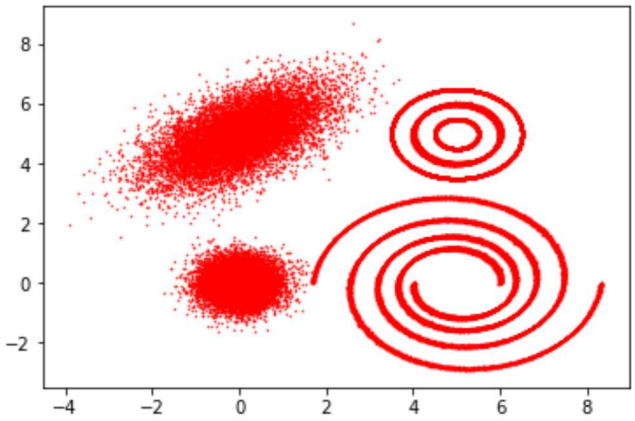
Out[1]: The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.

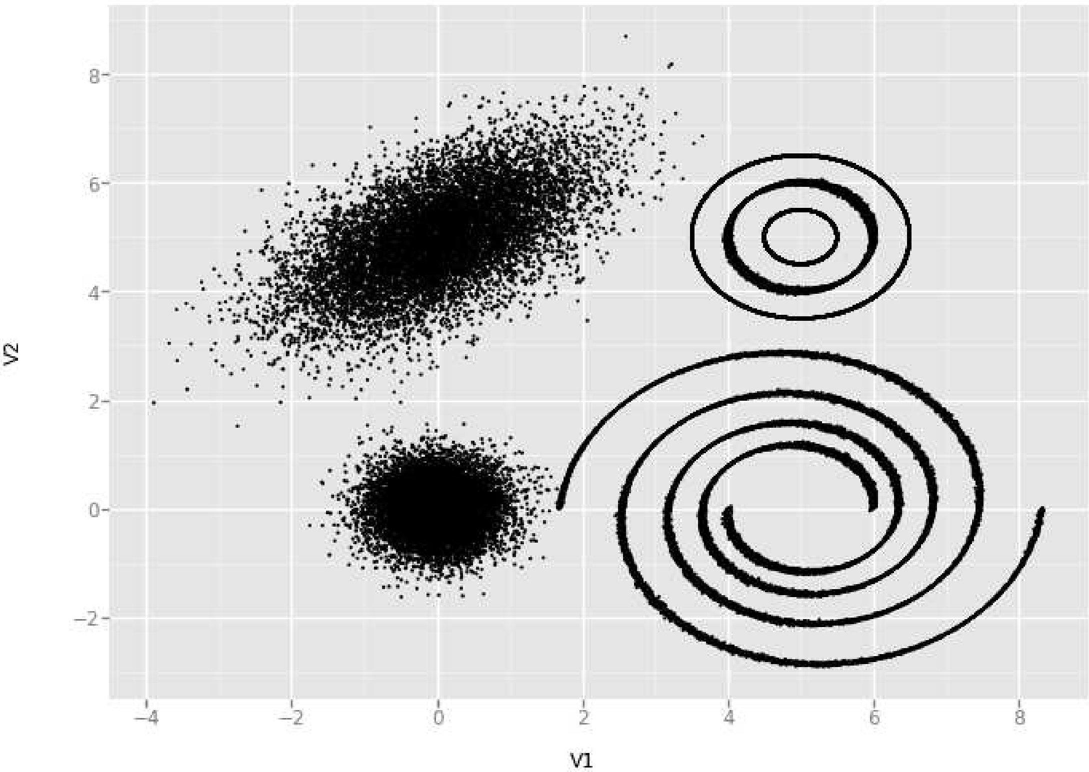
Exercice 1

- Chargez les données donclassif.csv, qui contient des point sur un plan de deux dimensions.
- Afficher ces sur un plan de deux dimensions
- Testez les algorithmes de clustering : hiérarchique et kmeans. Essayer d'évaluer le nombre de clusters optimal à l'aide de de la métrique Davies Bouldin
- Affichez le résultat sur un graphe qui illustre les différents points, portant des couleurs refletant le cluster auquel ils appartiennent.
- Refaire la même chose avec les données donclassif2.csv. Que remarquez vous au sujet du clustering hiérachique ?
- Pour résoudre ces problèmes de lenteur du clustering hiérarchique, une des méthodes utilisée consiste à faire un clustering à l'aide de l'algorithme kmeans, avec k très grand, puis appliquer la classification hiérarchique sur les centroides détectés par kmeans. Appliquez cette méthode sur le jeux de données donclassif2.csv

Out[3]:

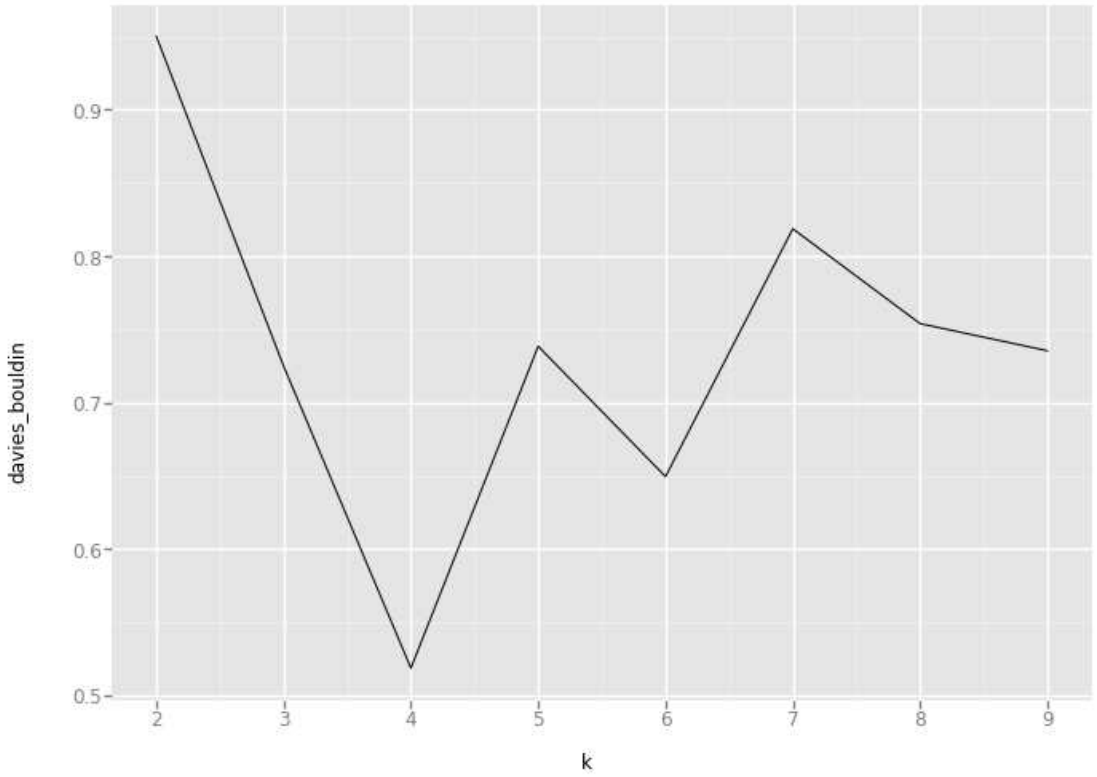
	V1	V2
0	5.971089	-0.027773
1	6.035601	0.037879
2	5.968887	-0.027694
3	5.953036	-0.042402
4	6.034341	0.040049






```
Out[5]: <ggplot: (19214463)>
```

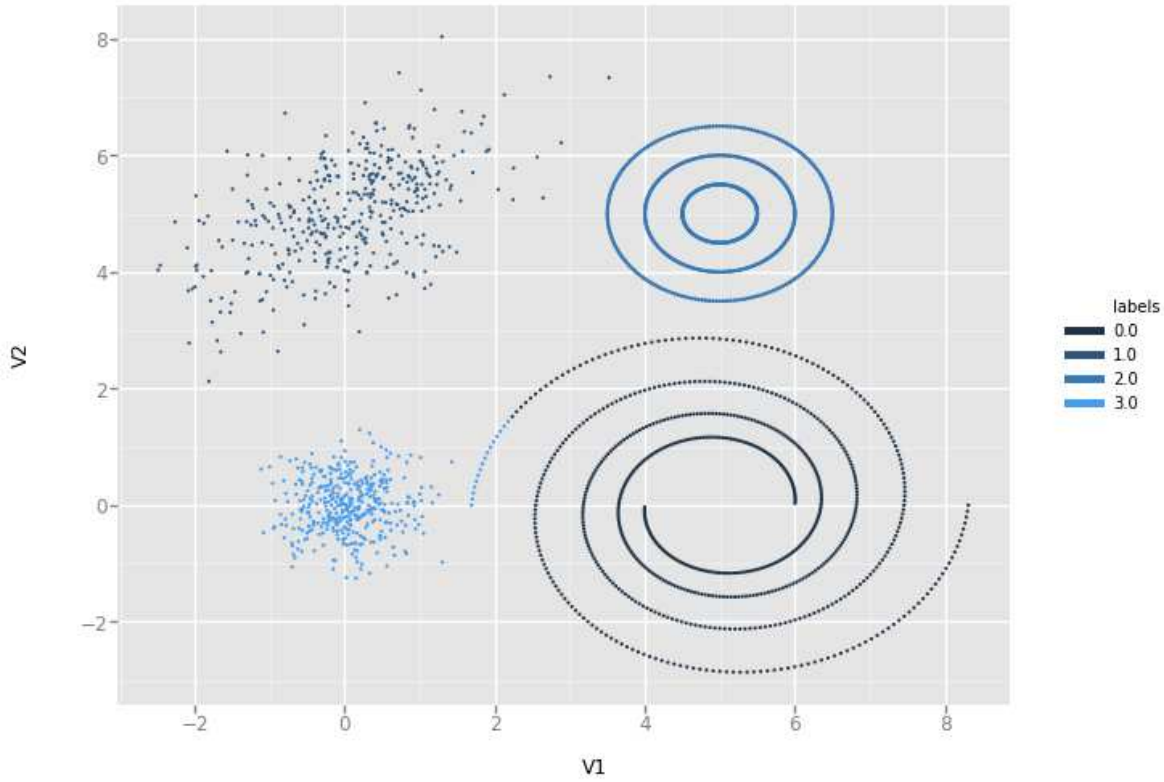
```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cluster\unsupervised.
py:342: RuntimeWarning: divide by zero encountered in true_divide
  score = (intra_dists[:, None] + intra_dists) / centroid_distances
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cluster\unsupervised.
py:342: RuntimeWarning: divide by zero encountered in true_divide
  score = (intra_dists[:, None] + intra_dists) / centroid_distances
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cluster\unsupervised.
py:342: RuntimeWarning: divide by zero encountered in true_divide
  score = (intra_dists[:, None] + intra_dists) / centroid_distances
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cluster\unsupervised.
py:342: RuntimeWarning: divide by zero encountered in true_divide
  score = (intra_dists[:, None] + intra_dists) / centroid_distances
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cluster\unsupervised.
py:342: RuntimeWarning: divide by zero encountered in true_divide
  score = (intra_dists[:, None] + intra_dists) / centroid_distances
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cluster\unsupervised.
py:342: RuntimeWarning: divide by zero encountered in true_divide
  score = (intra_dists[:, None] + intra_dists) / centroid_distances
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cluster\unsupervised.
py:342: RuntimeWarning: divide by zero encountered in true_divide
  score = (intra_dists[:, None] + intra_dists) / centroid_distances
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cluster\unsupervised.
py:342: RuntimeWarning: divide by zero encountered in true_divide
  score = (intra_dists[:, None] + intra_dists) / centroid_distances
```



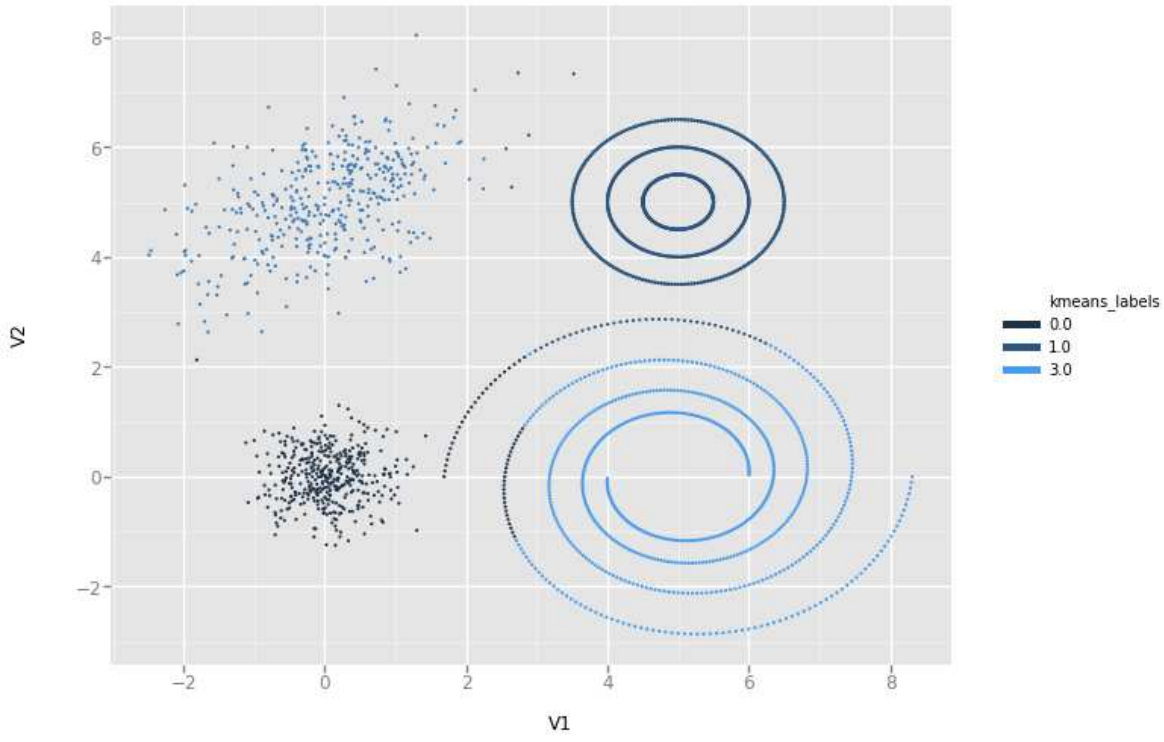
Out[16]: <ggplot: (-9223372036835716590)>

Out[26]:

	V1	V2	labels
0	6.002504	0.031505	0
1	6.004021	0.063168	0
2	6.004545	0.094957	0
3	6.004069	0.126843	0
4	6.002587	0.158794	0



Out[33]: <ggplot: (22797951)>



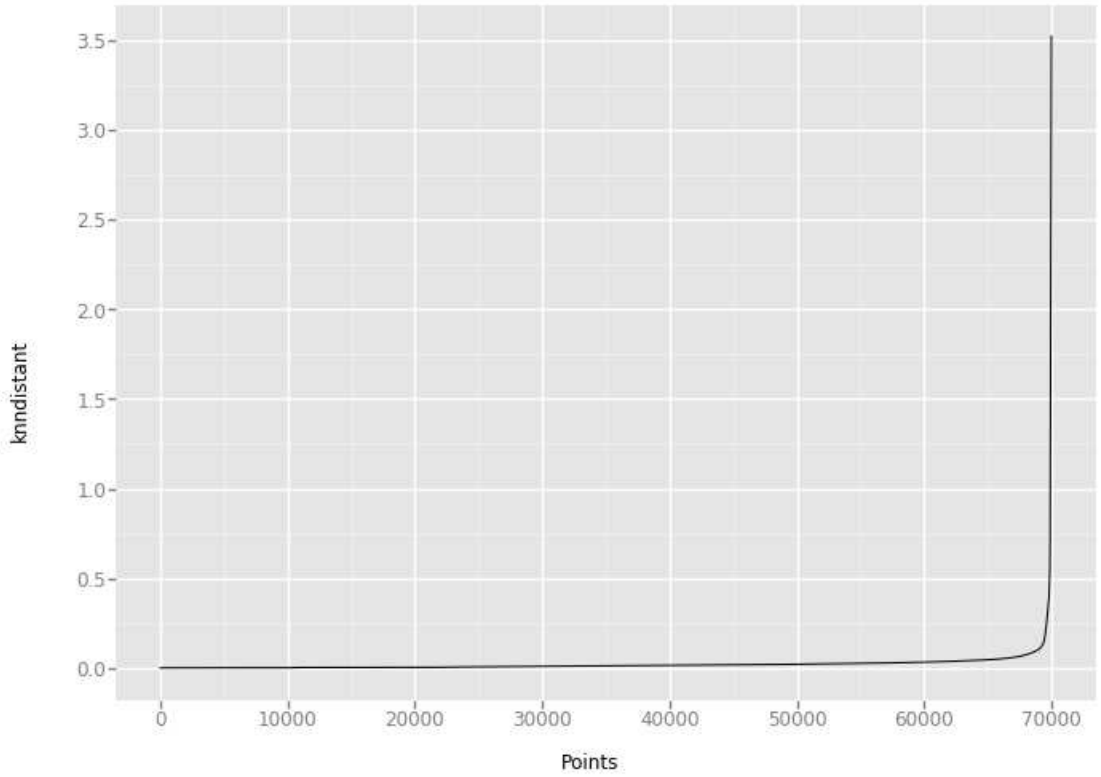
Out[56]: <ggplot: (-9223372036830527151)>

```
Out[14]: array([[ 0.        ,  0.03169906,  0.06348535, ...,  7.20851478,
           7.88895877,  7.15016071],
         [ 0.03169906,  0.        ,  0.03179412, ...,  7.18698237,
           7.86928246,  7.13333869],
         [ 0.06348535,  0.03179412,  0.        , ...,  7.16473568,
           7.84884996,  7.11572694],
         ...,
         [ 7.20851478,  7.18698237,  7.16473568, ...,  0.        ,
           0.89516726,  1.3426177 ],
         [ 7.88895877,  7.86928246,  7.84884996, ...,  0.89516726,
           0.        ,  1.10776747],
         [ 7.15016071,  7.13333869,  7.11572694, ...,  1.3426177 ,
           1.10776747,  0.        ]])
```

Exercice 2

- DBSCAN est un algorithme de clustering basé sur la notion de densité. Il nécessite deux paramètres (eps, et minPts) qui reflète la densité des cluster recherchés. Estimez cette densité à l'aide de la méthode vue en cours, à savoir le graphe qui illustre la distance de chaque point à son Nieme plus proche voisin
- Déduisez les paramètre à utiliser pour DBSCAN, exécutez le, et affichez le résultat sur un graphe qui illustre les différents points, portant des couleurs refletant le cluster auquel ils appartiennent.

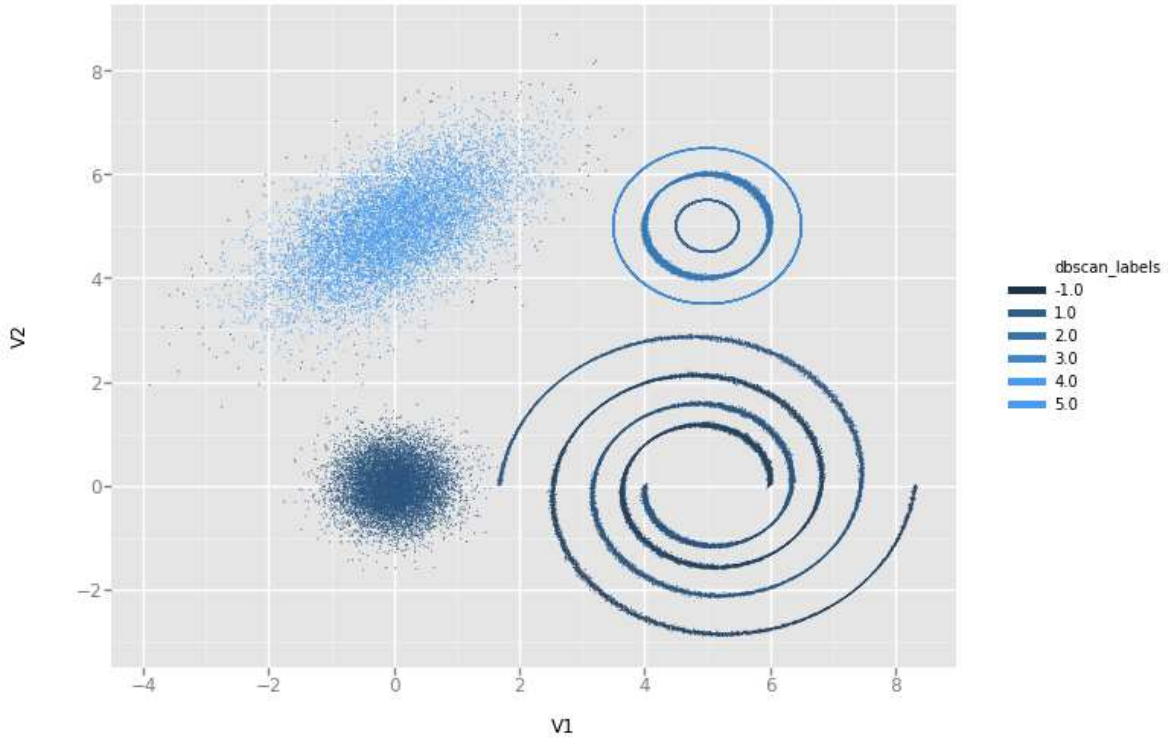
```
[ 0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664  0.00125664
```




```
Out[133]: <ggplot: (29480133)>
```

Out[152]:

	V1	V2	dbscan_labels
0	5.971089	-0.027773	0
1	6.035601	0.037879	0
2	5.968887	-0.027694	0
3	5.953036	-0.042402	0
4	6.034341	0.040049	0
5	5.988497	-0.004648	0
6	6.011308	0.019312	0
7	6.017807	0.026961	0
8	5.991385	0.001692	0
9	5.990503	0.001965	0
10	6.016895	0.029513	0
11	6.013283	0.027059	0
12	6.019942	0.034879	0
13	6.006124	0.022222	0
14	6.019036	0.036298	0
15	5.973406	-0.008167	0
16	6.009020	0.028615	0
17	5.980327	0.001091	0
18	5.992259	0.014195	0
19	6.004450	0.027559	0
20	6.019915	0.044198	0
21	6.007305	0.032765	0
22	6.023378	0.050017	0
23	6.027773	0.055592	0
24	5.992140	0.021141	0
25	5.987520	0.017705	0
26	6.011919	0.043290	0
27	6.013203	0.045762	0
28	6.010351	0.044099	0
29	5.992740	0.027680	0
...	...	...	...
9970	8.284040	-0.135053	0
9971	8.324143	-0.091362	0
9972	8.300520	-0.111391	0
9973	8.300699	-0.107611	0
9974	8.319475	-0.085229	0
9975	8.279685	-0.121405	0
9976	8.281236	-0.116235	0
9977	8.329228	-0.064617	0
9978	8.346606	-0.043607	0




```
Out[139]: <ggplot: (36901675)>
```

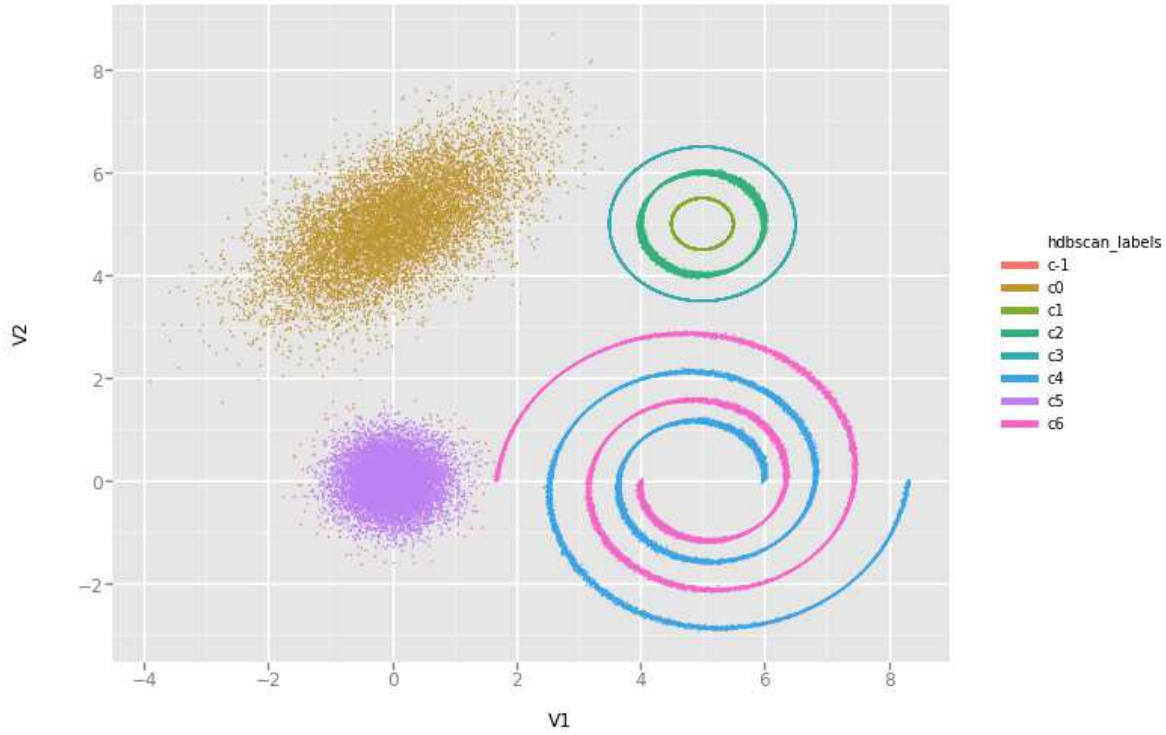
Exercice 3

- Effectuez le clustering HDBSCAN sur les deux jeux de données en testant différentes valeurs du paramètre min_cluster_size
- Affichez l'arbre couvrant de poid minimal (Minimum Spanning Tree)
- Affichez la vue hiérarchique des clusters

```
Out[55]: HDBSCAN(algorithm='best', allow_single_cluster=False, alpha=1.0,
             approx_min_span_tree=True, cluster_selection_method='eom',
             core_dist_n_jobs=4, gen_min_span_tree=True, leaf_size=40,
             match_reference_implementation=False, memory=Memory(cachedir=None),
             metric='euclidean', min_cluster_size=6, min_samples=None, p=None,
             prediction_data=False)
```

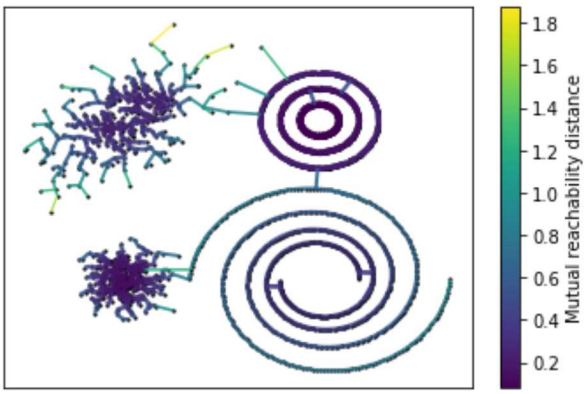
Out[48]:

	V1	V2	hdbscan_labels
0	5.971089	-0.027773	c4
1	6.035601	0.037879	c4
2	5.968887	-0.027694	c4
3	5.953036	-0.042402	c4
4	6.034341	0.040049	c4
5	5.988497	-0.004648	c4
6	6.011308	0.019312	c4
7	6.017807	0.026961	c4
8	5.991385	0.001692	c4
9	5.990503	0.001965	c4
10	6.016895	0.029513	c4
11	6.013283	0.027059	c4
12	6.019942	0.034879	c4
13	6.006124	0.022222	c4
14	6.019036	0.036298	c4
15	5.973406	-0.008167	c4
16	6.009020	0.028615	c4
17	5.980327	0.001091	c4
18	5.992259	0.014195	c4
19	6.004450	0.027559	c4
20	6.019915	0.044198	c4
21	6.007305	0.032765	c4
22	6.023378	0.050017	c4
23	6.027773	0.055592	c4
24	5.992140	0.021141	c4
25	5.987520	0.017705	c4
26	6.011919	0.043290	c4
27	6.013203	0.045762	c4
28	6.010351	0.044099	c4
29	5.992740	0.027680	c4
...	...	...	...
69970	0.913362	5.535590	c0
69971	0.383776	5.631329	c0
69972	0.131741	4.417212	c0
69973	-0.968164	5.390710	c0
69974	0.519996	5.213785	c0
69975	3.084504	6.884631	c0
69976	-1.220914	5.000711	c0
69977	1.343912	5.642842	c0
69978	0.336318	5.504442	c0



Out[58]: <ggplot: (36154639)>

Out[100]: <matplotlib.axes._subplots.AxesSubplot at 0x43ccc780>



Out[102]: <matplotlib.axes._subplots.AxesSubplot at 0x43ffd940>