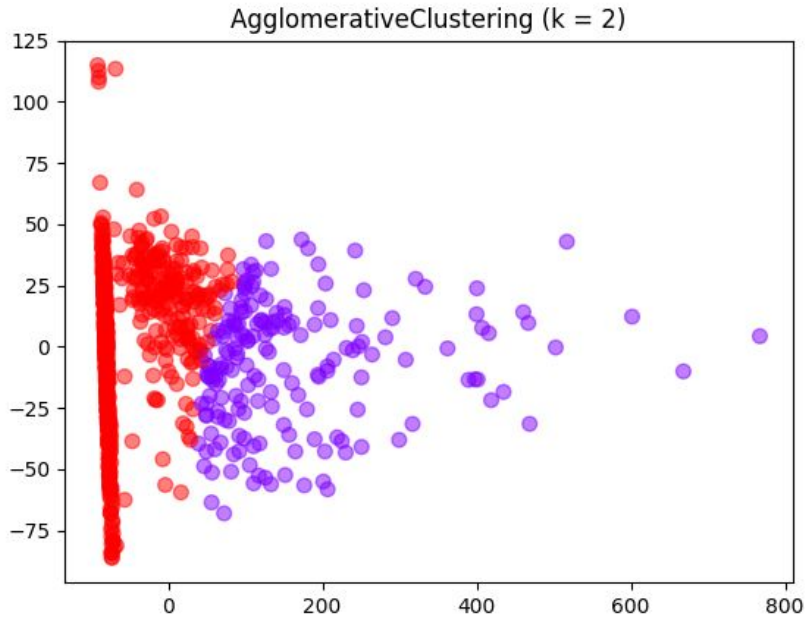
Código: https://github.com/Barbalho12/statistical-test

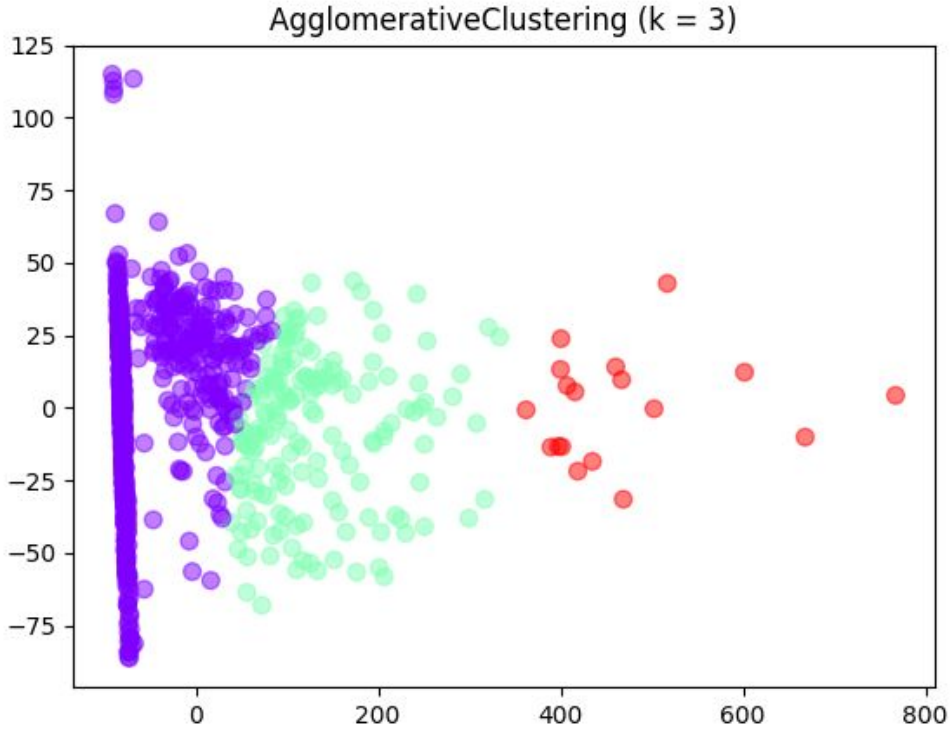
Tarefa 1 (clustering.py)

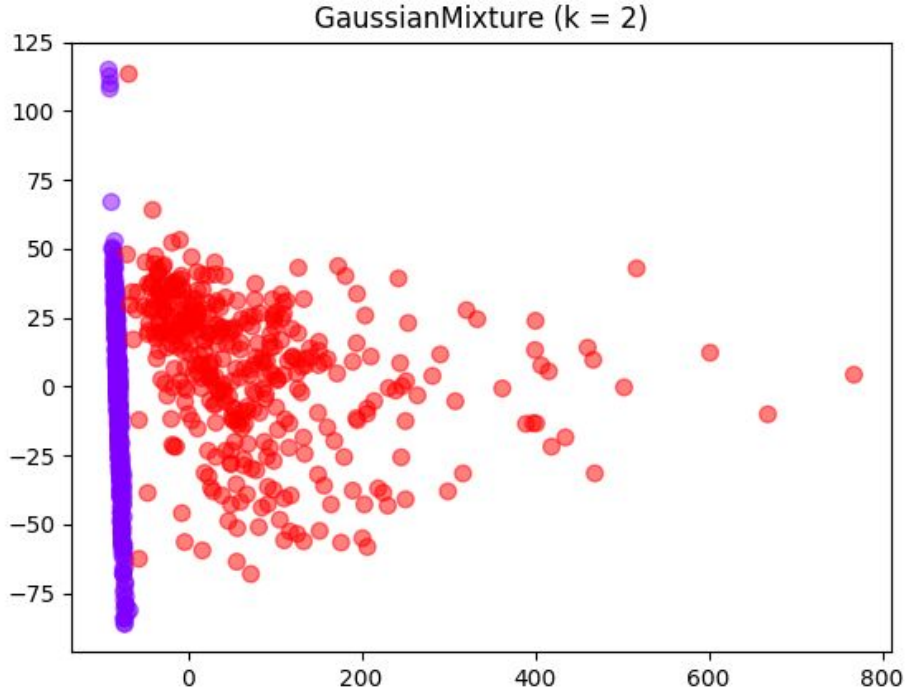
1. Executar os três algoritmos de clustering sobre o dataset, variando o número de grupos (k) de 2 a 3

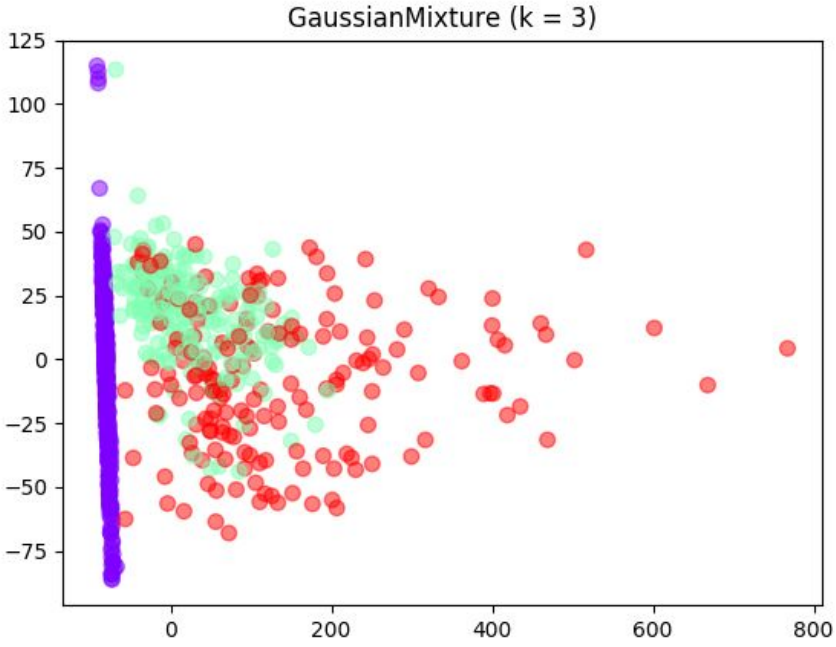
```
models = [
    { 't': "AgglomerativeClustering (k = 2)", 'm': AgglomerativeClustering(n_clusters=2) },
    { 't': "AgglomerativeClustering (k = 3)", 'm': AgglomerativeClustering(n_clusters=3) },
    { 't': "KMeans (k = 2)", 'm': KMeans(n_clusters=2) },
    { 't': "KMeans (k = 3)", 'm': KMeans(n_clusters=3) },
    { 't': "GaussianMixture (k = 2)", 'm': GaussianMixture(n_components=2) },
    { 't': "GaussianMixture (k = 3)", 'm': GaussianMixture(n_components=3) }
]
```

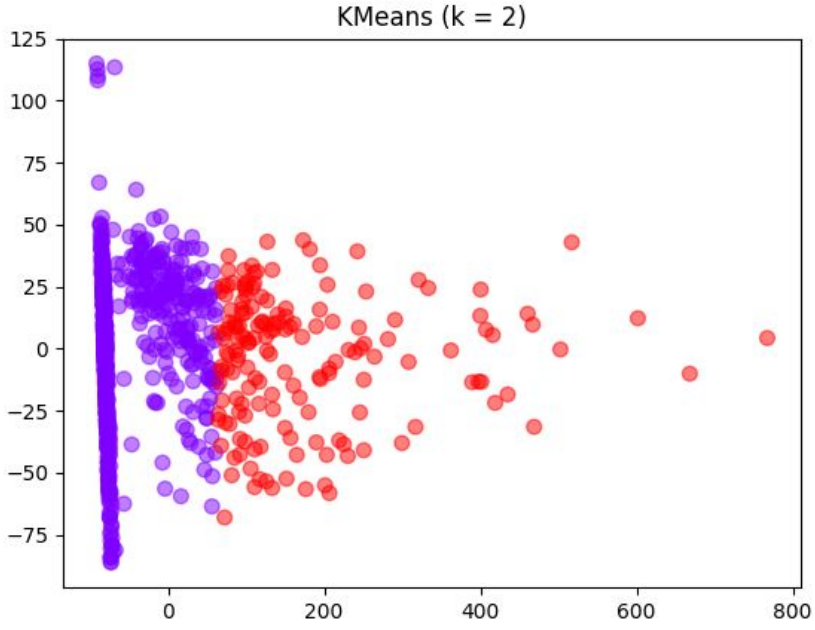
2. Visualizar os resultados através de gráficos

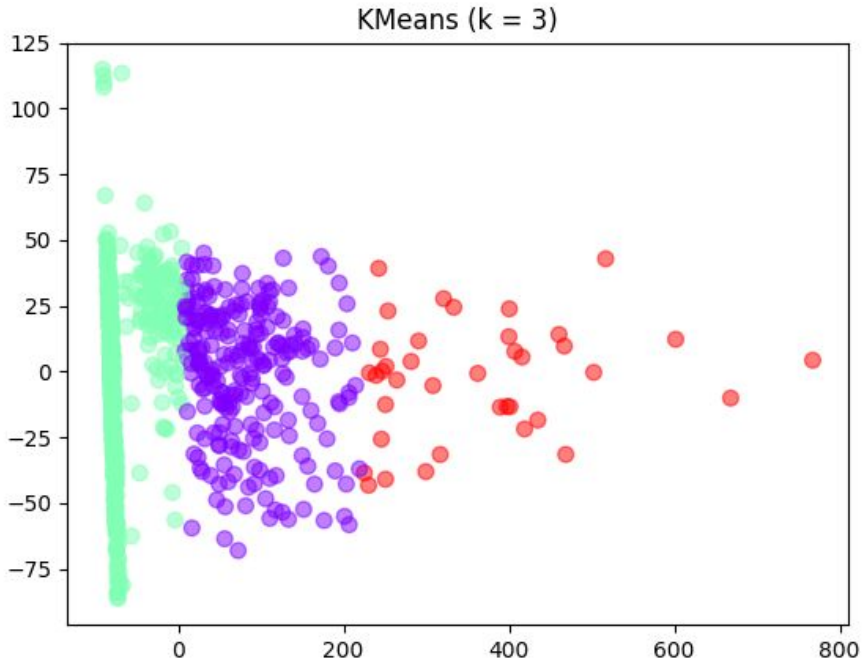



AgglomerativeClustering (k = 3)
125
100
75
50
25
0
−25
−50
−75
0
200
400
600
800


GaussianMixture (k = 2)
125
100
75
50
25
0
−25
−50
−75
0
200
400
600
800


GaussianMixture (k = 3)
125
100
75
50
25
0
−25
−50
−75
0
200
400
600
800


KMeans (k = 2)
125
100
75
50
25
0
−25
−50
−75
0
200
400
600
800



3. Fazer a validação dos grupos através da utilização dos índices (Davies Bouldin e Silhouette)

```
46  print('\n3. Fazer a validação dos grupos através da utilização dos índices (Davies Bouldin e Silhouette)')
47
48  print('{}\t{}\t{}'.format('Model', 'Bouldin', 'Silhouette'))
49  for m in models:
50      labels = m['m'].fit_predict(X)
51      dbScore = davies_bouldin_score(X, labels)
52      sScore = silhouette_score(X, labels, metric='euclidean')
53      print('{}\t{}\t{}'.format(m['t'], dbScore, sScore))
54
55  print('\n4. Mostrar os resultados de ambos os índices para os agrupamentos criados')
```

4. Mostrar os resultados de ambos os índices para os agrupamentos criados

```
--------------------------------------------------------------------------
Modelo                              Bouldin                 Silhouette
--------------------------------------------------------------------------
AgglomerativeClustering (k = 2)     0.7330018210488929      0.5532678504628996
KMeans (k = 2)                      0.7133822795826191      0.5687897205830247
GaussianMixture (k = 2)             0.8604650094596924      0.3919496047300402

AgglomerativeClustering (k = 3)     0.6041813066360704      0.5281675826566276
KMeans (k = 3)                      0.6680978941351275      0.5104287492214447
GaussianMixture (k = 3)             0.7379219639790152      0.4240572985480604
--------------------------------------------------------------------------
```

5. Analisar os resultados obtidos

Quanto maior o silhouette, melhor é o resultado. Já o DB, é o contrário. Dessa forma para k=2, o *KMeans* obteve melhor resultado e o *GaussianMixture* o pior. Já para k = 3, o *AgglomerativeClustering* foi o melhor e o *GaussianMixture* permaneceu como a pior solução.

Tarefa 2 (statistical_test.py)

```python
# coding=utf-8
from scipy.stats import friedmanchisquare
import scikit_posthocs as sp #pip3 install scikit-posthocs
import pandas as pd

df = pd.read_csv('csv/comparativo_tecnicas.csv',encoding='utf-8')

print('\nFriedman')

stat, p = friedmanchisquare(df['dt'].tolist(), df['nb'].tolist(), df['mlp'].tolist())
print('p=%.3f' % (p))
if p > 0.05:
    print('Não há diferença significativa')

else:
    print('Há diferença significativa')

    print('\n Posthoc')
    posthoc = sp.posthoc_nemenyi([df['dt'].tolist(), df['nb'].tolist(), df['mlp'].tolist()])
    print(posthoc)
```

Saída:

Friedman
pValue = 0.068
Não há diferença significativa