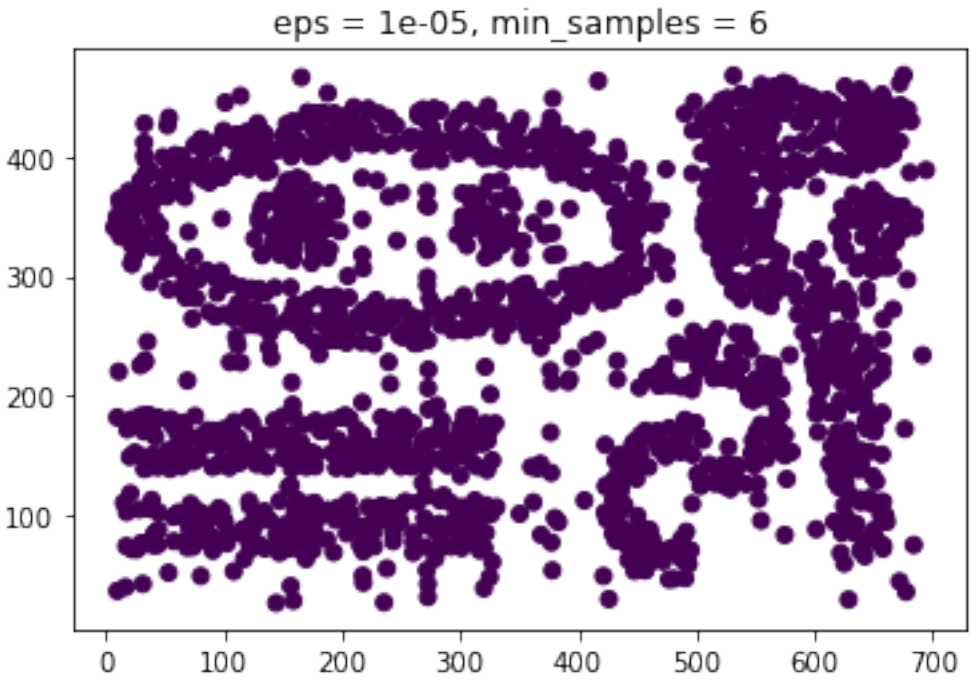
[ ]:
```
plt.scatter(data['x'], data['y'], c=data['labels'])
plt.title('solutions for (b)')
```

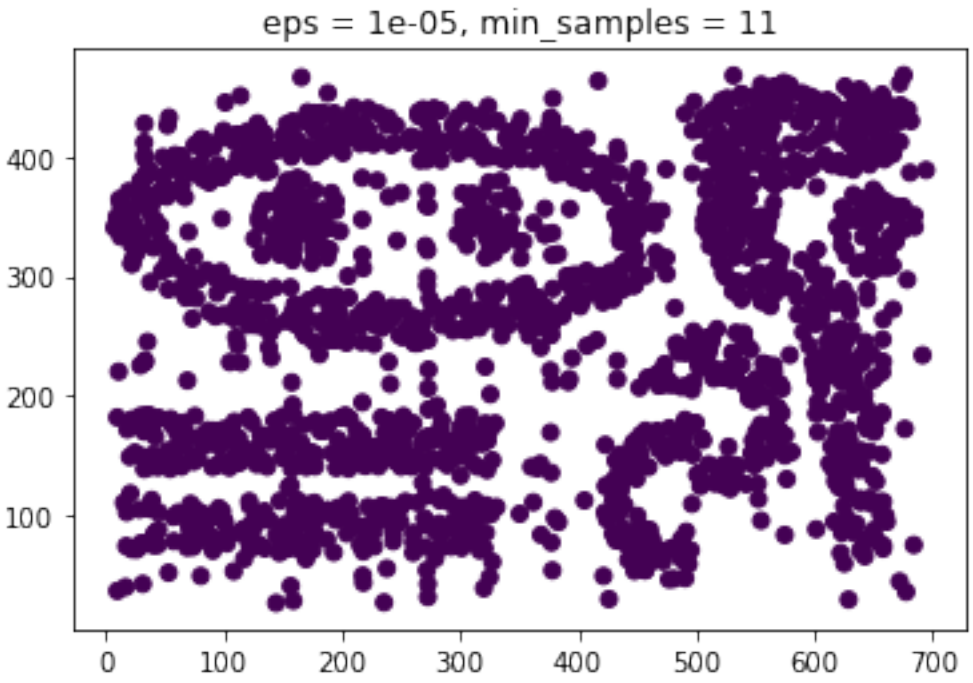
[ ]:
```
Text(0.5, 1.0, 'solutions for (b)')
```

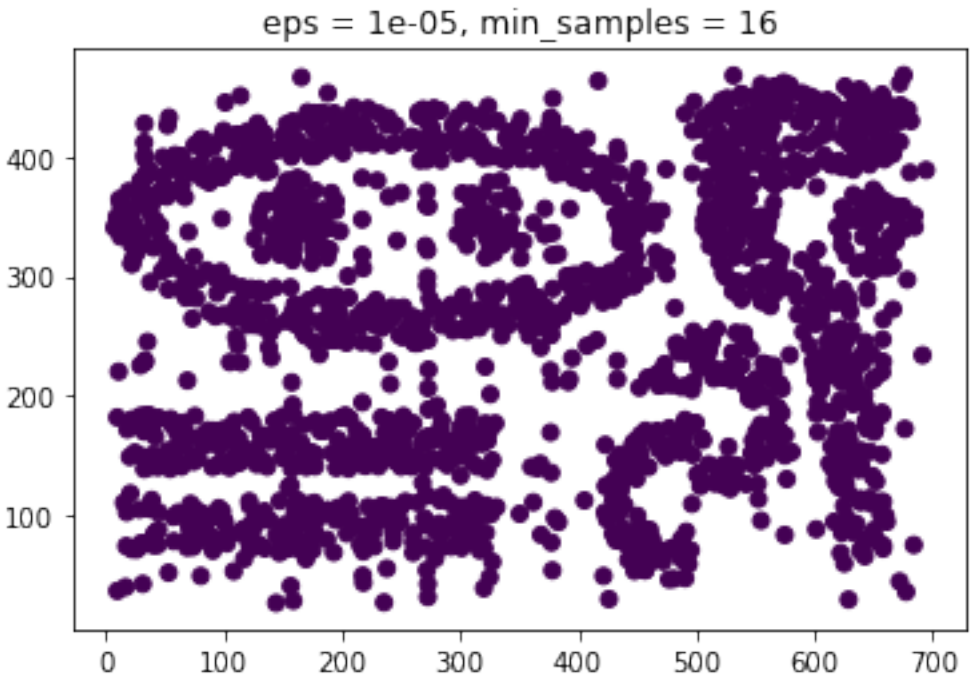


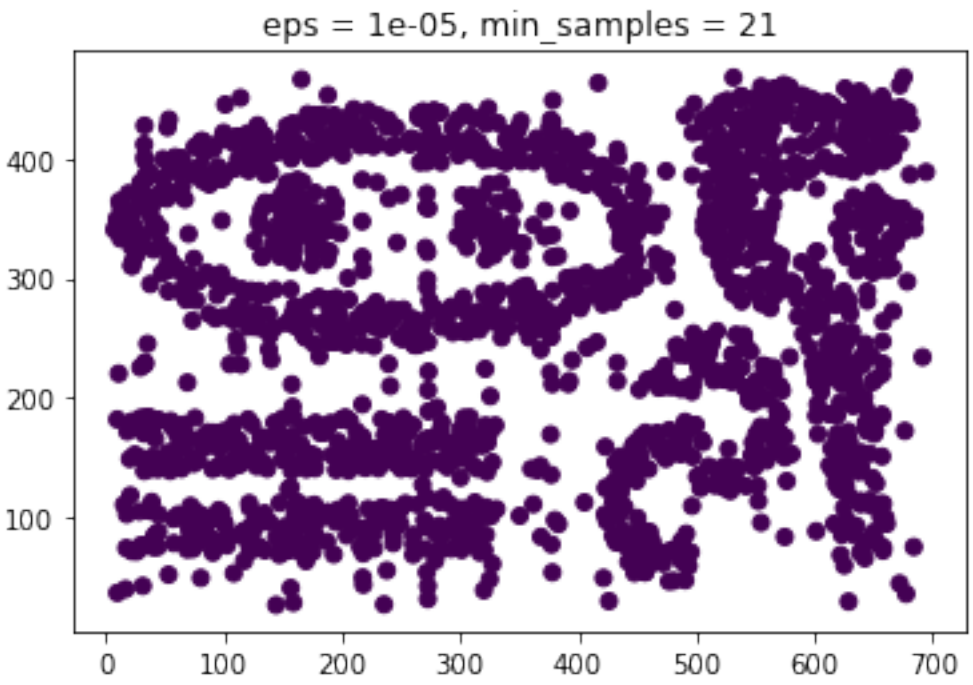
[ ]:
```
for eps in np.hstack((1e-5,np.arange(5,21,5))):
    for min_sample in range(1,22,5):
        dbscan = DBSCAN(eps=eps, min_samples=min_sample)
        class_pred = dbscan.fit_predict(data)
        data['labels'] = class_pred
        plt.scatter(data['x'], data['y'], c=data['labels'])
        plt.title(f'eps = {eps}, min_samples = {min_sample}')
        plt.show()
```

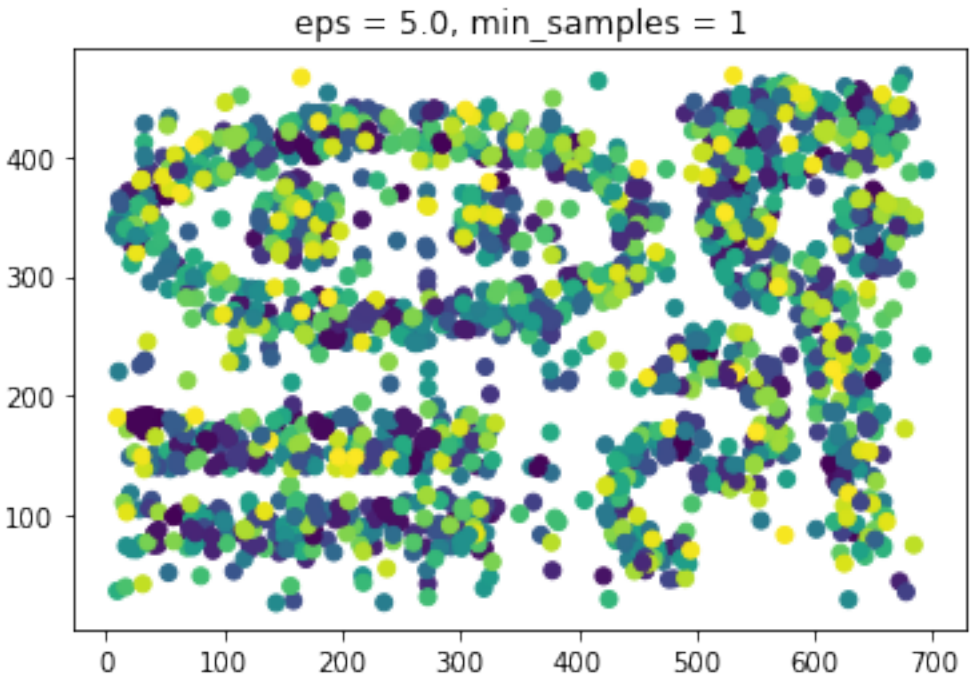

eps = 1e-05, min_samples = 1
400
300
200
100
0
100
200
300
400
500
600
700

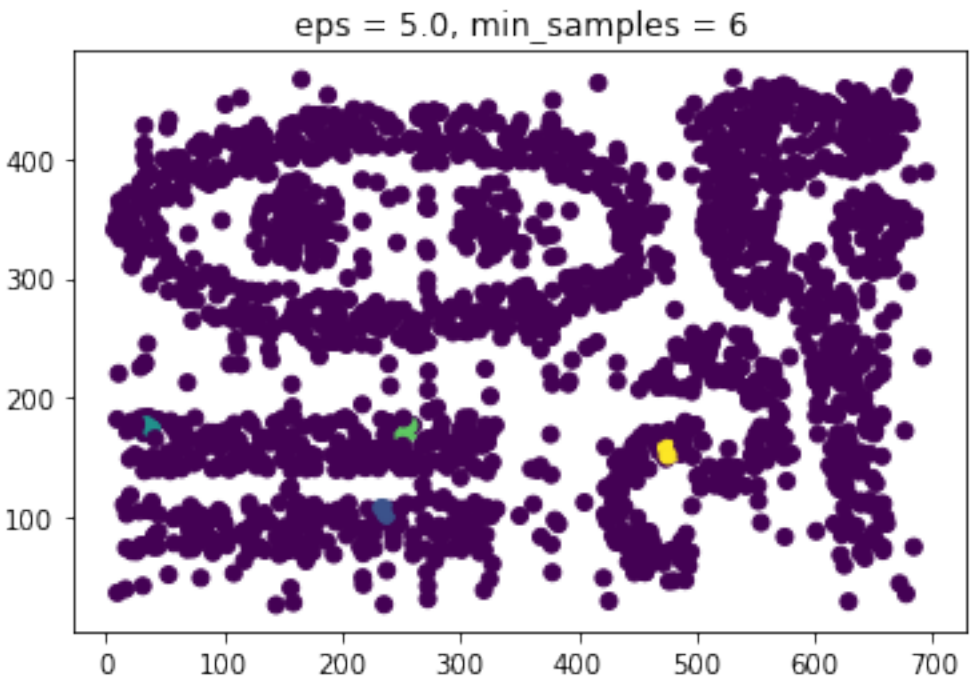

eps = 1e-05, min_samples = 6
400
300
200
100
0
100
200
300
400
500
600
700

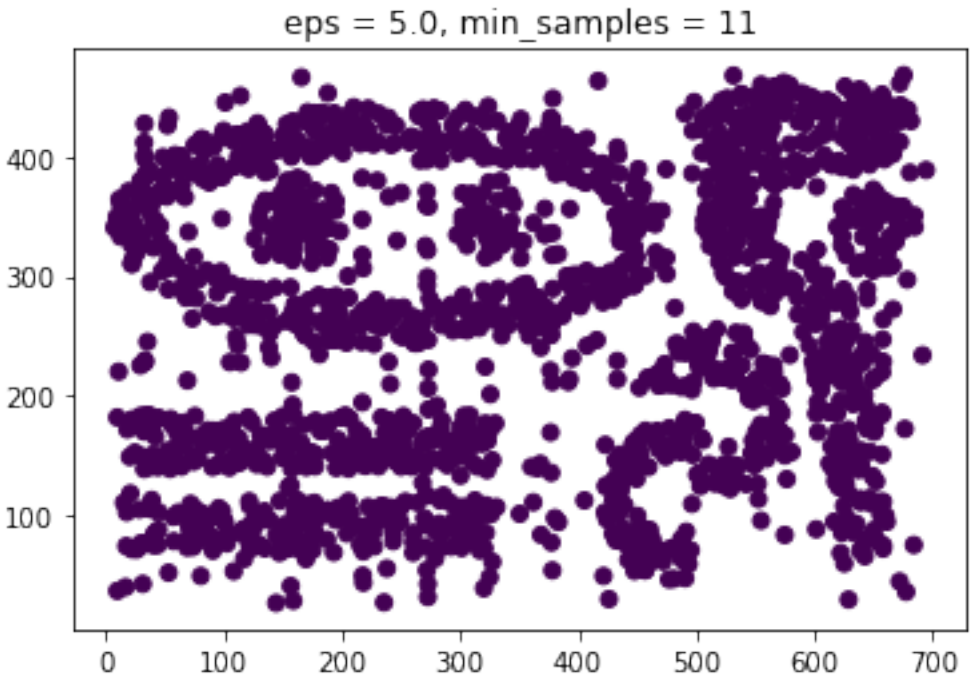

eps = 1e-05, min_samples = 11
400
300
200
100
0
100
200
300
400
500
600
700

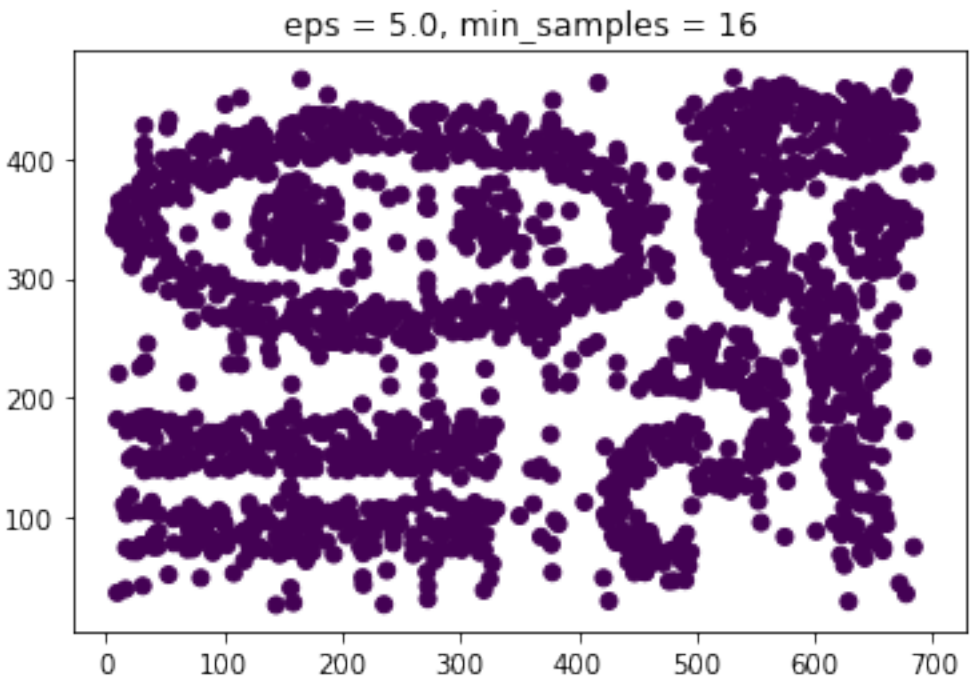

eps = 1e-05, min_samples = 16
400
300
200
100
0
100
200
300
400
500
600
700

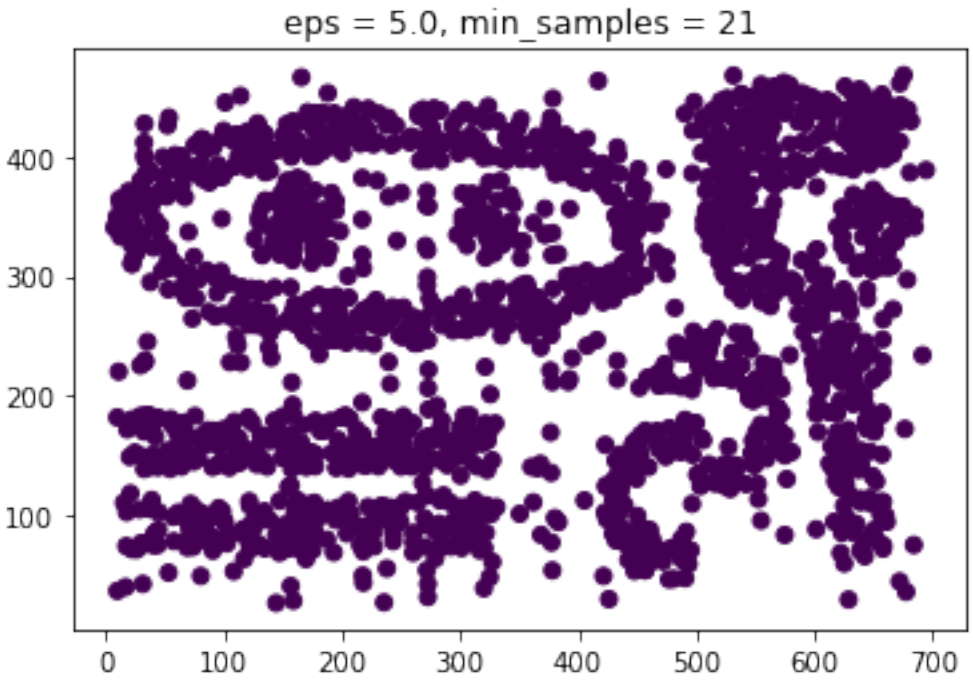

eps = 1e-05, min_samples = 21
400
300
200
100
0
100
200
300
400
500
600
700

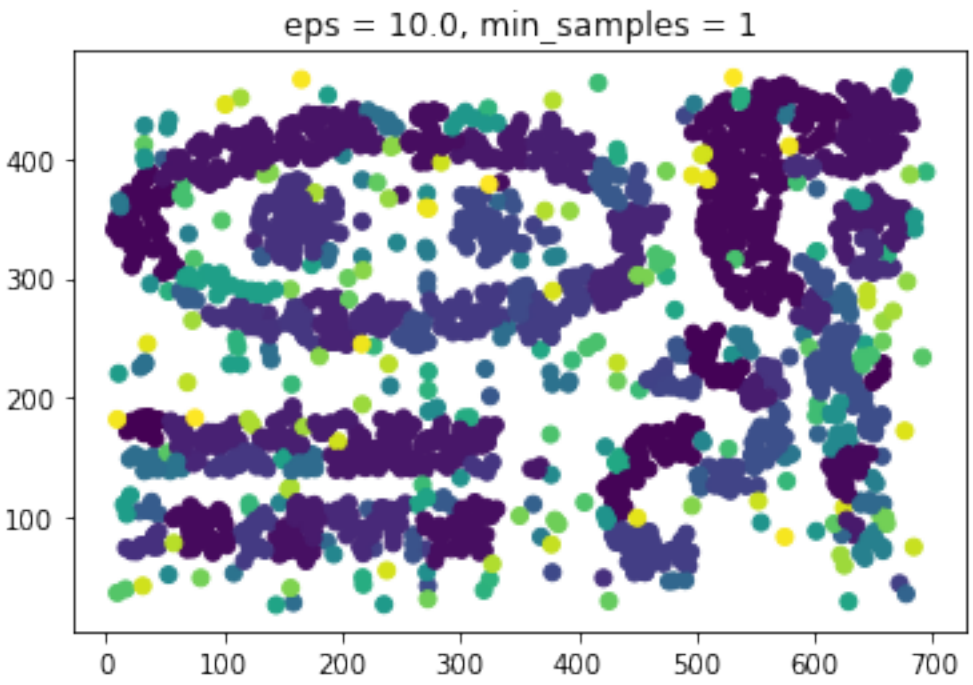

eps = 5.0, min_samples = 1
400
300
200
100
0
100
200
300
400
500
600
700

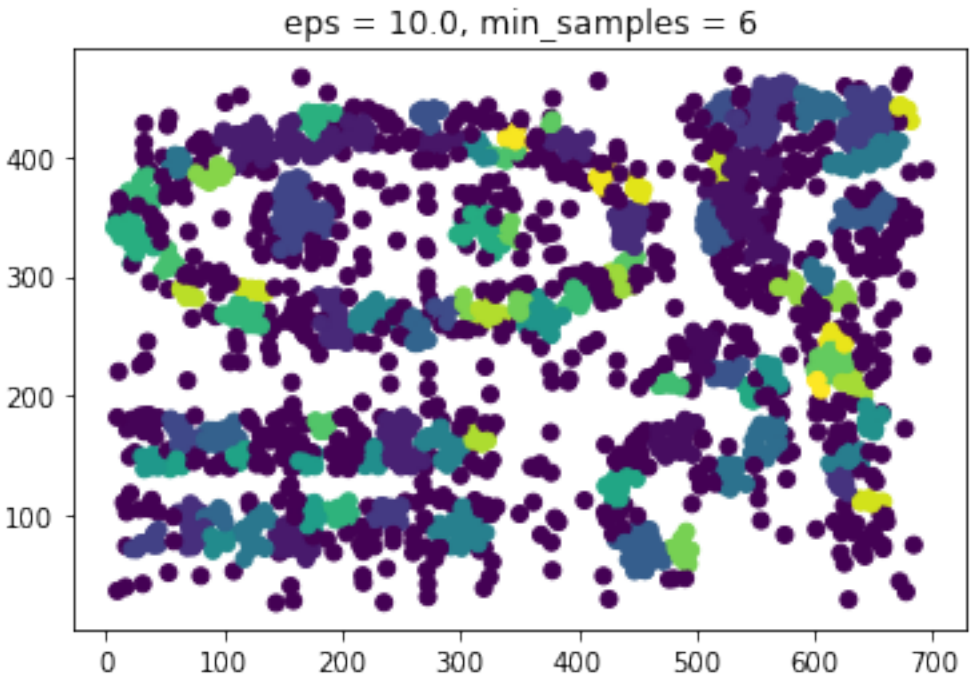

eps = 5.0, min_samples = 6
400
300
200
100
0
100
200
300
400
500
600
700

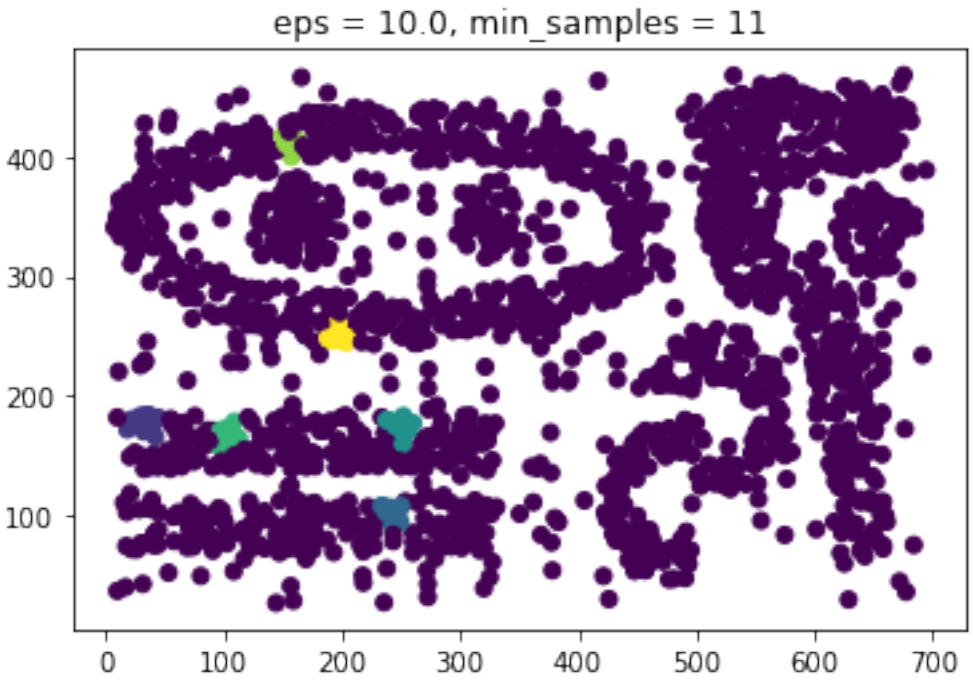

eps = 5.0, min_samples = 11
400
300
200
100
0
100
200
300
400
500
600
700

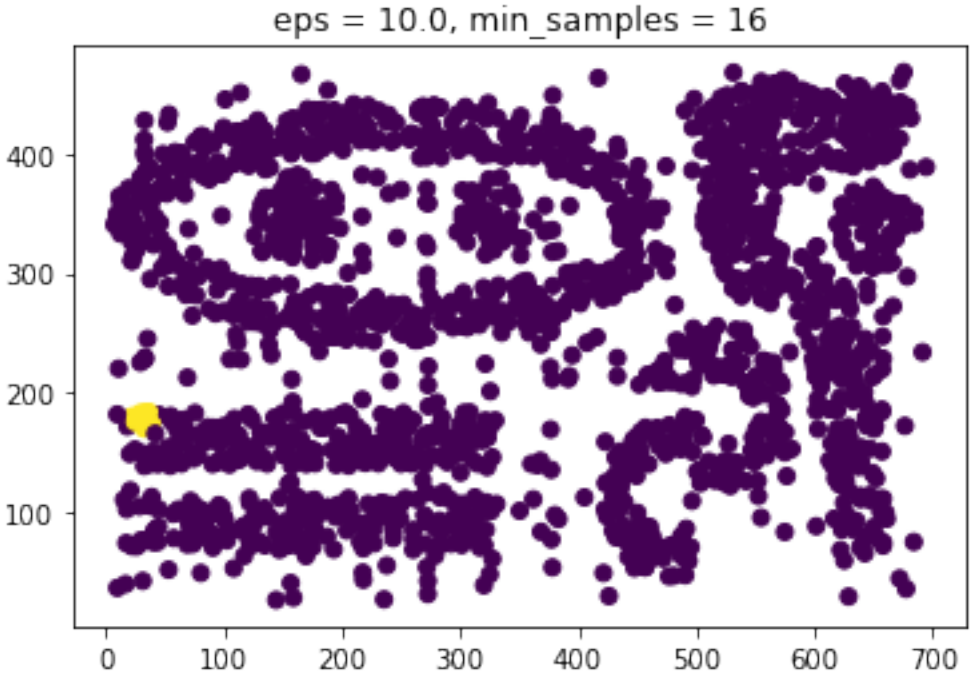

eps = 5.0, min_samples = 16
400
300
200
100
0
100
200
300
400
500
600
700

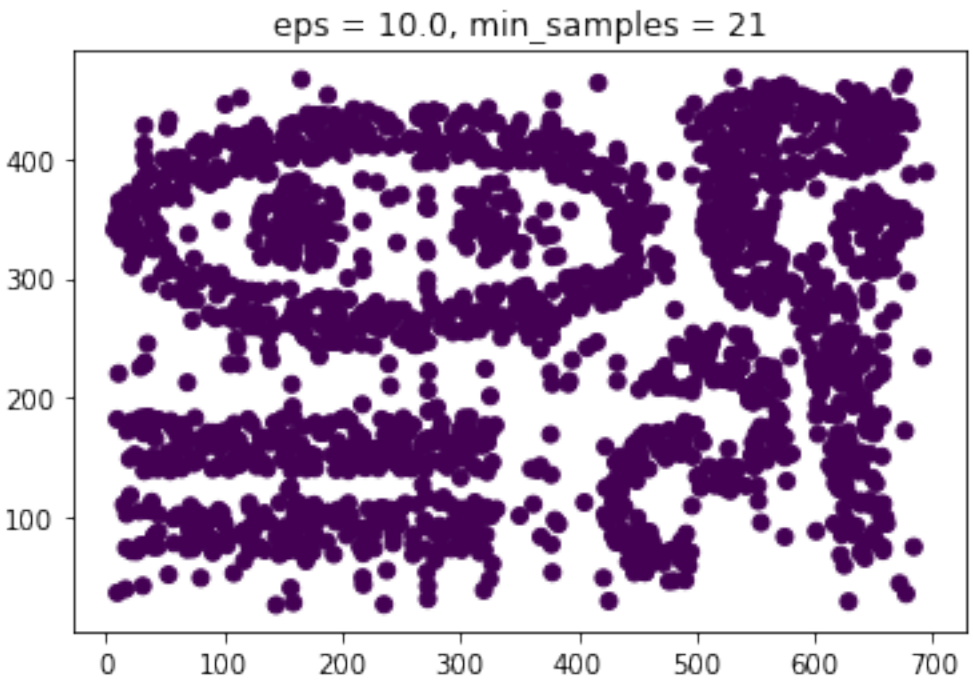

eps = 5.0, min_samples = 21
400
300
200
100
0
100
200
300
400
500
600
700

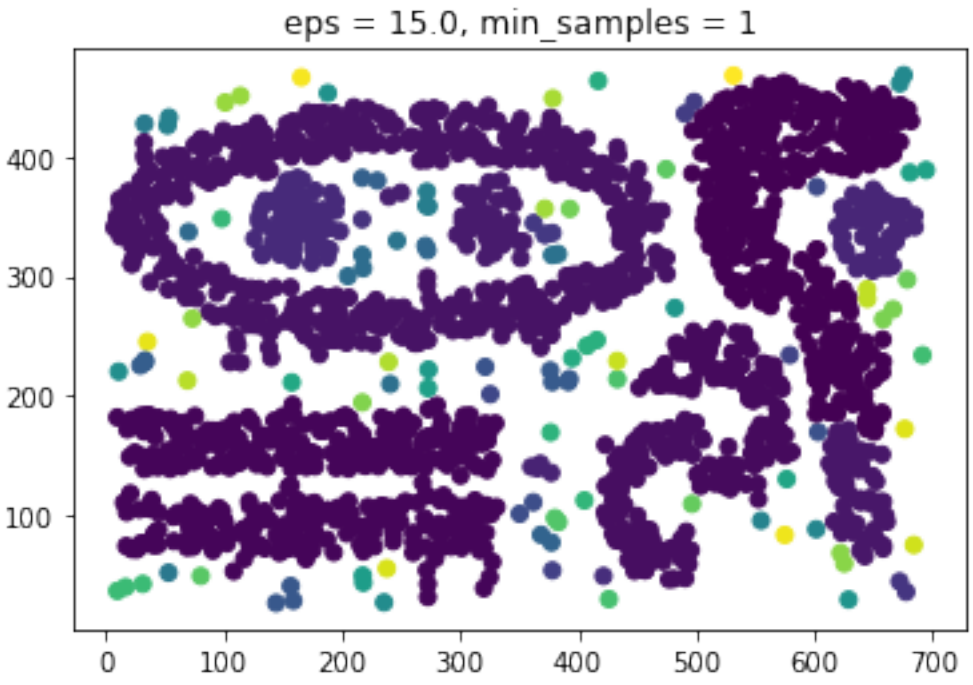

eps = 10.0, min_samples = 1
400
300
200
100
0
100
200
300
400
500
600
700

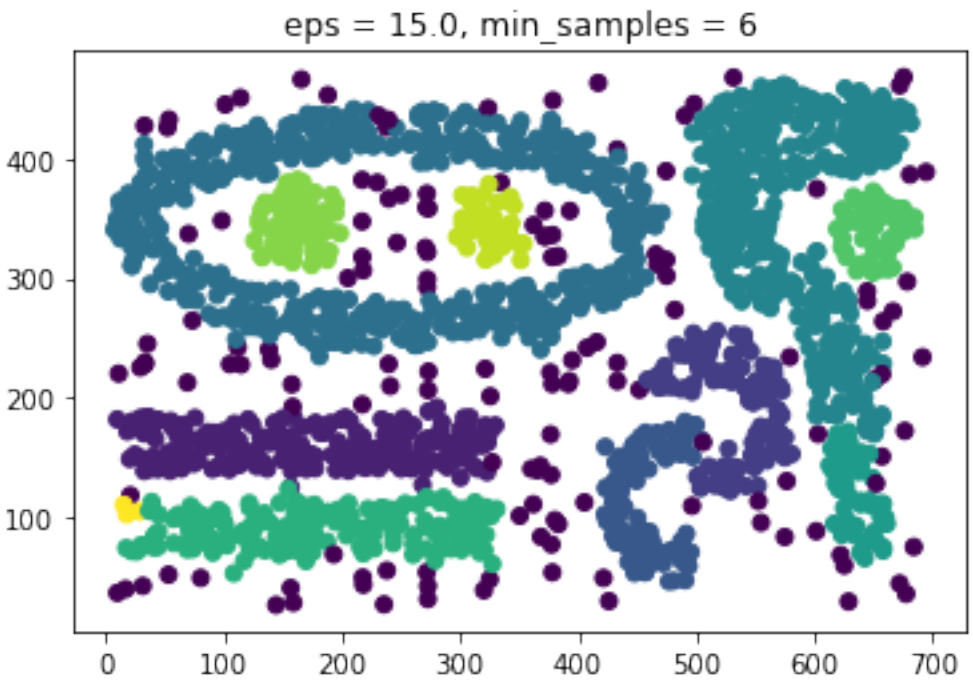

eps = 10.0, min_samples = 6
400
300
200
100
0
100
200
300
400
500
600
700

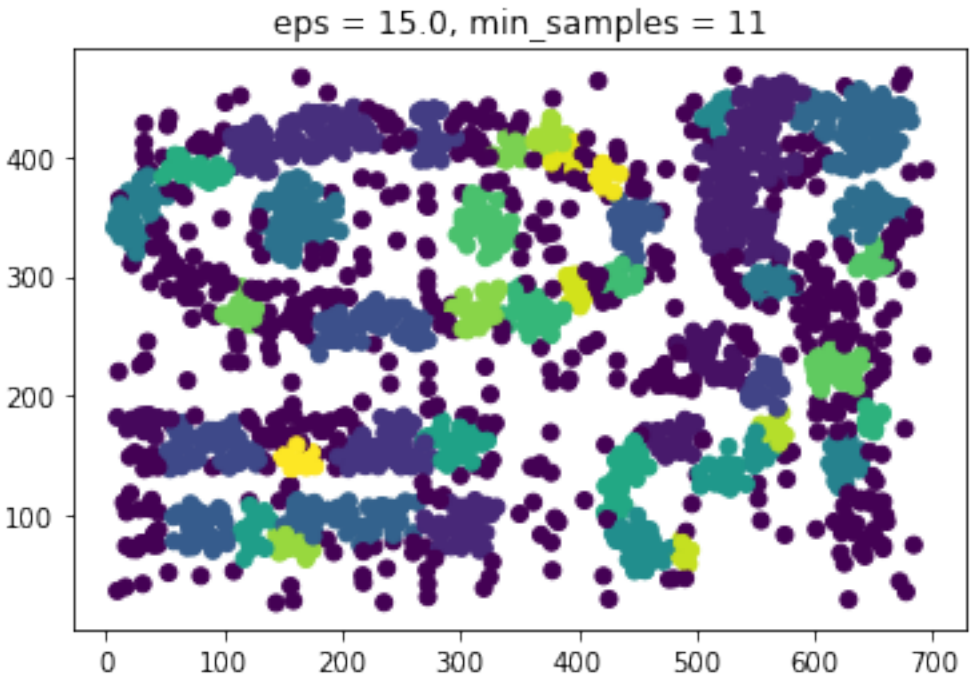

eps = 10.0, min_samples = 11
400
300
200
100
0
100
200
300
400
500
600
700

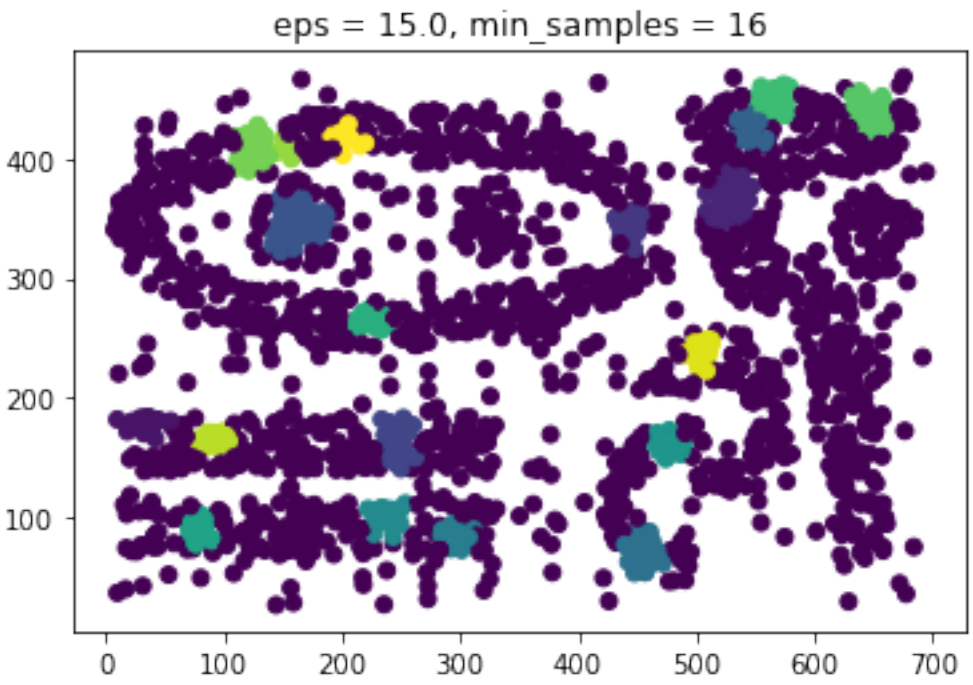

eps = 10.0, min_samples = 16
400
300
200
100
0
100
200
300
400
500
600
700

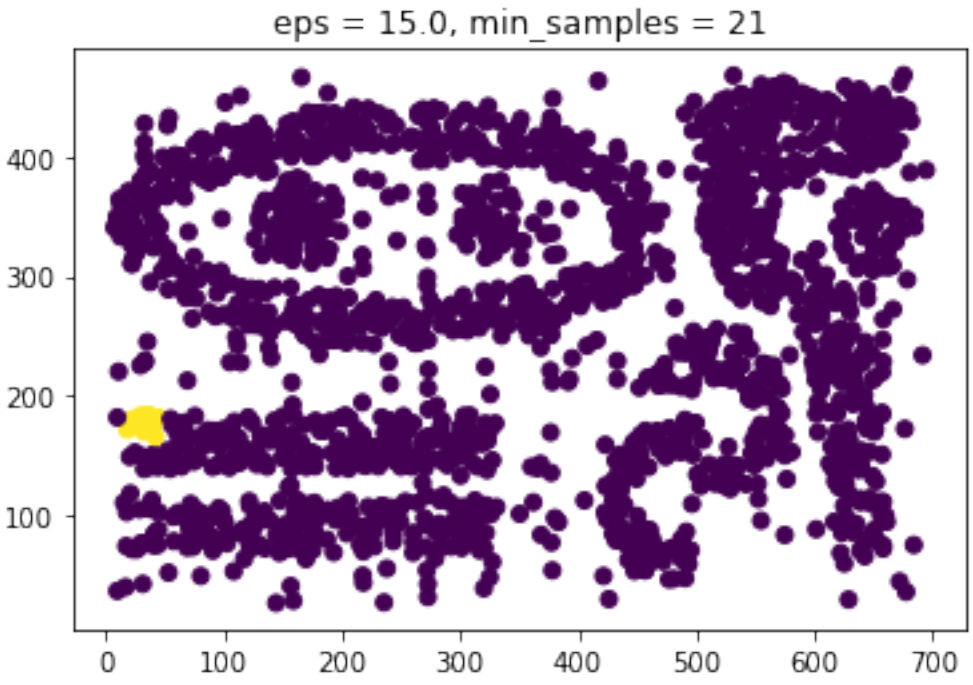

eps = 10.0, min_samples = 21
400
300
200
100
0
100
200
300
400
500
600
700

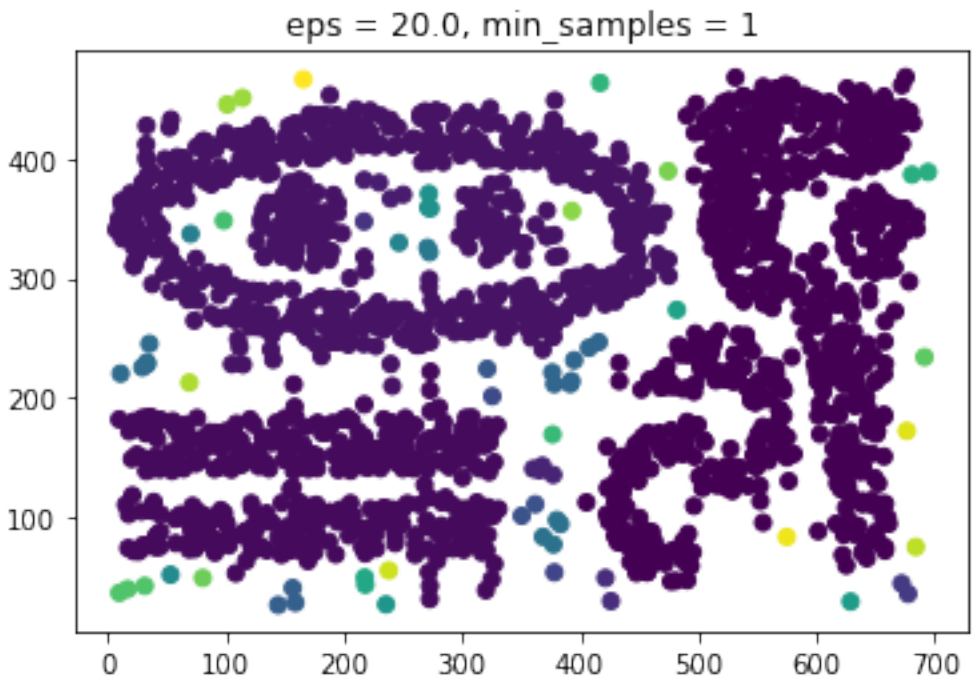

eps = 15.0, min_samples = 1
400
300
200
100
0
100
200
300
400
500
600
700

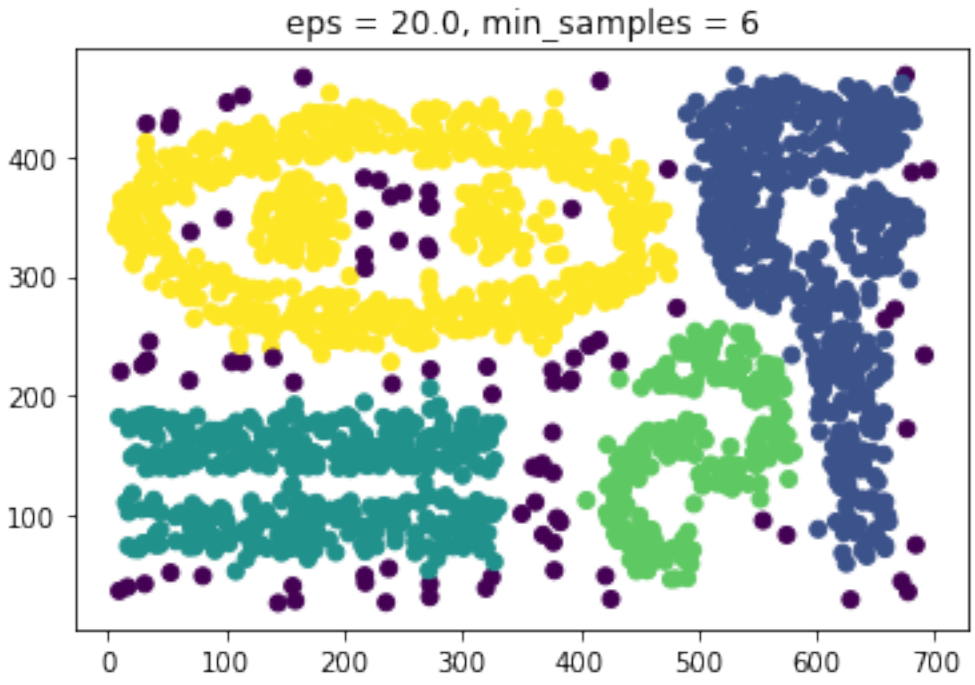

eps = 15.0, min_samples = 6
400
300
200
100
0
100
200
300
400
500
600
700

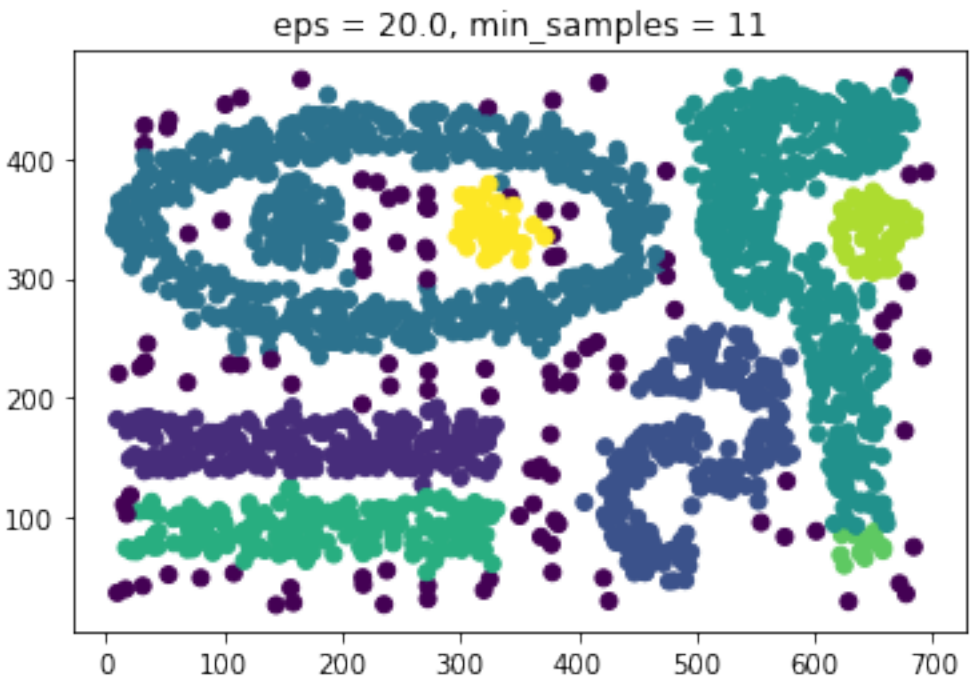

eps = 15.0, min_samples = 11
400
300
200
100
0
100
200
300
400
500
600
700

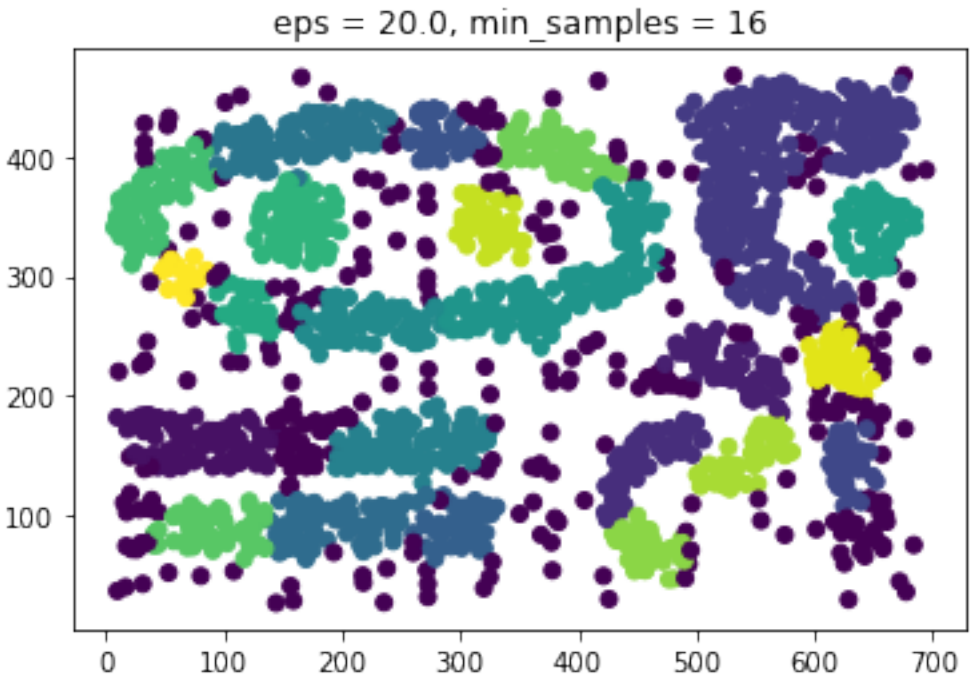

eps = 15.0, min_samples = 16
400
300
200
100
0
100
200
300
400
500
600
700

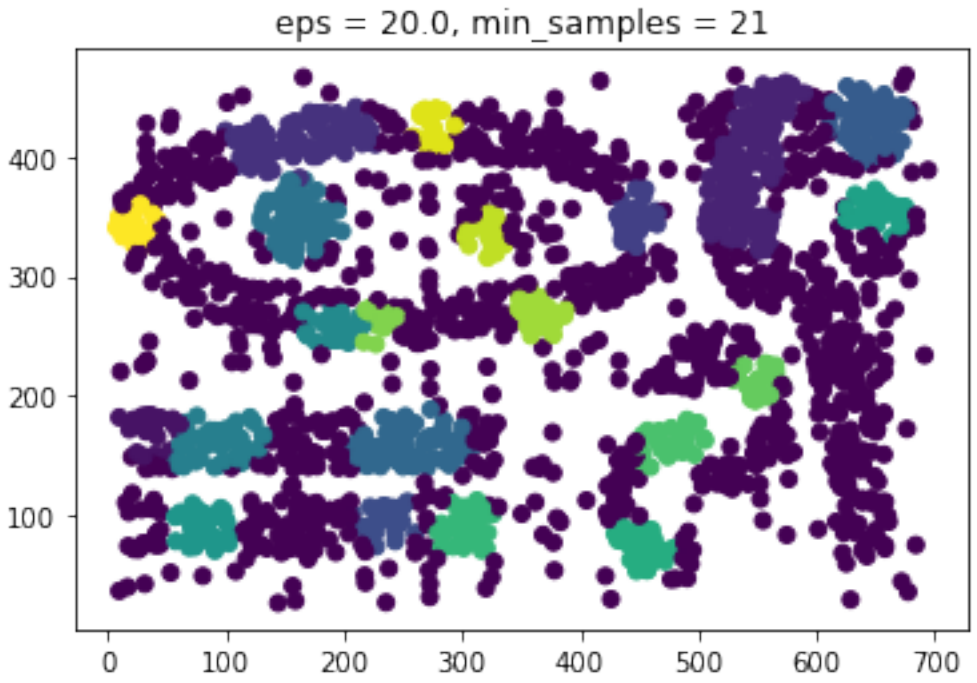

eps = 15.0, min_samples = 21
400
300
200
100
0
100
200
300
400
500
600
700


eps = 20.0, min_samples = 1
400
300
200
100
0
100
200
300
400
500
600
700


eps = 20.0, min_samples = 6
400
300
200
100
0
100
200
300
400
500
600
700


eps = 20.0, min_samples = 11
400
300
200
100
0
100
200
300
400
500
600
700


eps = 20.0, min_samples = 16
400
300
200
100
0
100
200
300
400
500
600
700



The result provided in (b) seems to be the best. However, some solutions in (c) can be considered good as well. especially eps=15 and min_samples=6 (unsuprisingly as this is quite close to the parameters used in (b)), eps=20 and min_samples=6 and eps=20 and min_samples=11. Generally speaking we get an increased number of clusters with increasing min_samples, which peaks at some number, after which no points in the neighborhood of another point can be found. Eps on the other hand seems to have a severe influence on the size of the obtained clusters. Which is backed up by the documentation which describes eps as The maximum distance between two samples for one to be considered as in the neighborhood of the other