UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Geociências e Ciências Exatas - IGCE
Curso de Bacharelado em Ciências da Computação

GABRIEL LUIZ

TRABALHO DE CLUSTERING

Professora: Dra. Adriane Beatriz de Souza Serapião

Rio Claro - SP

2020

1 Introdução

Este trabalho consiste em aplicar o conhecimento de clustering adquirido na disciplina Tópicos: Aprendizado de Máquina, tendo assim como objetivo:

- Escolha dois datasets rotulados.
- Realize a análise estatística, visualização e pré-processamento dos dados.
- Realize os experimentos criando duas bases de teste distintas:
  - considerando todos os atributos do dataset ;
  - selecionando alguns atributos e descartando outros;
- Aplique três métodos de clustering distintos nas duas bases acima.
- Para cada dataset , em cada uma das bases, analise os resultados segundo medidas de qualidade de clustering , usando índices de validação interna (SSW, SSB, silhueta, Calinski-Harabasz, Dunn e Davis-Bouldin) e externa (pureza, entropia, acurácia, F-measure , ARI, NMI).
- Proponha uma maneira adicional de comparar os resultados obtidos além das medidas acima.
- Compare e interprete os resultados dos dois experimentos em cada dataset

2 Desenvolvimento

Para o desenvolvimento das atividades inicialmente foi escolhido duas base de dados. As bases foram encontradas no site http://cs.uef.fi/sipu/datasets/ onde possuem datasets próprios para a tarefa de clustering, os dataset não possuem informações de que se referem cada atributo ou cada instancia.

2.1 Pré-processamento e Visualização

Para realizar o pré-processamento foi necessário validar se os dados não possuiam números vazios ou algum tipo de valor que foge do esperado.

2.2 Validação dos dados

Foi validado que os dataset não possuem nenhum valor nulo ou valores diferentes de inteiro maior do que zero.

2.3 Análise dos dados

Com os valores todos normalizados podemos ver a correlação entre os atributos, que possuem alta relação em alguns casos. 1

Após a visualazação dos dados foi gerado o bloxpot 3 para ver como está a distribuição dos dados onde é possível ver que poucos atributos possuem outliers e os dados possuem certa distribuição padrão, e também os valores de media, moda e mediana para cada atributo.2

2.4 Escalonamento

Para aplicar os algoritmos de clustering, é necessário escalonar os dados, normalizando eles em uma faixa de -1 a 1, onde os dados irão manter a mesma proporção e similaridades.

2.5 Algoritmos de Clustering

Como solicitado na tarefa, deve ser aplicado 3 métodos de clustering para visualização dos dados, o que foi selecionado neste caso são, K-means, Agglomerative Clustering

	0	1	2	3	4	5	6	7	8	9	...	22	23	24
0	1.000000	0.268198	-0.051122	-0.068849	0.599398	-0.438830	0.041834	0.122806	0.140755	0.017996	...	0.298212	0.022004	0.193500
1	0.268198	1.000000	-0.434193	0.065247	0.147223	-0.087154	0.052960	0.112432	0.227755	0.442574	...	0.218035	-0.323157	0.388152
2	-0.051122	-0.434193	1.000000	-0.212930	0.077845	0.065985	-0.022429	-0.216804	-0.180707	-0.143382	...	0.142550	-0.167976	-0.479971
3	-0.068849	0.065247	-0.212930	1.000000	-0.049977	-0.004621	0.348210	0.042810	0.093820	0.186358	...	0.056818	0.125137	0.050242
4	0.599398	0.147223	0.077845	-0.049977	1.000000	-0.512794	-0.189263	0.334837	0.483320	0.257335	...	0.427911	-0.102514	0.071684
5	-0.438830	-0.087154	0.065985	-0.004621	-0.512794	1.000000	0.549921	-0.635187	-0.236149	-0.517697	...	0.125967	-0.170035	0.165715
6	0.041834	0.052960	-0.022429	0.348210	-0.189263	0.549921	1.000000	-0.410581	-0.194464	-0.334774	...	0.240729	-0.315245	0.261003
7	0.122806	0.112432	-0.216804	0.042810	0.334837	-0.635187	-0.410581	1.000000	0.546772	0.702223	...	-0.178068	0.247391	0.277945
8	0.140755	0.227755	-0.180707	0.093820	0.483320	-0.236149	-0.194464	0.546772	1.000000	0.271853	...	-0.024533	0.151648	0.257148
9	0.017996	0.442574	-0.143382	0.186358	0.257335	-0.517697	-0.334774	0.702223	0.271853	1.000000	...	0.087693	-0.236956	0.198970
10	-0.416168	-0.247372	-0.078198	-0.256024	-0.421847	0.422388	0.085398	-0.092287	-0.240591	-0.328448	...	-0.033714	0.088643	0.031608
11	-0.325456	0.049006	-0.005908	-0.023452	-0.093708	0.000218	-0.346374	0.120182	0.052129	0.429568	...	0.142397	-0.407174	-0.075290
12	0.149802	-0.101557	-0.327047	0.109485	0.133371	-0.173939	0.251279	0.409645	0.260425	0.078683	...	-0.473271	0.256056	0.430193
13	0.494533	0.135398	-0.275698	0.165248	0.393054	-0.037131	0.037670	0.161548	0.404304	0.132367	...	0.173041	0.090210	0.280425
14	0.054404	0.253658	0.027675	0.226054	0.026300	0.018823	0.180490	-0.027761	0.383065	-0.032232	...	-0.243979	0.092168	0.168610
15	-0.029940	-0.281726	-0.376820	-0.159598	-0.033689	-0.125013	0.108979	0.221851	-0.183596	-0.171017	...	-0.164908	0.150529	0.192498
16	0.076611	-0.338553	0.435520	-0.105110	0.021825	0.339884	0.230447	-0.337374	-0.054222	-0.653484	...	-0.010527	0.264512	-0.244634
17	-0.302741	-0.274373	0.298525	-0.188220	-0.401813	0.141096	-0.093745	-0.366459	-0.809519	-0.130695	...	0.146363	0.046578	-0.353528
18	0.527674	0.629009	-0.384632	0.489437	0.561829	-0.389271	0.212298	0.233610	0.380397	0.387718	...	0.400243	-0.182152	0.287166
19	0.320353	-0.094752	0.120599	0.444076	0.131633	-0.174375	0.417336	0.085567	-0.114620	0.213432	...	0.440008	-0.035431	0.041517
20	-0.479669	-0.452581	0.081423	0.053097	-0.336928	0.405705	0.502963	-0.158576	-0.371150	-0.320810	...	-0.065699	-0.132771	0.109687
21	0.148475	0.161413	-0.386965	-0.059567	0.288114	-0.083627	-0.269380	-0.064146	0.156344	-0.158748	...	-0.110877	0.414004	0.017234
22	0.298212	0.218035	0.142550	0.056818	0.427911	0.125967	0.240729	-0.178068	-0.024533	0.087693	...	1.000000	-0.369970	-0.059085
23	0.022004	-0.323157	-0.167976	0.125137	-0.102514	-0.170035	-0.315245	0.247391	0.151648	-0.236956	...	-0.369970	1.000000	0.069852
24	0.193500	0.388152	-0.479971	0.050242	0.071684	0.165715	0.261003	0.277945	0.257148	0.198970	...	-0.059085	0.069852	1.000000
25	-0.007820	-0.007483	-0.183073	-0.082015	0.042287	-0.463263	-0.589061	0.474793	0.219656	0.278513	...	-0.624774	0.399422	0.221850
26	-0.391960	-0.222688	0.094760	-0.042200	-0.246345	-0.138948	-0.626816	0.290873	0.184472	0.186412	...	-0.701118	0.442418	-0.092619
27	0.478101	0.224718	-0.298575	-0.079141	0.260857	0.029176	0.142038	-0.147543	0.315283	-0.334449	...	-0.193228	0.013597	0.388245
28	-0.658871	0.126030	0.080989	0.141606	-0.326250	0.013689	-0.162874	-0.145856	-0.126491	0.165812	...	-0.079521	-0.058917	-0.264124
29	0.275498	0.268201	-0.458842	0.276326	0.085673	-0.169076	0.393038	0.229722	0.228800	-0.066543	...	-0.320609	0.253746	0.468917
30	-0.166015	-0.014487	0.172647	0.313186	0.004851	0.132724	0.493069	0.120050	-0.127096	0.323513	...	0.285268	-0.297381	0.211759

Figura 1 – Correlação entre os dados.
Fonte: pessoal.

	0	1	2	3	4	5	6	7	8	9
count	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000
mean	95.626953	109.116211	112.750000	127.612305	139.097656	130.491211	142.145508	134.344727	97.023438	135.126953
std	33.615901	56.908917	51.135914	48.141948	59.470162	39.287918	45.671907	59.378414	42.142075	66.366363
min	30.000000	40.000000	40.000000	41.000000	28.000000	48.000000	48.000000	25.000000	24.000000	29.000000
25%	73.000000	56.000000	72.000000	81.750000	88.000000	104.000000	106.000000	79.000000	63.000000	58.500000
50%	88.500000	97.000000	97.000000	142.000000	169.000000	129.000000	159.000000	145.000000	85.000000	169.500000
75%	121.000000	145.000000	168.000000	162.000000	186.000000	150.000000	171.000000	188.750000	134.750000	187.000000
max	162.000000	219.000000	217.000000	217.000000	218.000000	225.000000	220.000000	229.000000	174.000000	222.000000

Figura 2 – Distribuição dos dados.
Fonte: pessoal.

e por final Spectral Clustering.

Para todos os problemas foram selecionado 16 clusters, visto a quantidade de grupos, e de dados que possue o dataset.

Para execução dos algoritmos é utilizado a biblioteca sklearn, onde possui grande parte dos algoritmos de clustering já implementados

2.6 Algoritmos de Clustering K-means dataset completo

Como é possível visualizar abaixo, para o dataset completo o K-means gerou clusters bem esparços com centroides bem centralizados. 4

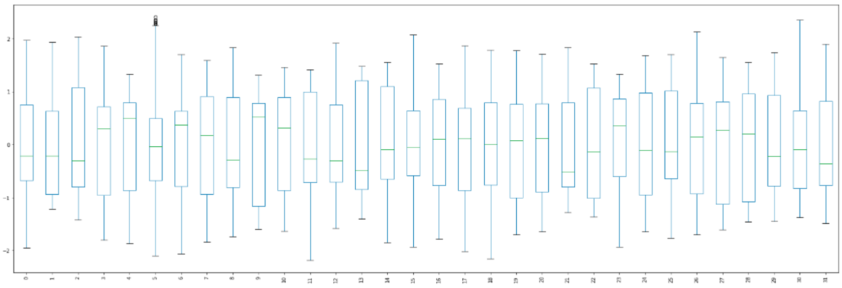


Figura 3 – Bloxpot exibindo outliers.
Fonte: pessoal.

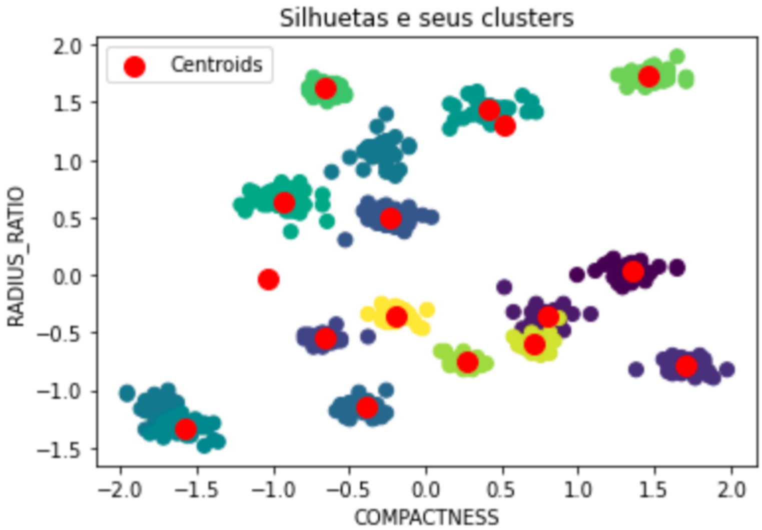


Figura 4 – Algoritmo K-means com dataset completo.
Fonte: pessoal.

2.7 Algoritmos de Clustering Agglomerative Clustering dataset completo

Como é possível visualizar abaixo, para o dataset completo o Agglomerative Clustering gerou clusters onde alguns estão se sobrepondo e com dados entre dois clusters, outros estao bem separados. 5

2.8 Algoritmos de Clustering Spectral Clustering dataset completo

Como é possível visualizar abaixo, para o dataset completo o Spectral Clustering gerou alguns clusters que se sobrepoem e nem sempre estão bem esparcos. 6

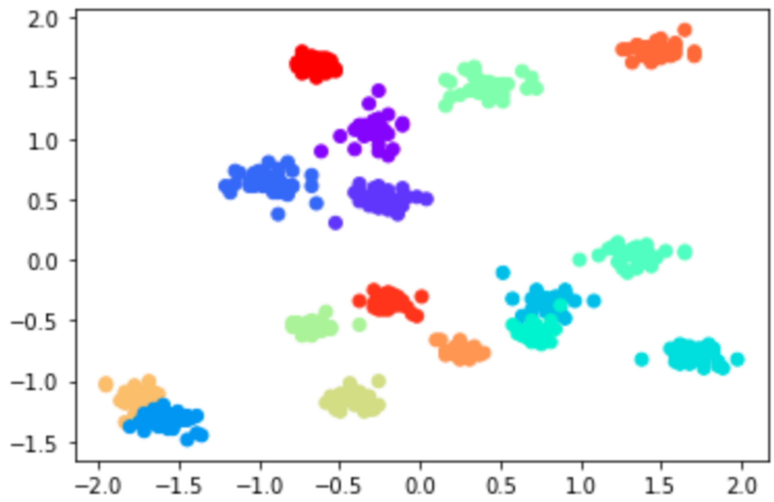


Figura 5 – Algoritmo Agglomerative Clustering com dataset completo.
Fonte: pessoal.

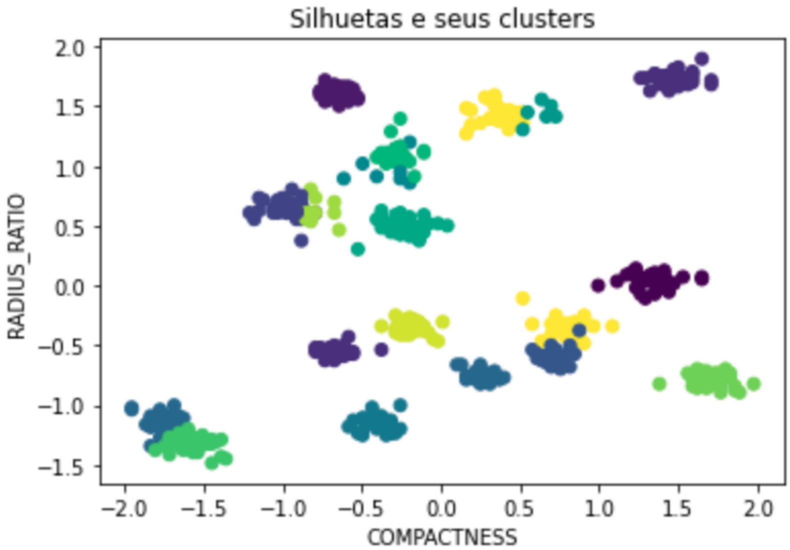


Figura 6 – Algoritmo Spectral Clustering com dataset completo.
Fonte: pessoal.

2.9 Seleção de atributos

Após executar os três algoritmos de clustering com o dataset em questão, foi realizado um processo de seleção de atributos para realizar novamente a execução destes mesmos algoritmos.

O dataset possuia 32 atributos numericos, para realizar a selação foi utilizado um algoritmo que utiliza um parametro k como score para selecionar os melhores atributos.

Neste dataset o algoritmo selecionou apenas 4 atributos

2.10 Algoritmos de Clustering K-means dataset selecionado

Como é possível visualizar abaixo, para o dataset selecionado o K-means gerou clusters bem esparços com centroides bem centralizados. 7

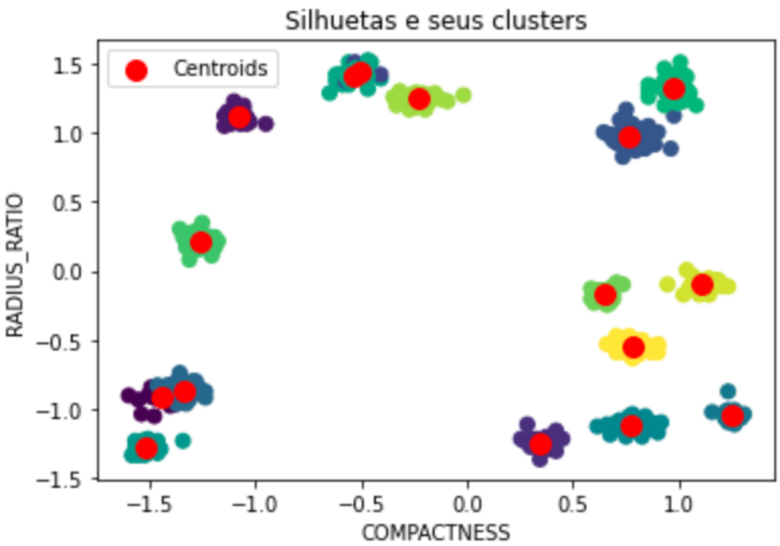


Figura 7 – Algoritmo K-means com dataset selecionado.
Fonte: pessoal.

2.11 Algoritmos de Clustering Agglomerative Clustering dataset selecionado

Como é possível visualizar abaixo, para o dataset selecionado o Agglomerative Clustering gerou clusters onde alguns estão se sobrepondo e com dados entre dois clusters, outros estao bem separados. 8

2.12 Algoritmos de Clustering Spectral Clustering dataset selecionado

Como é possível visualizar abaixo, para o dataset selecionado o Spectral Clustering gerou alguns clusters que se sobrepoem e nem sempre estão bem esparcos. 9

2.13 Análise de resultados

Para analisar os resultados obtidos com os algoritmos foram utilizados 5 métodos de validação, dentre eles:

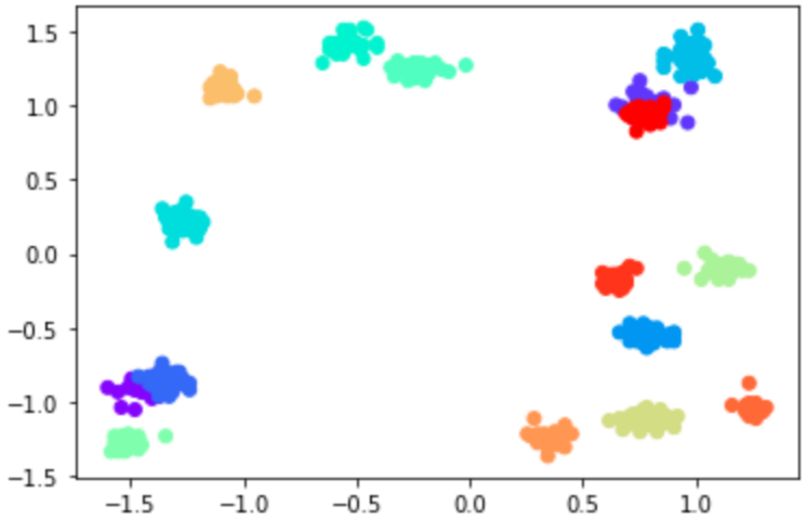


Figura 8 – Algoritmo Agglomerative Clustering com dataset selecionado.
Fonte: pessoal.

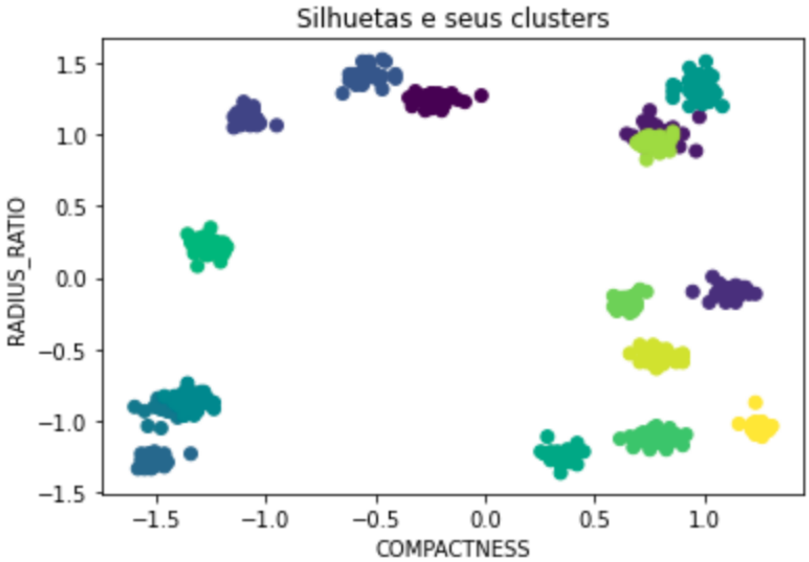


Figura 9 – Algoritmo Spectral Clustering com dataset selecionado.
Fonte: pessoal.

- Confusion Matrix.

  Uma matrix exibindo todos os grupos gerados e seus erros e acertos, onde conta também os erros que devia ter colocado em uma categoria e deveria ser outra. (falsos positivos, falso negativo, verdadeiro positivo, verdadeiro negativo)

  Neste método, os valores na diagonal significam os valores que foram preditos corretamente, neste caso quanto mais numeros na diagonal mais o modelo acertou.

- Calinski-Harabaz Score

  Um indicativo que leva em conta dispersão interna do cluster e também a dispersão entre os clusters, um valor também entre -1 e 1 que quanto mais próximo de 1 melhor é seu resultado.

- Adjusted-Rand Score

  É um indicativo de validação interna do cluster onde o valor possível é entre -1 e 1 que valida quanto os pontos internos do cluster são similares, então quanto mais próximo de 1 melhor os clusters estão definidos.

- Adjusted Mutual Info Score

  É um indicativo de similaridade externa do clusters, neste caso quanto mais próximo de 0 menos similaridade eles possuem, é o esperado visto que cada cluster não deve possuir pontos em comum e devem ser distintos.

- F1 Score

  Um indicativo que utiliza a precissão e o recall do modelo para dizer se ele está errando muito, ou acertando, quanto mais próximo de 1 mais o modelo está correto.

- Accuracy Score

  A acurácia calcula quanto o modelo acertou baseado no total de instancias do dataset. Neste caso quanto mais próximo de 1 mais o modelo está acertando suas predições.

- Silhouette Score

  Um indicativo que também determina quão bem foram classificados os itens dos clusters, quanto maior o indice melhor os clusters foram separados.

2.14 K-Means completo

Como podemos ver na figura exibindo os resultados do K-Means completo, 10 o modelo tendeu a errar para uma classe especifica onde a maioria dos resultados se concentraram em um unico cluster, também podemos perceber um rand score baixo de apenas vinte porcento, onde não é considerado um bom valor. Seu score F1 é apenas de 37 porcento e a acuracia de 50 porcento, onde não demonstrou grandes resultados.

2.15 K-Means selecionado

2.16 Segundo dataset

Após realizar todos os experimentos com o dataset em questão, foi realizado um novo processamento com outro dataset, que possui mais dimensões, o código utilizado foi o mesmo, onde apenas foi alterado o arquivo de dados. Segue abaixo os resultados encontrados.

```
KMeans Completo

Confusion Matrix:
 [[64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 58  0  6  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  2  0  0  0  0  0  0  0  0 62]]
```

Figura 10 – Métricas de validação do k-means
Fonte: pessoal.

```
KMeans Selecionado

Confusion Matrix:
 [[ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  6 58  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]]
```

Figura 11 – Métricas de validação do k-means
Fonte: pessoal.

```
Agglomerative Clustering - Completo

Confusion Matrix:
 [[63  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 63  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  3  0  0  0  0 61  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]]
```

Figura 12 – Métricas de validação do k-means
Fonte: pessoal.

```
Agglomerative Clustering - Selecionado

Confusion Matrix:
 [[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]
 [ 0  0  0  0  0  0  0  0 57  0  0  0  0  0  1  6]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]]
```

Figura 13 – Métricas de validação do k-means
Fonte: pessoal.

2.17 Análise dos dados

2.18 Algoritmos de Clustering Completo

2.19 Algoritmos de Clustering K-means dataset completo

Como é possível visualizar abaixo, para o dataset completo o K-means gerou clusters bem esparços com centroides bem centralizados. 19

```
Spectral Clustering - Completo

Confusion Matrix:
 [[45  0  0  0  0 19  0  0  0  0  0  0  0  0  0  0]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 63  0  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  3  0  0  0 61  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]]
```

Figura 14 – Métricas de validação do k-means
Fonte: pessoal.

```
Spectral Clustering - Selecionado

Confusion Matrix:
 [[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]
 [ 0  0  0  0  0  0  0  0 57  0  0  0  0  0  1  6]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]]
```

Figura 15 – Métricas de validação do k-means
Fonte: pessoal.

2.20 Algoritmos de Clustering Agglomerative Clustering dataset completo

Como é possível visualizar abaixo, para o dataset completo o Agglomerative Clustering gerou clusters onde alguns estão se sobrepondo e com dados entre dois clusters, outros estao bem separados. 20

	0	1	2	3	4	5	6	7	8	9	...
0	1.000000	-0.298556	0.099976	0.022622	0.147617	0.121319	-0.126459	0.467975	0.279398	-0.138266	...
1	-0.298556	1.000000	-0.185263	0.432802	-0.292705	0.061192	-0.019780	0.016283	-0.213111	0.143765	...
2	0.099976	-0.185263	1.000000	0.067625	0.362365	-0.028879	0.080121	0.012888	0.007419	0.235775	...
3	0.022622	0.432802	0.067625	1.000000	0.099326	-0.144059	-0.112927	0.266907	-0.429601	-0.028823	...
4	0.147617	-0.292705	0.362365	0.099326	1.000000	0.293487	-0.131999	-0.170449	0.577666	-0.063625	...
...	...	...	...	...	...	...	...	...	...	...	...
123	0.034045	0.016683	-0.044062	-0.358739	0.033921	0.005289	0.050475	-0.220242	0.284786	-0.384904	...
124	0.463972	-0.548772	0.567061	-0.241791	0.189920	-0.258597	0.129884	0.046233	0.049035	0.115393	...
125	-0.023602	0.217301	-0.184789	0.213103	-0.551513	-0.268762	0.541192	0.318943	-0.447063	0.044297	...
126	-0.188079	0.429549	-0.021832	-0.361639	0.109259	0.058695	0.386794	-0.197502	0.433910	0.145782	...
127	0.311024	-0.073918	0.222362	-0.177917	-0.318805	0.066725	0.368015	0.154835	0.012821	0.257718	...

128 rows × 128 columns

Figura 16 – Correlação entre os dados.
Fonte: pessoal.

	0	1	2	3	4	5	6	7	8	9	...
count	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	...
mean	125.248047	150.040039	134.053711	134.069336	118.694336	145.112305	125.099609	117.110352	108.508789	126.273438	...
std	51.254859	48.465458	49.652222	38.661577	54.941676	44.562082	51.200904	48.900247	51.715931	50.317170	...
min	31.000000	45.000000	42.000000	46.000000	35.000000	65.000000	52.000000	41.000000	31.000000	41.000000	...
25%	89.500000	129.500000	104.500000	100.750000	76.500000	111.250000	66.000000	72.000000	68.000000	89.000000	...
50%	117.000000	145.000000	142.000000	139.500000	111.000000	143.000000	130.000000	116.000000	100.000000	121.500000	...
75%	158.500000	191.000000	174.000000	167.000000	158.000000	180.000000	171.250000	152.250000	137.250000	176.000000	...
max	220.000000	225.000000	205.000000	195.000000	227.000000	218.000000	207.000000	220.000000	207.000000	218.000000	...

8 rows × 128 columns

Figura 17 – Distribuição dos dados.
Fonte: pessoal.

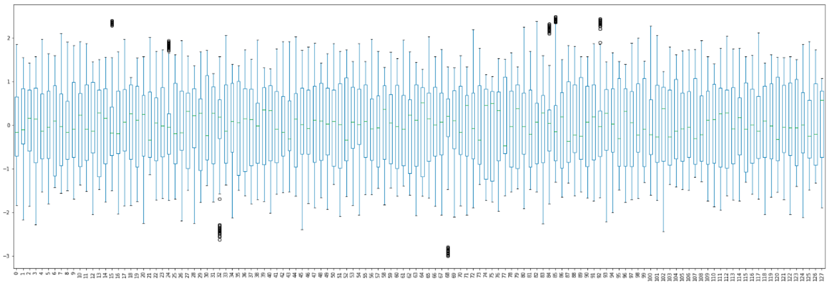


Figura 18 – Bloxpot exibindo outliers.
Fonte: pessoal.

2.21 Algoritmos de Clustering Spectral Clustering dataset completo

Como é possível visualizar abaixo, para o dataset completo o Spectral Clustering gerou alguns clusters que se sobrepoem e nem sempre estão bem esparcos. 6

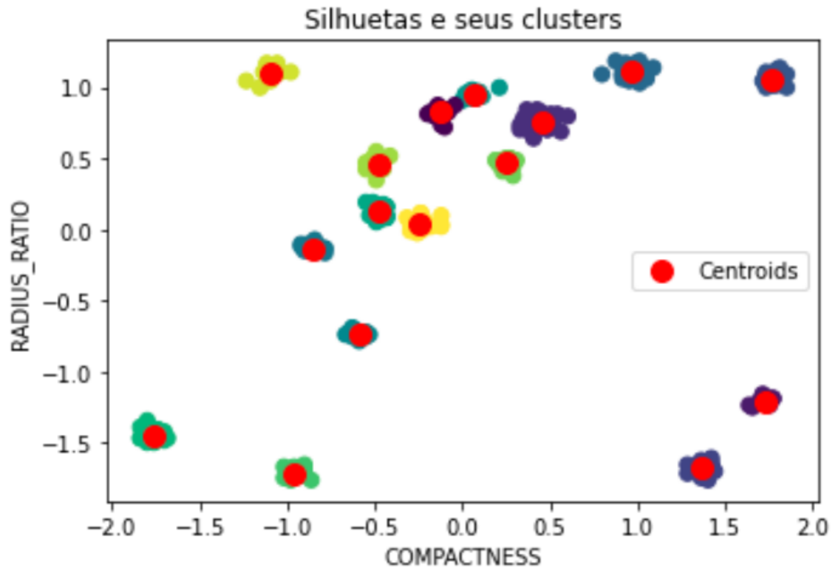


Figura 19 – Algoritmo K-means com dataset completo.
Fonte: pessoal.

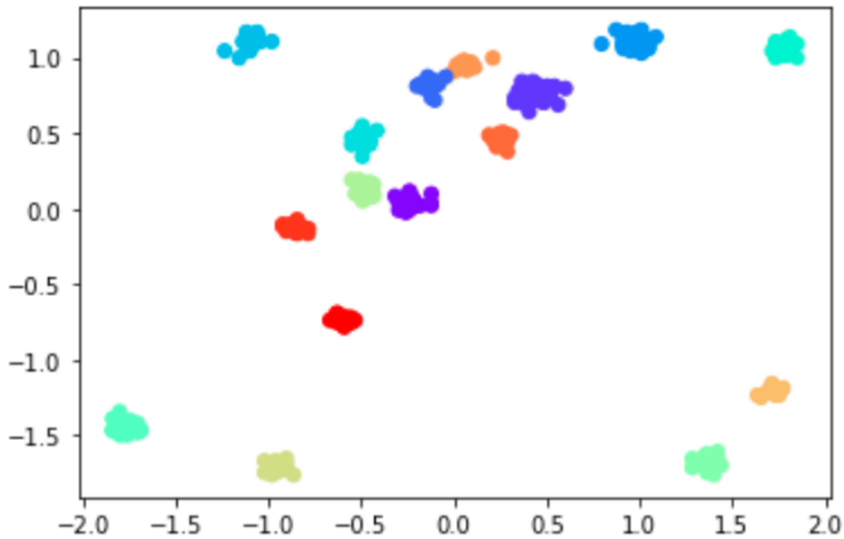


Figura 20 – Algoritmo Agglomerative Clustering com dataset completo.
Fonte: pessoal.

2.22 Algoritmos de Clustering Selecionado

2.23 Algoritmos de Clustering K-means dataset selecionado

Como é possível visualizar abaixo, para o dataset selecionado o K-means gerou clusters bem esparços com centroides bem centralizados. 22

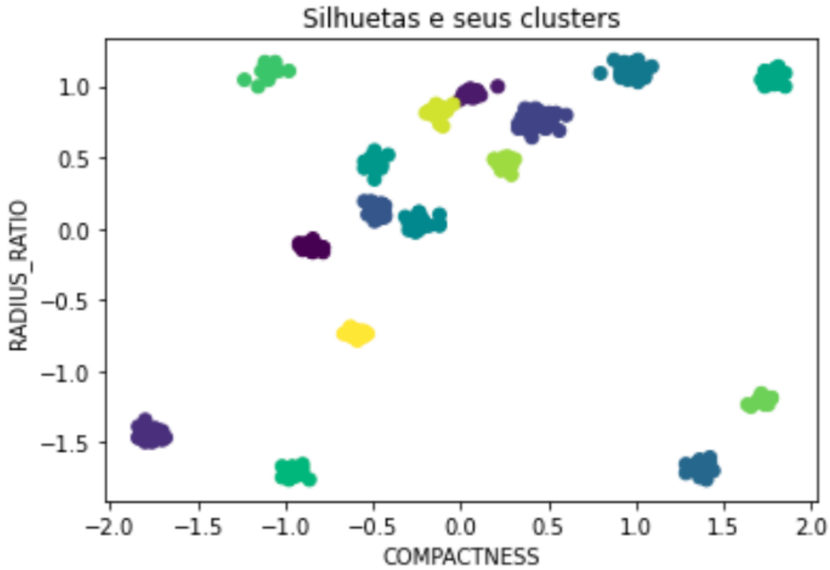


Figura 21 – Algoritmo Spectral Clustering com dataset completo.
Fonte: pessoal.

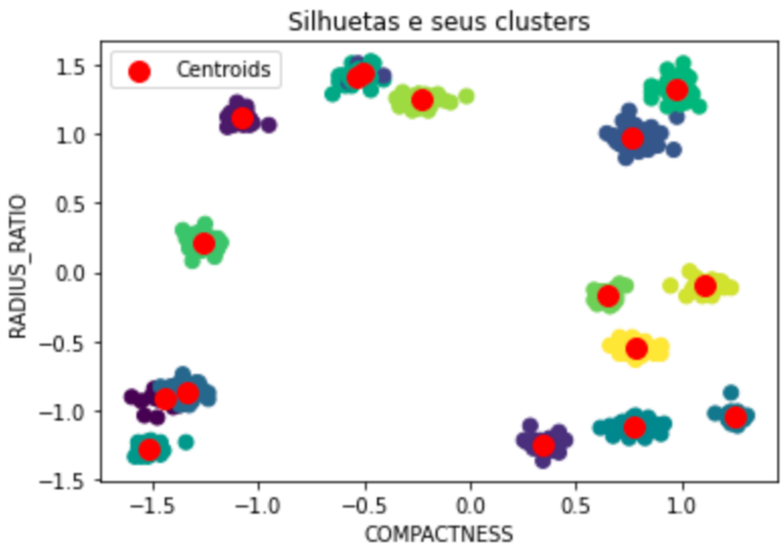


Figura 22 – Algoritmo K-means com dataset selecionado.
Fonte: pessoal.

2.24 Algoritmos de Clustering Agglomerative Clustering dataset selecionado

Como é possível visualizar abaixo, para o dataset selecionado o Agglomerative Clustering gerou clusters onde alguns estão se sobrepondo e com dados entre dois clusters, outros estao bem separados. 23

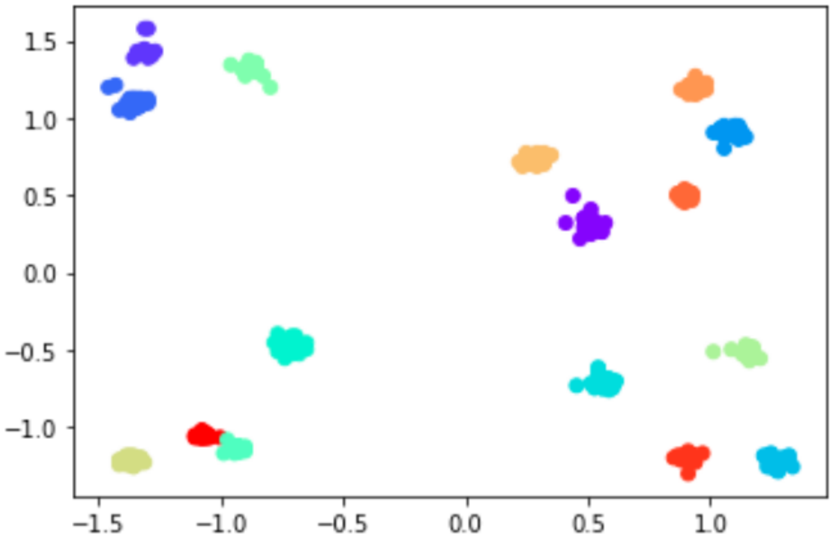


Figura 23 – Algoritmo Agglomerative Clustering com dataset selecionado.
Fonte: pessoal.

2.25 Algoritmos de Clustering Spectral Clustering dataset selecionado

Como é possível visualizar abaixo, para o dataset selecionado o Spectral Clustering gerou alguns clusters que se sobrepoem e nem sempre estão bem esparcos. 24

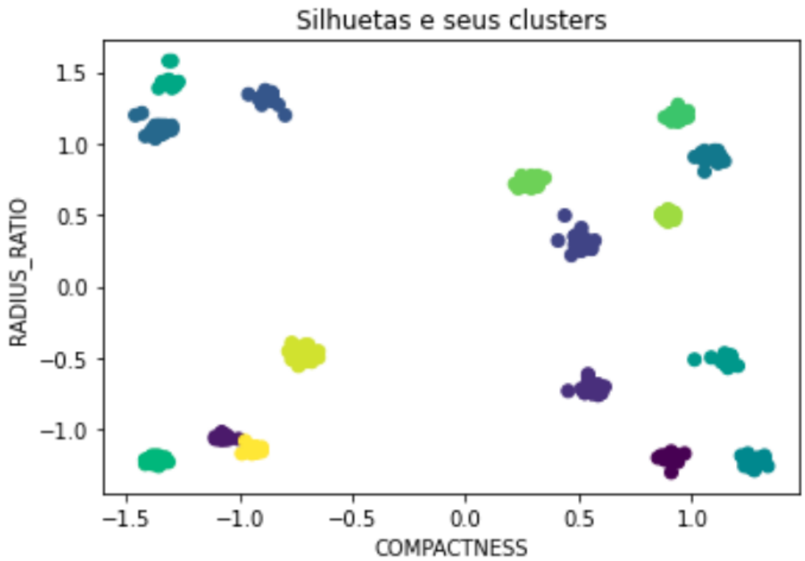


Figura 24 – Algoritmo Spectral Clustering com dataset selecionado.
Fonte: pessoal.

```
KMeans - Completo

Confusion Matrix:
 [[64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]]
```

Figura 25 – Métricas de validação do k-means
Fonte: pessoal.

```
KMeans - Selecionado

Confusion Matrix:
 [[64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0]
 [64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]]
```

Figura 26 – Métricas de validação do k-means
Fonte: pessoal.

```
Agglomerative Clustering - Completo

Confusion Matrix:
 [[64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]]
```

Figura 27 – Métricas de validação do k-means
Fonte: pessoal.

```
Agglomerative Clustering - Selecionado

Confusion Matrix:
 [[64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]]
```

Figura 28 – Métricas de validação do k-means
Fonte: pessoal.

2.26 Análise de resultados

2.27 Resultados

2.28 Primeiro dataset

A figura 31 representa os resultados obtidos com o primeiro dataset, onde é possível visualizar que houve uma melhora dos resultados obtidos com a redução do data set e

```
Spectral Clustering - Completo

Confusion Matrix:
 [[ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0 56  0  0  0  8  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]]
```

Figura 29 – Métricas de validação do k-means
Fonte: pessoal.

```
Spectral Clustering - Selecionado

Confusion Matrix:
 [[64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64]]
```

Figura 30 – Métricas de validação do k-means
Fonte: pessoal.

também que o algoritmo de agglomerative com o dataset completo obteve melhor resultado.

2.29 Segundo dataset

A figura 32 representa os resultados obtidos com o segundo dataset, onde é possível visualizar que houve uma melhora dos resultados obtidos com a redução do data set e

	Cluster	Calinski Harabas	Adjusted-Rand	Adjusted Mutual Info	F1	Accuracy	Silhouette
0	KmeansCompleto	112.0329517361432	0.22264590117920985	0.6984479791750576	0.4500585229306281	0.5	0.31362997777394586
1	KmeansSelecionado	115.67102926026134	0.3275844774111263	0.7374134101430895	0.44652994948653874	0.5107421875	0.2512744586288056
2	AgglomerativeCompleto	155.21737618569244	0.3667464674928476	0.8059812774842965	0.5782167455668188	0.625	0.44679079182198017
3	AgglometariveSelecionado	176.872627934496	0.4125980769947982	0.8016046515297214	0.4721228867638497	0.5556640625	0.47151646969955086
4	SpectralCompleto	85.64255673251895	0.1864278398140855	0.6375583337953774	0.3730326944848341	0.4375	0.22218521892042298
5	SpectralSelecionado	176.872627934496	0.4125980769947982	0.8016046515297214	0.4721228867638497	0.5556640625	0.47151646969955086

Figura 31 – Resultados para todos os clusters encontrados.
Fonte: pessoal.

também que alguns algoritmos conseguiram uma acuracia de 100

	Cluster	Calinski Harabas	Adjusted-Rand	Adjusted Mutual Info	F1	Accuracy	Silhouette
0	KmeansCompleto	665.3427879451361	0.8790351188364668	0.966477564994113	0.8333333333333333	0.875	0.8571602631963691
1	KmeansSelecionado	379.7358691155742	0.7802669341919405	0.9403276294581092	0.7603233262216804	0.8125	0.7838513069493764
2	AgglomerativeCompleto	86413.56951870107	1.0	1.0	1.0	1.0	0.9746405449945033
3	AgglometariveSelecionado	94093.58820563472	1.0	1.0	1.0	1.0	0.9752837913952893
4	SpectralCompleto	86413.56951870107	1.0	1.0	1.0	1.0	0.9746405449945033
5	SpectralSelecionado	94093.58820563472	1.0	1.0	1.0	1.0	0.9752837913952893

Figura 32 – Resultados para todos os clusters encontrados.
Fonte: pessoal.

dim032-clustering

May 26, 2020

1 0. Introdução

Trabalho Clustering:
Aluno: Gabriel Luiz
Disciplina: Tópico em Aprendizado de Máquina
Objetivos :

- Escolha dois datasets rotulados.
- Realize a análise estatística, visualização e pré-processamento dos dados.
- Realize os experimentos criando duas bases de teste distintas:
-
  - considerando todos os atributos do dataset ;
-
  - selecionando alguns atributos e descartando outros;
- Aplique três métodos de clustering distintos nas duas bases acima.
- Para cada dataset , em cada uma das bases, analise os resultados segundo medidas de qualidade de clustering , usando índices de validação interna (SSW, SSB, silhueta, Calinski-Harabasz, Dunn e Davis-Bouldin) e externa (pureza, entropia, acurácia, F-measure , ARI, NMI).
- Proponha uma maneira adicional de comparar os resultados obtidos além das medidas acima.
- Compare e interprete os resultados dos dois experimentos em cada dataset

1.1 0.1 Dependências

Para realização da tarefa foram utilizados as seguintes bibliotecas:

[107]:
```
from datetime import datetime
import numpy as np
import pandas as pd
from sklearn.cluster import *
import seaborn as sns
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.feature_selection import SelectKBest
```

```python
from sklearn.feature_selection import chi2
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import silhouette_score
from sklearn.metrics import calinski_harabasz_score
from sklearn.metrics import adjusted_rand_score
from sklearn.metrics import adjusted_mutual_info_score
from sklearn.metrics.pairwise import euclidean_distances
from scipy.stats import mode
from munkres import Munkres
```

2 1. Dados

Para realização das tarefas envolvidas neste relatório utilizou-se o arquivo **dim032.csv** que contém dados não descritos, onde foram feitos para a realização de clustering que se encontram no site: http://cs.uef.fi/sipu/datasets/

2.1 1.1 Carregamento do arquivo

[108]:
```python
from clustering.labelMatch import rotulos, labelmatch
dataset = './dataset/dim032/dim032.csv'
clusters = './dataset/dim032/dim032-pa.csv'
```

[109]:
```python
data = pd.read_csv(
    dataset,
    header = None
    )

label = pd.read_csv(
    clusters,
    header = None
    )
```

[110]:
```python
data.head()
```

[110]:
```
    0    1    2   3    4    5    6    7   8    9  ...   22  23   24   25  26  \
0  84  152  100  52   95  186  169  106  37  186  ...  190  65  214  116  75
1  86  149  101  56   93  181  171  116  37  192  ...  191  79  215  116  76
2  83  149   99  51   96  187  169  108  34  191  ...  190  65  213  118  73
3  86  142  101  64  105  183  172  116  49  180  ...  186  69  209  120  68
4  89  145  108  54   91  180  175  107  35  192  ...  188  67  212  118  91

   27   28  29   30   31
0  55  123  65  154  177
1  60  130  71  151  181
2  55  125  63  155  178
```

```
3   56   123   67   144   181
4   50   135   58   147   165

[5 rows x 32 columns]
```

[111]:
```
data.describe()
```

[111]:
```
                 0            1            2            3            4  \
count  1024.000000  1024.000000  1024.000000  1024.000000  1024.000000
mean     95.626953   109.116211   112.750000   127.612305   139.097656
std      33.615901    56.908917    51.135914    48.141948    59.470162
min      30.000000    40.000000    40.000000    41.000000    28.000000
25%      73.000000    56.000000    72.000000    81.750000    88.000000
50%      88.500000    97.000000    97.000000   142.000000   169.000000
75%     121.000000   145.000000   168.000000   162.000000   186.000000
max     162.000000   219.000000   217.000000   217.000000   218.000000

                 5            6            7            8            9  ...  \
count  1024.000000  1024.000000  1024.000000  1024.000000  1024.000000  ...
mean    130.491211   142.145508   134.344727    97.023438   135.126953  ...
std      39.287918    45.671907    59.378414    42.142075    66.366363  ...
min      48.000000    48.000000    25.000000    24.000000    29.000000  ...
25%     104.000000   106.000000    79.000000    63.000000    58.500000  ...
50%     129.000000   159.000000   145.000000    85.000000   169.500000  ...
75%     150.000000   171.000000   188.750000   134.750000   187.000000  ...
max     225.000000   220.000000   229.000000   174.000000   222.000000  ...

                22           23           24           25           26  \
count  1024.000000  1024.000000  1024.000000  1024.000000  1024.000000
mean    120.544922   154.849609   123.900391   123.157227   105.608398
std      67.089616    60.070835    58.308579    55.723743    48.049909
min      29.000000    39.000000    28.000000    25.000000    24.000000
25%      53.000000   118.750000    69.000000    87.500000    61.000000
50%     111.500000   176.000000   117.500000   116.000000   113.000000
75%     192.000000   207.000000   181.000000   179.750000   143.250000
max     223.000000   235.000000   222.000000   218.000000   208.000000

                27           28           29           30           31
count  1024.000000  1024.000000  1024.000000  1024.000000  1024.000000
mean    122.179688   130.062500   130.897461   106.218750   116.990234
std      58.800397    61.676195    55.330114    47.630102    55.882102
min      28.000000    40.000000    51.000000    41.000000    34.000000
25%      56.000000    64.000000    88.000000    67.000000    74.000000
50%     138.000000   143.000000   118.500000   102.000000    97.000000
75%     169.750000   189.000000   182.250000   136.750000   162.750000
max     219.000000   226.000000   227.000000   218.000000   223.000000

[8 rows x 32 columns]
```

3 2. Pré-processamento

Validações efetivadas:

- 1. Dados faltantes representados por "NaN"
- 2. Dados que não possuem valores númericos

[112]:
```
data.isna().sum()
```

[112]:
```
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
11    0
12    0
13    0
14    0
15    0
16    0
17    0
18    0
19    0
20    0
21    0
22    0
23    0
24    0
25    0
26    0
27    0
28    0
29    0
30    0
31    0
dtype: int64
```

[113]:
```
for col in data:
    print(col, data[col].unique())
```

```
0 [ 84  86  83  89  85  82  88  92  87  75  90  79  61  68  63  65  64  57
  66  73  62  69  60  67  55  58  56  74 150 153 152 158 154 159 151 148
 149 156 142 157 155 162  91  78  95  97  93 138 140 143 141 144 137 147
```

```
 136 133 135 139 129 145  72  76  71  70  77 124 126 132 119 123 122 125
 121 120 128 113 115 117  37  39  34  35  40  38  36  30  33  41 105 110
 107 108 102 111 112 104 101 106 109 118 114  94  96  43  44  50  47  42
  49  46  48  45 116  80  81 146 103  99 100]
1 [152 149 142 145 154 151 148 150 153 158 162 155 147 159 119 107 113 115
 112 114 110 111 108 118 106 128 116 117 105 104 103 100 101 109 102  51
  50  49  48  52  47  55  46  53  56 200 203 206 205 214 204 207 208 201
 211 212 209 213 215 218 210 216 219  68  75  70  67  69  72  66  80  65
  71  79  73  64  74  62 121 124 125 123 122 126 120  76  77  78  57  59
  58  63  60  61  54  83  84  89  82  88  94  87  85  81  90 141 140 143
 137 139 146 144 134 202 198 199 197  45  44  40]
2 [100 101  99 108  97  94  98 103 105  96 104 106  95 102 109  83  76  75
  67  72  74  79  71  70  73  82  78  80  81 170 172 173 171 176 174 167
 175 164 169 168 178 107  93  91  87  92  88  90  77  69 161 155 158 162
 159 152 157 154 160 165 156 163 116 112 110 177 166 184  44  41  43  42
  48  45  50  47  40  46 210 211 204 212 215 208 214 205 207 213 206 217
 209 203  56  53  54  55  60  52  57  51  49  58  59  63  64  62  65  61
  66  68]
3 [ 52  56  51  64  54  55  53  50  49  62  57  46  48  65  45  41  44 204
 193 198 199 202 201 200 192 196 205 197 207 203 194 206 195 117 119 121
 116 118 122 113 126 120 110 115 125 163 166 165 160 164 157 159 162 161
 145 174 155 154 158 167 168 146 151 148 153 150 149 147 152 139 173 169
 214 217 213 208 211 210 209 215 212 156  58  63  60  59  61  77  78  86
  73  72  74  76  79  71  82  85  75  81  80  84  83  88  91  87  92  89
 170 140 143 142 141 144  90  94 138 137 136 134 133]
4 [ 95  93  96 105  91  97  99  98  94 102 103  92 100  90 207 210 203 205
 198 201 189 199 204 206 208 200 212 202 197 193 209 196 218  35  36  32
  31  34  39  38  28  37  42  33  29  43  40  44 186 185 182 192 181 184
 183 178 180 173 177 179 153 152 149 154 150 151 157 155 146 165 167 171
 164 169 168 166 159 170 176 175 162 163  30 194 190 191 195 188 172 174
  54  55  53  63  59  52  50  49  56  57  62  60  51  64  48  61 111 108
 109 106 110 107 104 112 113  74  77  75  72  69  82  79  70  73  78  76
  81 187]
5 [186 181 187 183 180 185 184 179 188 182 190 193 178 177 131 129 125 126
 120 124 121 127 123 132 128 133 113 118 116 134 119 130 137 135 138 152
 148 150 156 149 159 158 146 151 153 154 155 140 143 147 141 142 163 162
 144 145 115 117 111 114 122 219 216 212 221 220 217 223 215 218 225 222
 214 210 112 104 103 102 106 101  98  99 105 107 109 108 110  94  97  96
 136  56  54  52  55  57  59  53  51  58  60  48  62  90  88  89  91  93
  92  87  85  84  86  83  82  81  95  80  79 175 173 171 172 168 169 174
 170 176 139]
6 [169 171 172 175 170 164 166 167 165 173 168 163 177 131 121 122 115 123
 124 120 119 111 129 116 117 118 127 159 158 157 160 150 153 162 155 156
 154 161 212 214 216 209 208 213 217 200 215 207 220 199 210 205 211 219
 174 178 176 180 181 179 186  82  72  85  80  83  84  77  81  88  91  79
  74  78  87  86  60  63  59  61  62  58  66  48  53  64  56  65  68  70
  69  73  67  75  71  76  55  57 151 152 182 183 184 185 189 187 190 146]
7 [106 116 108 107 109 103 105 112  99 115 111 102 100 114 117 162 164 165
```

```
 172 170 167 160 163 161 148 168 159 166 169 158 157 171  55  58  51  61
  59  60  57  63  53  56  65  52  62  64  80  88  84  85  81  77  76  83
  78  82  79  67  73  94  87  91  90  89  92  86  93 181 180 179 178 184
 183 177 182 195 194 196 200 198 197 205 191 199 202 206 201  34  37  35
  41  32  33  31  38  39  36  25  27  69  68  71 220 222 221 226 223 224
 225 217 216 229 219 218 227 154 156 155 153 149 150 193 136 138 135 139
 134 133 132 131 130 142 137 140 207 204 203 210 208 211 209  74  75 187
 186 176 188]
8 [ 37  34  49  35  38  36  31  43  33  32  39  30  26  40 163 164 167 165
 174 166 160 173 169 170 162 168 161 157 159 118 122 124 130 125 121 123
 119 117 127 120 109  61  62  60  58  63  65  74  59  64  54  68  56  66
  76  71  81  75  77  82  80  78  70  73  79  84  83  72 144 145 142 146
 143 141 147 140 111 114 116 112 115 113 107  67  69 149 150 148 156 151
  97  96  95 100  89  98 104  99  86  94  93 105  92  55  57  24  29 158
 155 171 154 128 129 132 133]
9 [186 192 191 180 187 185 184 188 189 190 194 195 183 200 178 199 203 198
 197 204 193 201 207 202 205  99 100  98  92 101 108  96  95  97 102  94
 103 104 105  39  37  40  29  41  44  36  47  35  38  32  43  42  33 123
 121 120 112 117 119 118 114 113 116 134 115 125 129 127 124 122 219 218
 217 216 222 212 220 221 182 179  51  50  55  56  53  52  48  46  49  45
  54  57 160 158 159 163 156 157 155 153 154 162 152 161 165 196 176 181
 206 208 211 210 209 215 213 214 174 177 175  31  34  30  63  61  59  65
  62  60  64  66  67  72]
10 [140 141 142 147 138 139 143 134 136 135 130 145 146 133 157 144 137  76
  77  80  82  71  75  73  70  74  81  85  79  88  83  34  31  36  32  33
  35  38  29  28  30  42  37  25 181 182 178 180 186 179 183 191 177 187
 185 174 184 173 176 175 170 201 200 197 199 198 202 205 189 207 190 196
 203 195 194  72  84  39  46  40  41  43 193 188  51  48  52  47  49  56
  45  55  44  57  50  54 204 208 211 206 212  78  89  69 156 154 155 159
 152 150 160 161 158 153  93  92 165 167 168 163 169 164 166 162 215 217
 216 213 218 210 214 221]
11 [202 195 198 196 207 200 201 203 206 197 199 204 194 208 205 191 185 178
 186 181 189 190 187 177 184 188 193 183 192 182 179 180 115 114 116 113
 112 109 120 118 128 119 125 117 104 102 106 105 107 108 110 103  94  97
  96  95  92  91  93  85  98 121 126 123 122 124  88  87  82  89  90  84
  86  83 176 175 173 127 129 140 132 130 131 135 133 136 139  44  43  46
  41  49  42  45  47  40  39 111]
12 [ 99 100 102  90 101  97  86 103  98 104  95 107 106  92 111 110 109 105
 108 113 112 114 116  53  52  54  56  51  58  70  50  59  66  55  48  57
  47  49 148 143 147 151 149 150 152 142 154 144 145 153 146 141 139 136
 138 135 137 134 196 197 193 191 194 195 200 205 199 198 189 209 192  44
  45  42  93  94 115  96 170 169 171 178 173 172 176 180 175 168 179 174
  76  77  75  74  72  79  78  71  84  69  73  80  82  83  60  61  62  65
  64 129 130 128 131 127 132 124 126 125 133 119 206 207 203 210 204 208
 202 211 215 212 188 190 184 186  91]
13 [105 106 102  91 100 104 103 107  99 109 110 111  98 112  95 178 161 165
 167 164 172 170 166 162 174 163 168 156 160 169 159 176 158 173 198 196
 200 193 192 188 199 189 197 194 195 190  70  73  64  72  77  75  69  68
```

```
  67  71  56  65  63  74 205 210 202 204 201 203 206 207 208 209  43  46
  48  42  45  47  50  44  51 212 214 213 211  92  93  90  86  89  94  85
  96  87  88  66  59  55  58  54  60  62  57  61  81  82  83  80  84  79
  76  78 157 155 153 154 151 150]
14 [ 88  85  91  90  95  89  84  98  86  87  96  73  93  81  78  94 140 138
 141 137 139 143 151 142 149 136 148 147 135 134 133 144 132 145 150 146
 128 126 125 129 131 130  31  42  36  37  39  38  41  44  35  34  32  40
 208 207 206 205 203 210 209 204 211 199 197 193 196 198 195 194 200 191
 181 183 186 182 176 179 180 188 184 187 185 178 189 177 192 190  49  51
  52  45  50  54  48  53  47  83  82  77  75  92 101 105 106 104 102 108
 107 110 111 103  97  18  33  43  27  28]
15 [114 113 122 110 115 116 124 103 111 117 106 118  99 120  96  72  66  60
  70  65  68  63  69  67  73  64  62  56  71  59  61  30  32  33  40  31
  34  38  39  26  28  35  36  29  24  37  25 167 168 164 165 169 160 170
 166 172 157 171 162 173 179 161 180 119 109 112 108 104 107  97  93  91
  94  92  88  89  95  90  84  87  98 125 121 133 123 126 100 101 105 102
 216 217 220 219 215 222 213 221 223 214 218 210  41  43  42  44  45 130
 128 196 202 198 197 193 200 199 191 203 201 192 195 146 145 150 152 144
 148 143 149 141 147 142 139 140 138 134 132 136 135 137 154 163 156 155
 159]
16 [ 55  50  64  47  54  57  53  56  52  48  51  66  58  61  78  75  73  74
  76  71  72  80  63  67  95  77  81 133 134 137 135 131 127 132 136 130
 138 140 139 129 119 118 117 120 122 114 128 116 112 107 121 126 124 188
 180 179 189 186 176 178 169 183 182 181 177 173 187 184  59 190 185 191
 193 198 215 217 222 214 207 216 213 219 212 208 218 211 196 192 194 195
 108 104 106 110 105  98 109 113 111  97 146 147 142 148 143 150 144 149
 145 152 151 154 153  49  41  44 141 157 160 158 163 159 156 155 162 161
 209 220]
17 [186 188 184 176 183 182 187 185 179 193 189 192 177 195 121 105 109 111
 104 110 115 107 118 112 119 108 106 113 114  93  68  69  70  66  67  71
  75  72  73  60  77  79  74 165 166 168 164 167 161 170 171 160 163 162
 169 153 172 174 173 175  36  33  35  34  38  32  41  37  30  88  85  99
  87  86  83  89  91  84  82  92  80 154 151 150 155 158 159 146 157 156
 152  96  90 103  94  95 198 199 202 196 201 194 197 200 203 204 190 219
 221 220 227 228 223 215 213 222 224 216 218  78  76 124 123 126 122 125
 127 129 120 131 128 116 117]
18 [ 78  83  81  75  80  76  77  79  74  66  73  82  84  91  72  87 158 165
 162 161 160 155 167 156 164 159 157 163 153 166 144 142 146 149 145 143
 141 148 136 147 152 140 151 150 138 108 105 120 106 111 109 112 107 103
 101 110 113 116  98 118 114  96 184 188 190 195 185 187 194 181 186 182
 189 183 177 169 168 170 173 172 132 129 135 133 134 137 131 128 122 130
 124 127 139 102 100  95  99 104  89  85  88  86  92  97  94  41  42  38
  40  44  43  55  46  33  37  45  31  48  39  47  49 180 176 178 179 175
  71  65  67  62  69  68  63  70  64  90  93]
19 [152 145 149 153 151 154 155 147 156 157 136 150 146 158 166 168 169 163
 178 170 171 173 174 167 162 165 164 172 161 176 184 175 191 185 183 186
 181 182 177 189 194 180 190  76  83  80  79  81  78  84  82  77  70  75
  72  73  74  69  71 219 215 218 214 211 213 217 216 207 222 208 209 212
```

108 106 107 111 109 112 105 110 104 113 114 68 64 40 47 46 38 49
48 45 42 43 53 44 58 57 56 52 51 60 55 66 63 59 54 62
61 199 195 192 198 196 193 200 197 202 201 188 159 160 67 123 122 126
124 121 128 125 119 118]
20 [155 154 153 151 156 150 152 160 148 149 159 157 145 163 165 89 85 84
83 96 81 78 86 79 91 88 80 82 87 51 50 52 46 54 53 55
38 47 49 45 56 48 207 214 209 204 203 205 208 211 206 199 210 215
202 200 212 73 77 75 70 76 74 144 146 147 142 125 122 124 128 123
119 126 117 127 114 120 121 132 180 179 181 178 177 175 182 173 188 184
170 185 176 95 101 93 98 97 107 94 102 99 100 137 135 134 138 136
131 139 140 141 133 72 64 63 68 65 67 62 66 61 60 59 57 213
216 58 143 194 190 193 201 195 192 196 197 191]
21 [69 75 68 72 70 73 67 71 63 60 66 64 58 74 76 65 61 198
196 195 201 193 191 200 192 194 202 189 197 186 199 190 51 53 55 47
52 54 59 56 48 50 49 40 39 28 42 44 45 41 43 38 30 37
34 46 220 212 210 211 213 215 206 214 208 218 209 216 207 77 79 78
62 57 219 203 204 217 130 132 126 133 131 135 129 134 128 124 82 80
81 90 84 83 97 100 99 98 96 94 102 101 91 103 109 35 33 31
36 32 146 144 148 150 147 145 149 151 143 141 152 221 222 224 227 223
226]
22 [190 191 186 188 187 189 183 181 182 192 194 184 196 185 180 199 210 207
208 209 213 215 206 204 211 201 216 212 203 222 205 219 202 217 218 214
220 223 60 63 59 62 58 64 65 61 56 50 51 55 57 200 49 52
46 53 47 82 81 78 84 83 85 86 89 80 87 134 133 128 132 135
136 137 131 138 130 140 144 141 139 126 67 66 71 39 37 36 43 33
34 35 38 40 29 45 42 48 44 41 112 114 115 109 113 118 116 121
111 117 176 179 110 107 106 105 108 104 32 31 198 197 195 193]
23 [65 79 69 67 64 66 63 75 62 61 70 71 68 74 76 197 204 203
202 208 201 210 196 207 205 209 200 211 199 213 48 47 44 43 46 42
39 51 45 53 41 50 49 216 215 218 214 212 217 219 235 225 221 224
151 155 157 156 159 149 160 158 161 154 153 152 57 58 60 59 54 222
138 139 136 141 137 132 140 133 134 135 142 144 185 186 183 187 184 181
177 189 188 190 182 167 169 168 163 165 174 170 164 172 166 171 175 178
56 55 176 173 179 180 193 192 194 195 198 191]
24 [214 215 213 209 212 216 206 218 217 211 222 207 208 203 221 123 125 121
122 120 124 126 131 114 118 117 119 127 60 62 63 61 57 53 58 56
59 64 55 52 69 65 175 174 179 177 176 172 178 171 183 173 170 180
188 181 182 184 191 186 190 185 193 194 192 199 195 189 200 197 196 97
98 91 96 99 93 102 95 100 90 107 101 78 84 89 82 83 71 80
88 86 85 87 81 79 75 187 42 43 39 44 38 45 46 48 49 47
37 41 40 33 36 73 74 72 68 76 70 67 35 30 34 29 28 31
32 160 162 158 161 156 157 159 163 167 164 154 165 66 110 113 116 109
112 108 115]
25 [116 118 120 114 119 115 117 125 113 112 110 121 108 109 124 128 122 126
123 130 127 111 67 65 68 66 69 64 60 63 59 70 57 71 61 105
48 49 52 50 53 51 54 41 55 199 197 198 202 201 194 204 200 203
196 165 163 164 169 168 166 167 170 178 162 46 209 207 206 210 212 217
208 211 213 218 216 215 214 205 190 191 188 185 195 189 193 192 187 186

```
 136 137 138 146 140 134 135 142 133 129 145 144 139 141 132 131  36  34
  35  25  33  32  37  39  31  38  28  40  43 101 103 104 102 100  95  97
  93 107  99  98 106  96]
26 [ 75  76  73  68  91  74  77  72  79  67  80  71  78  70  69 113 105 112
 109 110 118 117 111 116 104 114 101 107 106 123 115 103  53  51  54  50
  48  55  52  56  61  49  60  57  45  64  66  63  62  81  65  24  36  35
  33  34  30  31  32  39  37  25  27  28  26  29 124 166 159 163 167 161
 165 160 164 168 170 169 154 162 122 119 121 126 120 125 130 153 156 151
 157 158 155 150 152 196 198 187 195 197 200 199 201 204 202 208 203 192
 190 135 134 133 132 136 131 138 140 137 129 139 141  41  42  44  43  40
  46  38  83  82  84  86  88  92  85  90  87  89  58  59  47]
27 [ 55  60  56  50  54  52  51  47  53  57  58  49  65  59  61  45  48  44
  41  43  40 178 184 182 179 180 183 181 175 190 186 172 185 143 149 141
 133 147 144 142 151 148 138 145 140 150 146 139 135 189 198 197 199 203
 196 200 202 194 193 206 192 201 136 134 120 137 130 128 132 131 129 152
 153 157 163 160 161 164 166 162 158 159 155 154 168 169 212 213 207 215
 211 216 210 219 217 214 209  33  35  39  36  32  28  31  38  34  37  30
  46  42  63  62 195 205 191 116 117 119 115 118 114 109 112 124 111  78
  82  77  76  74  73  86  79  80  75]
28 [123 130 125 135 124 121 126 122 120 131 119 132 133 127 136 209 212 215
 217 222 218 216 214 213 210 220 207 224 226  56  55  54  58  47  57  51
  52  53  59  42  50 160 164 156 159 163 161 154 162 158 155 157 166 150
 170 153  64  71  68  66  69  67  72  70  73  74  63  76  65 202 204 205
 206 203 200 201 208  43  49  40  45  44  48  46  41 211 176 182 188 180
 185 169 174 181 183 184 179 173 178 191  60  61 199 198 197 196 195 194
 192 189 152 151 167 168 171 172 175 165  93  92  94  91  88  90  89  96
  95  98  62  80  78  84  82  81  79  83  85]
29 [ 65  71  63  67  58  64  62  60  66  73  68  55  69  72  57  59  54 106
 111 110 117 107 108 115 116 112 109 104 113 105 114 103 120  98  56  51
  61  53 216 213 217 215 212 218 224 219 207 220 214 227 222 210 195 191
 198 197 186 190 199 202 196 200 194 201 203 204 208 209 206 211 205 149
 148 151 152 150 147 153 146 154  99 102 100 126 123 125 124 121 122 129
 127 128 118  70  74  75  76  95  93  97 101  96  92  94 119 223 221 174
 179 177 175 169 178 176 171 181 180 172 165 132 131 134 140 133 136 130]
30 [154 151 155 144 147 153 152 160 156 158 157 150 177 149 163 142 161 120
 115 119 118 116 125 117 112 124 122 114 126 110 113 101 121  69  68  67
  66  70  60  71  72  74  65  62  73  64 111 108 107 109 102 106 123  98
  76 105 104 183 185 186 184 187 189 195 192 188 191 182 190 193 148 143
  54  52  50  51  47  58  45  46  49  53  56  59  42  55  48  63  79  61
  80  57 209 211 208 212 203 207 200 210 206 215 205 204 218 202 129 128
 130 127 131 133 135 132 134  41  43  97  96  99  95 103  89  91  94]
31 [177 181 178 165 176 179 174 180 167 173 175 189 195 182 168 184 170 151
 147 153 152 150 154 157 148 155 158 162 159 156 149 138 143  72  70  69
  73  74  67  75  76  77  78  71  68 145 144 146 142 141 134 139 140 118
 122 119 121 114 120 116 124 123 113 111 125 117  88  86  84  85  87  93
  89  82  99 103  98  91  97  90  94  96  95  92 100 101  54  61  42  55
  52  53  56  60  51  57  50  46  59 205 196 201 194 199 197 204 193 198
 191 200 190 192 206 188 102  40  48  43  41  36  34  45  44  38  39  37
```

```
 47  83  81  79  80  49 213 216 211 208 212 214 219 210 218 223 215 217
209 207 203]
```

2.1 Conclusão:

- Os dados não possuem a necessidade de pré-processamento visto que já estão todos com valores validos

3.0.1 2.3 Análise estatística

[114]:
```
data.corr()
```

[114]:
```
           0         1         2         3         4         5         6  \
0   1.000000  0.268198 -0.051122 -0.068849  0.599398 -0.438830  0.041834
1   0.268198  1.000000 -0.434193  0.065247  0.147223 -0.087154  0.052960
2  -0.051122 -0.434193  1.000000 -0.212930  0.077845  0.065985 -0.022429
3  -0.068849  0.065247 -0.212930  1.000000 -0.049977 -0.004621  0.348210
4   0.599398  0.147223  0.077845 -0.049977  1.000000 -0.512794 -0.189263
5  -0.438830 -0.087154  0.065985 -0.004621 -0.512794  1.000000  0.549921
6   0.041834  0.052960 -0.022429  0.348210 -0.189263  0.549921  1.000000
7   0.122806  0.112432 -0.216804  0.042810  0.334837 -0.635187 -0.410581
8   0.140755  0.227755 -0.180707  0.093820  0.483320 -0.236149 -0.194464
9   0.017996  0.442574 -0.143382  0.186358  0.257335 -0.517697 -0.334774
10 -0.416168 -0.247372 -0.078198 -0.256024 -0.421847  0.422388  0.085398
11 -0.325456  0.049006 -0.005908 -0.023452 -0.093708  0.000218 -0.346374
12  0.149802 -0.101557 -0.327047  0.109485  0.133371 -0.173939  0.251279
13  0.494533  0.135398 -0.275698  0.165248  0.393054 -0.037131  0.037670
14  0.054404  0.253658  0.027675  0.226054  0.026300  0.018823  0.180490
15 -0.029940 -0.281726 -0.376820 -0.159598 -0.033689 -0.125013  0.108979
16  0.076611 -0.338553  0.435520 -0.105110  0.021825  0.339884  0.230447
17 -0.302741 -0.274373  0.298525 -0.188220 -0.401813  0.141096 -0.093745
18  0.527674  0.629009 -0.384632  0.489437  0.561829 -0.389271  0.212298
19  0.320353 -0.094752  0.120599  0.444076  0.131633 -0.174375  0.417336
20 -0.479669 -0.452581  0.081423  0.053097 -0.336928  0.405705  0.502963
21  0.148475  0.161413 -0.386965 -0.059567  0.288114 -0.083627 -0.269380
22  0.298212  0.218035  0.142550  0.056818  0.427911  0.125967  0.240729
23  0.022004 -0.323157 -0.167976  0.125137 -0.102514 -0.170035 -0.315245
24  0.193500  0.388152 -0.479971  0.050242  0.071684  0.165715  0.261003
25 -0.007820 -0.007483 -0.183073 -0.082015  0.042287 -0.463263 -0.589061
26 -0.391960 -0.222688  0.094760 -0.042200 -0.246345 -0.138948 -0.626816
27  0.478101  0.224718 -0.298575 -0.079141  0.260857  0.029176  0.142038
28 -0.658871  0.126030  0.080989  0.141606 -0.326250  0.013689 -0.162874
29  0.275498  0.268201 -0.458842  0.276326  0.085673 -0.169076  0.393038
30 -0.166015 -0.014487  0.172647  0.313186  0.004851  0.132724  0.493069
31  0.234555  0.132197 -0.408891 -0.219960  0.082626  0.028978  0.155258

           7         8         9  ...        22        23        24        25  \
0   0.122806  0.140755  0.017996  ...  0.298212  0.022004  0.193500 -0.007820
```

1	0.112432	0.227755	0.442574	...	0.218035	-0.323157	0.388152	-0.007483
2	-0.216804	-0.180707	-0.143382	...	0.142550	-0.167976	-0.479971	-0.183073
3	0.042810	0.093820	0.186358	...	0.056818	0.125137	0.050242	-0.082015
4	0.334837	0.483320	0.257335	...	0.427911	-0.102514	0.071684	0.042287
5	-0.635187	-0.236149	-0.517697	...	0.125967	-0.170035	0.165715	-0.463263
6	-0.410581	-0.194464	-0.334774	...	0.240729	-0.315245	0.261003	-0.589061
7	1.000000	0.546772	0.702223	...	-0.178068	0.247391	0.277945	0.474793
8	0.546772	1.000000	0.271853	...	-0.024533	0.151648	0.257148	0.219656
9	0.702223	0.271853	1.000000	...	0.087693	-0.236956	0.198970	0.278513
10	-0.092287	-0.240591	-0.328448	...	-0.033714	0.088643	0.031608	-0.376352
11	0.120182	0.052129	0.429568	...	0.142397	-0.407174	-0.075290	0.217259
12	0.409645	0.260425	0.078683	...	-0.473271	0.256056	0.430193	0.171557
13	0.161548	0.404304	0.132367	...	0.173041	0.090210	0.280425	-0.214557
14	-0.027761	0.383065	-0.032232	...	-0.243979	0.092168	0.168610	0.326534
15	0.221851	-0.183596	-0.171017	...	-0.164908	0.150529	0.192498	-0.069413
16	-0.337374	-0.054222	-0.653484	...	-0.010527	0.264512	-0.244634	-0.393127
17	-0.366459	-0.809519	-0.130695	...	0.146363	0.046578	-0.353528	-0.293029
18	0.233610	0.380397	0.387718	...	0.400243	-0.182152	0.287166	-0.171046
19	0.085567	-0.114620	0.213432	...	0.440008	-0.035431	0.041517	-0.244667
20	-0.158576	-0.371150	-0.320810	...	-0.065699	-0.132771	0.109687	-0.225979
21	-0.064146	0.156344	-0.158748	...	-0.110877	0.414004	0.017234	0.157122
22	-0.178068	-0.024533	0.087693	...	1.000000	-0.369970	-0.059085	-0.624774
23	0.247391	0.151648	-0.236956	...	-0.369970	1.000000	0.069852	0.399422
24	0.277945	0.257148	0.198970	...	-0.059085	0.069852	1.000000	0.221850
25	0.474793	0.219656	0.278513	...	-0.624774	0.399422	0.221850	1.000000
26	0.290873	0.184472	0.186412	...	-0.701118	0.442418	-0.092619	0.698714
27	-0.147543	0.315283	-0.334449	...	-0.193228	0.013597	0.388245	-0.020877
28	-0.145856	-0.126491	0.165812	...	-0.079521	-0.058917	-0.264124	0.123476
29	0.229722	0.228800	-0.066543	...	-0.320609	0.253746	0.468917	0.079350
30	0.120050	-0.127096	0.323513	...	0.285268	-0.297381	0.211759	-0.357671
31	-0.121036	-0.001655	-0.220399	...	-0.163059	0.283402	0.306188	0.149484

	26	27	28	29	30	31
0	-0.391960	0.478101	-0.658871	0.275498	-0.166015	0.234555
1	-0.222688	0.224718	0.126030	0.268201	-0.014487	0.132197
2	0.094760	-0.298575	0.080989	-0.458842	0.172647	-0.408891
3	-0.042200	-0.079141	0.141606	0.276326	0.313186	-0.219960
4	-0.246345	0.260857	-0.326250	0.085673	0.004851	0.082626
5	-0.138948	0.029176	0.013689	-0.169076	0.132724	0.028978
6	-0.626816	0.142038	-0.162874	0.393038	0.493069	0.155258
7	0.290873	-0.147543	-0.145856	0.229722	0.120050	-0.121036
8	0.184472	0.315283	-0.126491	0.228800	-0.127096	-0.001655
9	0.186412	-0.334449	0.165812	-0.066543	0.323513	-0.220399
10	-0.179221	-0.058145	-0.051353	0.065142	-0.060821	-0.216716
11	0.137810	-0.437226	0.259213	-0.526618	-0.172597	-0.439803
12	0.094848	0.344143	-0.423074	0.692584	0.200866	0.511520
13	-0.067142	0.508689	-0.684955	0.144636	-0.021558	0.228797

```
14  0.289699  0.241383  0.249653  0.154281  0.139658  0.214654
15 -0.244122  0.184142 -0.206692  0.347151  0.203000  0.091169
16 -0.020745  0.313308 -0.291156  0.051421  0.019573 -0.032069
17 -0.079840 -0.493866  0.325377 -0.354793  0.128913 -0.114960
18 -0.470166  0.358691 -0.076450  0.453114  0.222688  0.016510
19 -0.368946 -0.257975 -0.060471 -0.023146  0.636369 -0.017978
20 -0.251045 -0.177907  0.147623  0.103378  0.483521 -0.185867
21  0.138043  0.250837 -0.137726  0.282940 -0.534742  0.646400
22 -0.701118 -0.193228 -0.079521 -0.320609  0.285268 -0.163059
23  0.442418  0.013597 -0.058917  0.253746 -0.297381  0.283402
24 -0.092619  0.388245 -0.264124  0.468917  0.211759  0.306188
25  0.698714 -0.020877  0.123476  0.079350 -0.357671  0.149484
26  1.000000 -0.114451  0.229124 -0.243960 -0.226023 -0.060403
27 -0.114451  1.000000 -0.466836  0.431696 -0.227496  0.233082
28  0.229124 -0.466836  1.000000 -0.220608  0.153451 -0.178395
29 -0.243960  0.431696 -0.220608  1.000000 -0.038159  0.414073
30 -0.226023 -0.227496  0.153451 -0.038159  1.000000 -0.191372
31 -0.060403  0.233082 -0.178395  0.414073 -0.191372  1.000000

[32 rows x 32 columns]
```

3.0.2 2.4 Escalonando

Para aplicação dos algoritmos escalona-se os dados afim de parametriza-los num certo intervalor (-1 a 1)

[115]:
```
scaler = preprocessing.StandardScaler()
data_scaler = scaler.fit_transform(X = data)
```

[116]:
```
data_scaler
```

[116]:
```
array([[-0.34604559,  0.75391953, -0.24945736, ..., -1.19156922,
         1.00366357,  1.07438852],
       [-0.28652087,  0.70117796, -0.22989208, ..., -1.08307619,
         0.94064741,  1.14600277],
       [-0.37580795,  0.70117796, -0.26902264, ..., -1.22773356,
         1.02466895,  1.09229209],
       ...,
       [ 0.15991457, -1.00413298,  1.13967774, ...,  0.0199363 ,
        -0.17263801, -0.78758186],
       [ 0.30872638, -0.8810693 ,  1.02228604, ...,  0.00185413,
        -0.17263801, -0.78758186],
       [ 0.27896402, -0.98655246,  1.08098189, ...,  0.0199363 ,
        -0.17263801, -0.75177474]])
```

[117]:
```
data_scaled = pd.DataFrame(data_scaler)
data_scaled.head()
```

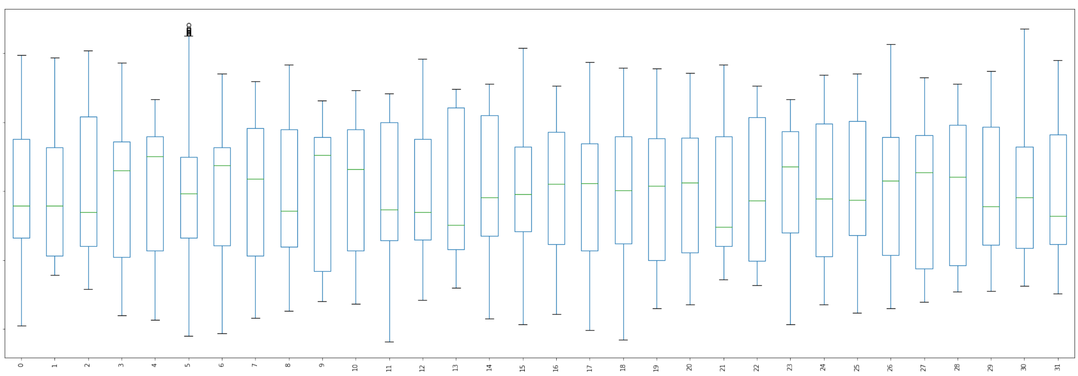
[117]:
```
          0         1         2         3         4         5         6  \
0 -0.346046  0.753920 -0.249457 -1.571379 -0.741871  1.413562  0.588274
1 -0.286521  0.701178 -0.229892 -1.488251 -0.775518  1.286234  0.632086
2 -0.375808  0.701178 -0.269023 -1.592161 -0.725048  1.439028  0.588274
3 -0.286521  0.578114 -0.229892 -1.321994 -0.573638  1.337165  0.653992
4 -0.197234  0.630856 -0.092935 -1.529815 -0.809165  1.260769  0.719710

          7         8         9  ...        22        23        24        25  \
0 -0.477591 -1.425007  0.766923  ...  1.035764 -1.496459  1.545976 -0.128504
1 -0.309097 -1.425007  0.857375  ...  1.050677 -1.263286  1.563134 -0.128504
2 -0.443892 -1.496230  0.842299  ...  1.035764 -1.496459  1.528817 -0.092595
3 -0.309097 -1.140117  0.676472  ...  0.976113 -1.429838  1.460183 -0.056686
4 -0.460741 -1.472489  0.857375  ...  1.005939 -1.463148  1.511658 -0.092595

         26        27        28        29        30        31
0 -0.637324 -1.143062 -0.114565 -1.191569  1.003664  1.074389
1 -0.616502 -1.057987 -0.001014 -1.083076  0.940647  1.146003
2 -0.678968 -1.143062 -0.082122 -1.227734  1.024669  1.092292
3 -0.783077 -1.126047 -0.114565 -1.155405  0.793610  1.146003
4 -0.304174 -1.228137  0.080094 -1.318144  0.856626  0.859546

[5 rows x 32 columns]
```

3.0.3 2.5 Plotando boxsplot

Pelo boxsplot é possivel visualizar que há alguns outliers.

[118]:
```
data_scaled.plot(kind = 'box', figsize=(30,10), rot=90, )
```

[118]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f20cb4775f8>
```



4 3. Clustering

4.1 3.1 Dataset Completo

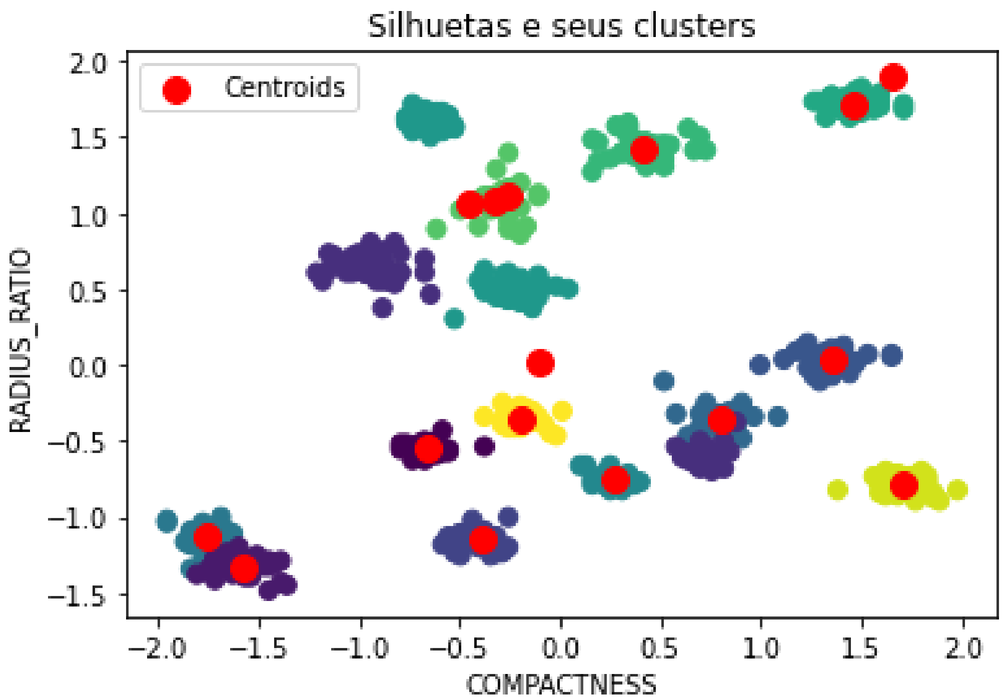
4.1.1 3.1.1 K-Means

[119]:
```
data_kmeans = data_scaled.copy()
```

[120]:
```
kmeans = KMeans(n_clusters = 16, init = 'random')
kmeans.fit(data_kmeans)
```

[120]:
```
KMeans(algorithm='auto', copy_x=True, init='random', max_iter=300,
       n_clusters=16, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

[121]:
```
plt.scatter(data_scaler[:,0], data_scaler[:,31], s = 50, c = kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 31], s =␣
 ↪100, c = 'red',label = 'Centroids')
plt.title('Silhuetas e seus clusters')
plt.xlabel('COMPACTNESS')
plt.ylabel('RADIUS_RATIO')
plt.legend()
plt.show()
```



4.1.2 3.1.2 Agglomerative Clustering

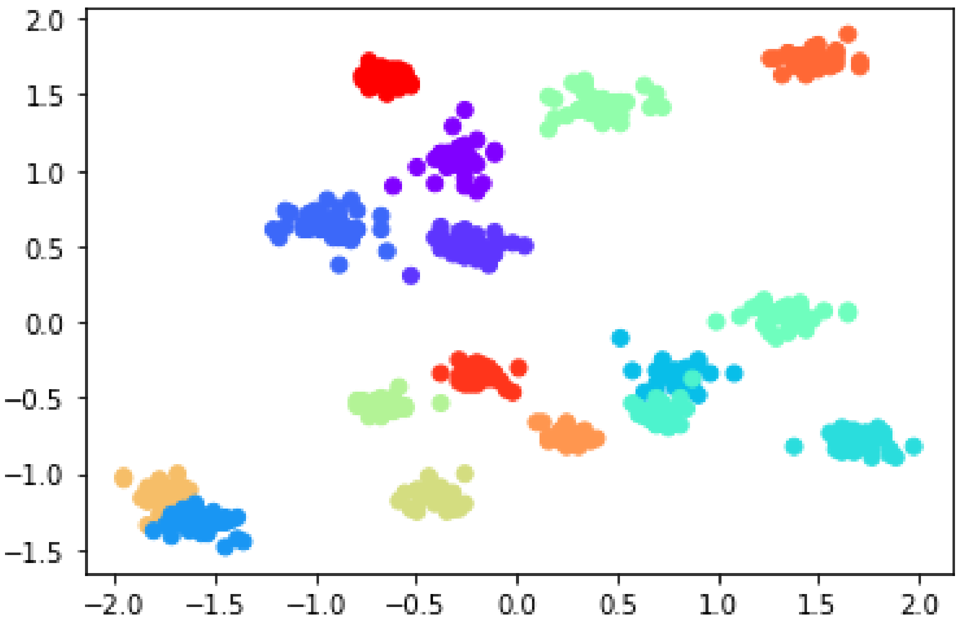
[122]:
```
data_agglo = data_scaled.copy()
```

[123]:
```
agglo = AgglomerativeClustering(n_clusters=16, linkage='ward')
agglo.fit(data_agglo)
```

[123]:
```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
                        connectivity=None, distance_threshold=None,
                        linkage='ward', memory=None, n_clusters=16)
```

[124]:
```
plt.scatter(data_scaler[:,0],data_scaler[:,31], c=agglo.labels_, cmap='rainbow')
```

[124]:
```
<matplotlib.collections.PathCollection at 0x7f20cc8ae8d0>
```



4.1.3 3.1.3 Spectral Clustering

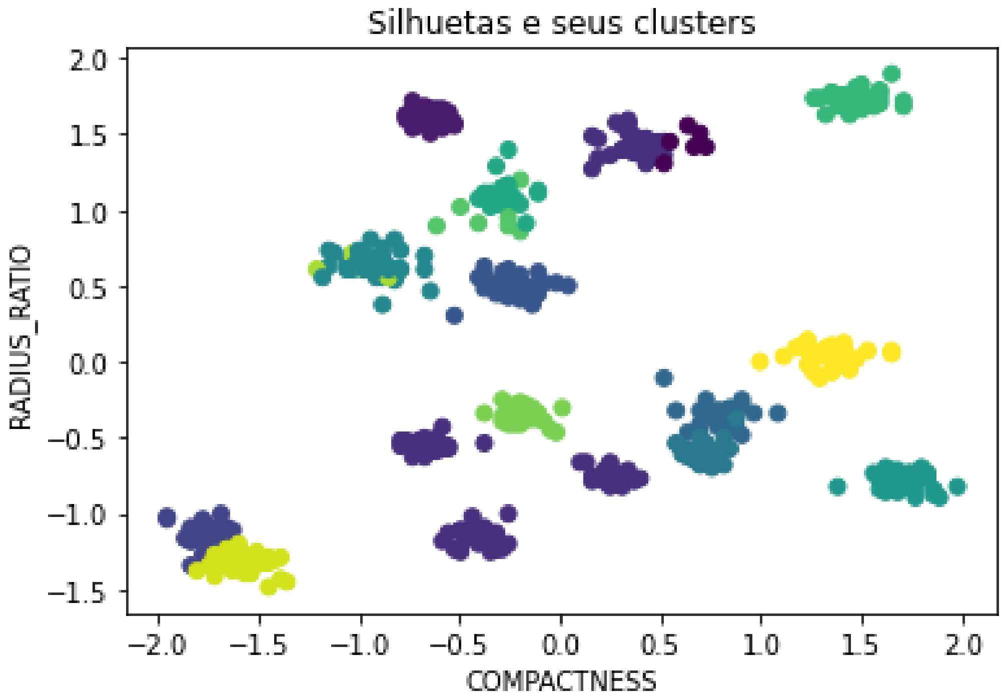
[125]:
```
data_spectral = data_scaled.copy()
```

[126]:
```
spectral = SpectralClustering(n_clusters=16)
spectral.fit(data_spectral)
```

[126]:
```
SpectralClustering(affinity='rbf', assign_labels='kmeans', coef0=1, degree=3,
                   eigen_solver=None, eigen_tol=0.0, gamma=1.0,
                   kernel_params=None, n_clusters=16, n_components=None,
                   n_init=10, n_jobs=None, n_neighbors=10, random_state=None)
```

[127]:
```
plt.scatter(data_scaler[:,0], data_scaler[:,31], c = spectral.labels_)
plt.title('Silhuetas e seus clusters')
```

```
plt.xlabel('COMPACTNESS')
plt.ylabel('RADIUS_RATIO')
plt.show()
```



4.2 3.2 Dataset com atributos selecionados

[128]:
```
data_reduzida = pd.DataFrame(SelectKBest(chi2, k=4).fit_transform(data, label))
data_reduzida.shape

data_scaler2 = scaler.fit_transform(X = data_reduzida)
```

[129]:
```
data_scaler2
```

[129]:
```
array([[ 0.76692323,  0.18389215, -0.63955056,  1.03576412],
       [ 0.85737465,  0.19968827, -0.54581938,  1.05067684],
       [ 0.84229941,  0.21548439, -0.65517242,  1.03576412],
       ...,
       [-1.0873309 ,  1.38439736, -0.67079429,  1.09541499],
       [-1.04210519,  1.40019348, -0.65517242,  1.11032771],
       [-1.0873309 ,  1.36860124, -0.67079429,  1.12524043]])
```

[130]:
```
data_scaled2 = pd.DataFrame(data_scaler2)
data_scaled2.head()
```

[130]:
	0	1	2	3
0	0.766923	0.183892	-0.639551	1.035764
1	0.857375	0.199688	-0.545819	1.050677
2	0.842299	0.215484	-0.655172	1.035764
3	0.676472	0.294465	-0.592685	0.976113
4	0.857375	0.215484	-0.545819	1.005939

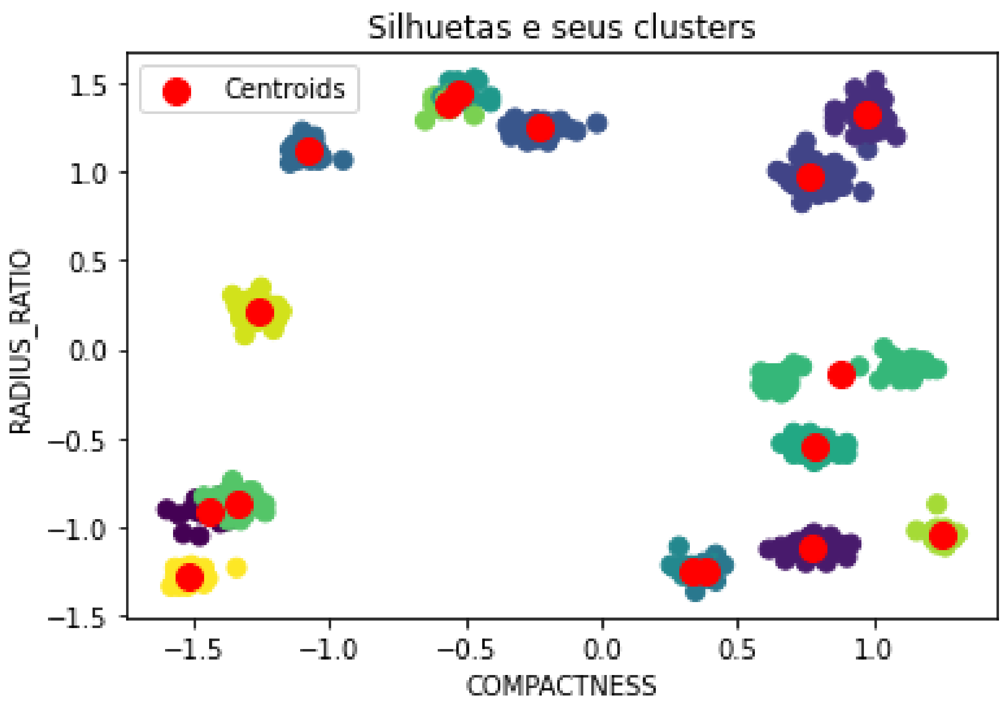
4.2.1 3.2.1 K-Means

[131]:
```
data_kmeans2 = data_scaled2.copy()
```

[132]:
```
kmeans2 = KMeans(n_clusters = 16, init = 'random')
kmeans2.fit(data_kmeans2)
```

[132]:
```
KMeans(algorithm='auto', copy_x=True, init='random', max_iter=300,
       n_clusters=16, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

[133]:
```
plt.scatter(data_scaler2[:,0], data_scaler2[:,3], s = 50, c = kmeans2.labels_)
plt.scatter(kmeans2.cluster_centers_[:, 0], kmeans2.cluster_centers_[:, 3], s =␣
 ↪100, c = 'red',label = 'Centroids')
plt.title('Silhuetas e seus clusters')
plt.xlabel('COMPACTNESS')
plt.ylabel('RADIUS_RATIO')
plt.legend()
plt.show()
```



4.2.2 3.2.2 Agglomerative Clustering

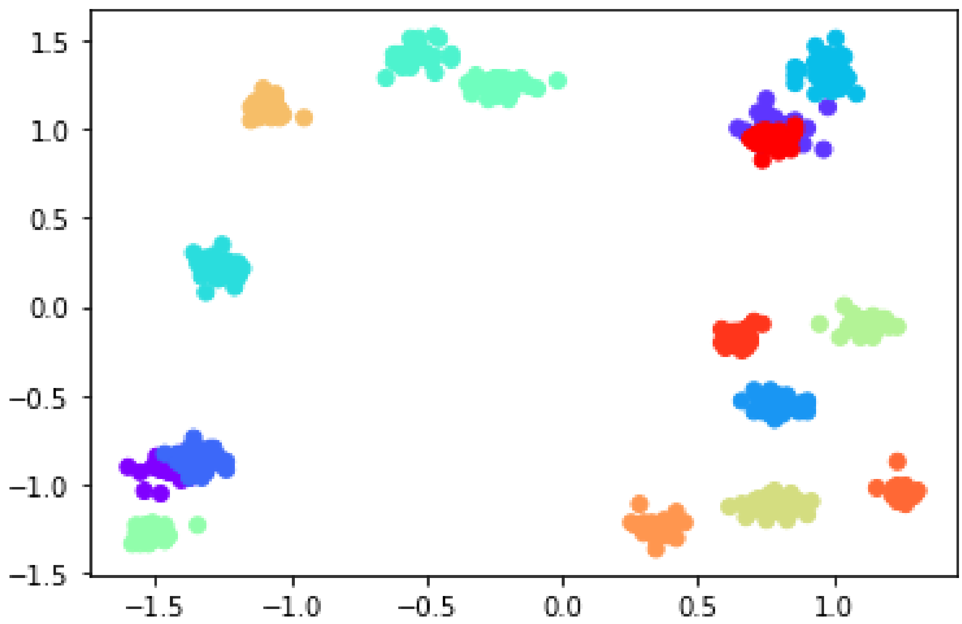
[134]:
```
data_agglo2 = data_scaled2.copy()
```

[135]:
```
agglo2 = AgglomerativeClustering(n_clusters=16, linkage='ward')
agglo2.fit(data_agglo2)
```

[135]:
```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
                        connectivity=None, distance_threshold=None,
                        linkage='ward', memory=None, n_clusters=16)
```

[136]:
```
plt.scatter(data_scaler2[:,0],data_scaler2[:,3], c=agglo2.labels_,␣
↪cmap='rainbow')
```

[136]:
```
<matplotlib.collections.PathCollection at 0x7f20cc904be0>
```



4.2.3 3.2.3

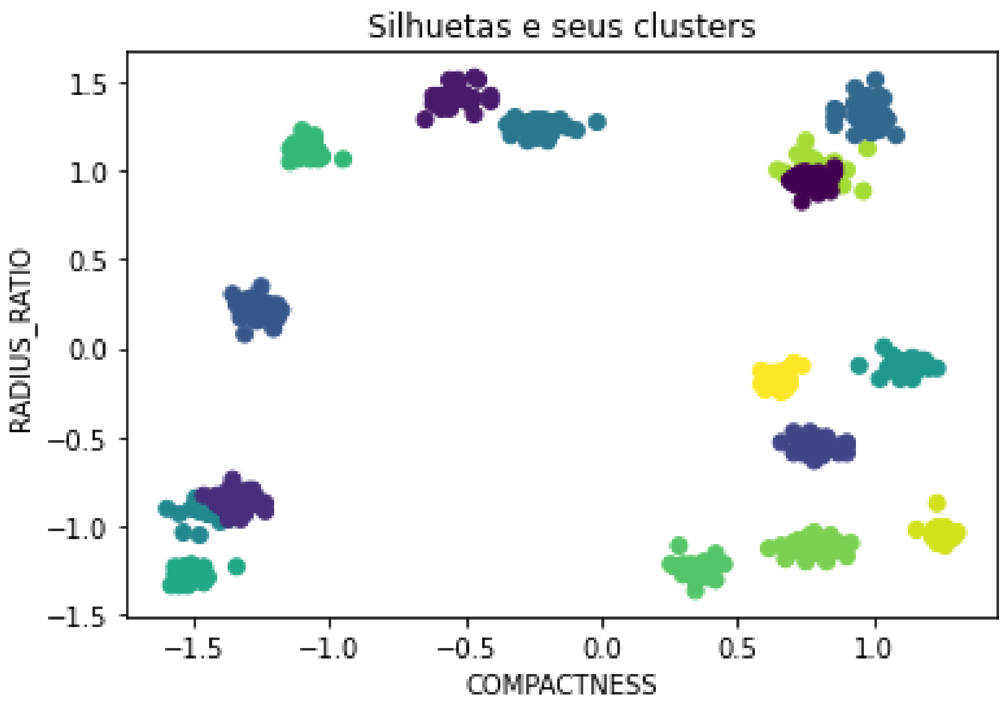
[137]:
```
data_spectral2 = data_scaled2.copy()
```

[138]:
```
spectral2 = SpectralClustering(n_clusters=16)
spectral2.fit(data_spectral2)
```

[138]:
```
SpectralClustering(affinity='rbf', assign_labels='kmeans', coef0=1, degree=3,
                   eigen_solver=None, eigen_tol=0.0, gamma=1.0,
```

```
                         kernel_params=None, n_clusters=16, n_components=None,
                         n_init=10, n_jobs=None, n_neighbors=10, random_state=None)
```

[139]:
```
plt.scatter(data_scaler2[:,0], data_scaler2[:,3], c = spectral2.labels_)
plt.title('Silhuetas e seus clusters')
plt.xlabel('COMPACTNESS')
plt.ylabel('RADIUS_RATIO')
plt.show()
```



5 4. Avaliação

[140]:
```
lista = np.array(label[0].tolist())
```

[141]:
```
for i in lista:
    lista[i] = lista[i] - 1
```

5.0.1 4.1.1 KMeans - Completo

[142]:
```
dataset = data.values

class Data:
    namostras = 0
    ndim = 0
```

```
    ncluster = 0

newData = Data()

newData.namostras = len(data)
newData.ndim = len(data.columns)
newData.ncluster = 16


labels_true = lista

# predict recebe os rotulos preditos pelo algoritmo de clustering
predict = rotulos(kmeans.cluster_centers_, 16, dataset, newData)
```

[143]:
```
# labels_predict sao as labels ja organizadas para comparacao correta com os␣
↪rotulos originais do conjunto de dados
labels_predict = labelmatch(labels_true,predict,newData.ncluster)
```

[144]:
```
# METRICAS PARA AVALIACAO DO CLUSTERING
cft = confusion_matrix(labels_true, labels_predict)
hbt = calinski_harabasz_score(dataset,labels_predict)
arit = adjusted_rand_score(labels_true, labels_predict)
amit = adjusted_mutual_info_score(labels_true, labels_predict)
f1t = f1_score(labels_true, labels_predict, average='macro')
accurracyt =accuracy_score(labels_true, labels_predict)
silhouettet = silhouette_score(dataset, labels_predict)

print('Confusion Matrix: \n', cft)
print('\nCalinski-Harabaz Score: ',hbt)
print('\nAdjusted-Rand Score: ',arit)
print('\nAdjusted Mutual Info Score: ',amit)
print('\nF1 Score: ',f1t)
print('\nAccuracy Score: ',accurracyt)
print('\nSilhouette Score: ',silhouettet)
```

```
Confusion Matrix:
 [[ 0  0  0 14  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 47  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 46  0  0  0  0 18  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0]
```

```
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]]

Calinski-Harabaz Score:  140.74286421202137

Adjusted-Rand Score:  0.32176406259196033

Adjusted Mutual Info Score:  0.7351855997173125

F1 Score:  0.34633076830541293

Accuracy Score:  0.4833984375

Silhouette Score:  0.4107273087344685
```

5.0.2 4.1.2 KMeans - Selecionado

[145]:
```python
dataset = data_reduzida.values

class Data:
    namostras = 0
    ndim = 0
    ncluster = 0

newData = Data()

newData.namostras = len(data_reduzida)
newData.ndim = len(data_reduzida.columns)
newData.ncluster = 16


labels_true = lista
```

[146]:
```python
# predict recebe os rotulos preditos pelo algoritmo de clustering
predict = rotulos(kmeans2.cluster_centers_, 16, dataset, newData)

# labels_predict sao as labels ja organizadas para comparacao correta com os
 ↪rotulos originais do conjunto de dados
labels_predict = labelmatch(labels_true,predict,newData.ncluster)

# METRICAS PARA AVALIACAO DO CLUSTERING
cft = confusion_matrix(labels_true, labels_predict)
hbt = calinski_harabasz_score(dataset,labels_predict)
```

```python
arit = adjusted_rand_score(labels_true, labels_predict)
amit = adjusted_mutual_info_score(labels_true, labels_predict)
f1t = f1_score(labels_true, labels_predict, average='macro')
accurracyt =accuracy_score(labels_true, labels_predict)
silhouettet = silhouette_score(dataset, labels_predict)

print('Confusion Matrix: \n', cft)
print('\nCalinski-Harabaz Score: ',hbt)
print('\nAdjusted-Rand Score: ',arit)
print('\nAdjusted Mutual Info Score: ',amit)
print('\nF1 Score: ',f1t)
print('\nAccuracy Score: ',accurracyt)
print('\nSilhouette Score: ',silhouettet)
```

```
Confusion Matrix:
 [[ 0  0  0  0  0  0 15  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 47  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]]

Calinski-Harabaz Score:  307.79807227114503

Adjusted-Rand Score:  0.10681904679356016

Adjusted Mutual Info Score:  0.3771666990795962

F1 Score:  0.02460697197539303

Accuracy Score:  0.125

Silhouette Score:  0.2626307993848876
```

5.0.3 4.2.1 Agglomerative Clustering - Completo

[147]:
```
def centroide(data):
    array2 = []
    for valor in range(0,16):
        df_aux = data.loc[data.Label == valor]
        array = []
        for coluna in df_aux:
            array.append(df_aux[coluna].mean())

        array2.append(array)

    return np.array(array2)
```

[148]:
```
data_agglo['Label'] = agglo.labels_
```

[149]:
```
centroide_hieraquico = centroide(data_agglo)
```

[150]:
```
dataset = data.values

class Data:
    namostras = 0
    ndim = 0
    ncluster = 0

newData = Data()

newData.namostras = len(data)
newData.ndim = len(data.columns)
newData.ncluster = 16


labels_true = lista

# predict recebe os rotulos preditos pelo algoritmo de clustering
predict = rotulos(centroide_hieraquico, 16, dataset, newData)

# labels_predict sao as labels ja organizadas para comparacao correta com os␣
↪rotulos originais do conjunto de dados
labels_predict = labelmatch(labels_true,predict,newData.ncluster)


# METRICAS PARA AVALIACAO DO CLUSTERING
cft = confusion_matrix(labels_true, labels_predict)
hbt = calinski_harabasz_score(dataset,labels_predict)
arit = adjusted_rand_score(labels_true, labels_predict)
amit = adjusted_mutual_info_score(labels_true, labels_predict)
f1t = f1_score(labels_true, labels_predict, average='macro')
accurracyt =accuracy_score(labels_true, labels_predict)
```

```
silhouettet = silhouette_score(dataset, labels_predict)

print('Confusion Matrix: \n', cft)
print('\nCalinski-Harabaz Score: ',hbt)
print('\nAdjusted-Rand Score: ',arit)
print('\nAdjusted Mutual Info Score: ',amit)
print('\nF1 Score: ',f1t)
print('\nAccuracy Score: ',accurracyt)
print('\nSilhouette Score: ',silhouettet)
```

```
Confusion Matrix:
 [[ 0  0  0 14  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 47  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 63  0  0  0  0  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  3 61  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]]

Calinski-Harabaz Score:  155.21737618569244

Adjusted-Rand Score:  0.3591399518886638

Adjusted Mutual Info Score:  0.798187694021631

F1 Score:  0.4796782841769782

Accuracy Score:  0.609375

Silhouette Score:  0.44679079182198017
```

5.0.4 4.2.2 Agglomerative Clustering - Selecionado

[151]:
```
data_agglo2['Label'] = agglo2.labels_
data_agglo2.head()
```

[151]:

	0	1	2	3	Label
0	0.766923	0.183892	-0.639551	1.035764	1
1	0.857375	0.199688	-0.545819	1.050677	1
2	0.842299	0.215484	-0.655172	1.035764	1
3	0.676472	0.294465	-0.592685	0.976113	1
4	0.857375	0.215484	-0.545819	1.005939	1

[152]:
```
centroide_hieraquico2 = centroide(data_agglo2)
```

[153]:
```
dataset = data_reduzida.values

class Data:
    namostras = 0
    ndim = 0
    ncluster = 0

newData = Data()

newData.namostras = len(data_reduzida)
newData.ndim = len(data_reduzida.columns)
newData.ncluster = 16


labels_true = lista

# predict recebe os rotulos preditos pelo algoritmo de clustering
predict = rotulos(centroide_hieraquico2, 16, dataset, newData)

# labels_predict sao as labels ja organizadas para comparacao correta com os␣
↪rotulos originais do conjunto de dados
labels_predict = labelmatch(labels_true,predict,newData.ncluster)


# METRICAS PARA AVALIACAO DO CLUSTERING
cft = confusion_matrix(labels_true, labels_predict)
hbt = calinski_harabasz_score(dataset,labels_predict)
arit = adjusted_rand_score(labels_true, labels_predict)
amit = adjusted_mutual_info_score(labels_true, labels_predict)
f1t = f1_score(labels_true, labels_predict, average='macro')
accurracyt =accuracy_score(labels_true, labels_predict)
silhouettet = silhouette_score(dataset, labels_predict)

print('Confusion Matrix: \n', cft)
print('\nCalinski-Harabaz Score: ',hbt)
```

```
print('\nAdjusted-Rand Score: ',arit)
print('\nAdjusted Mutual Info Score: ',amit)
print('\nF1 Score: ',f1t)
print('\nAccuracy Score: ',accurracyt)
print('\nSilhouette Score: ',silhouettet)
```

```
Confusion Matrix:
 [[ 0  0  0  0  0  0 15  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 47  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]]

Calinski-Harabaz Score:  307.79807227114503

Adjusted-Rand Score:  0.10681904679356016

Adjusted Mutual Info Score:  0.3771666990795962

F1 Score:  0.02460697197539303

Accuracy Score:  0.125

Silhouette Score:  0.2626307993848876
```

5.0.5 4.3.1 Spectral Clustering - Completo

[154]:
```
data_spectral['Label'] = spectral.labels_
data_spectral.head()
```

[154]:
```
          0         1         2         3         4         5         6  \
0 -0.346046  0.753920 -0.249457 -1.571379 -0.741871  1.413562  0.588274
1 -0.286521  0.701178 -0.229892 -1.488251 -0.775518  1.286234  0.632086
```

```
2 -0.375808  0.701178 -0.269023 -1.592161 -0.725048  1.439028  0.588274
3 -0.286521  0.578114 -0.229892 -1.321994 -0.573638  1.337165  0.653992
4 -0.197234  0.630856 -0.092935 -1.529815 -0.809165  1.260769  0.719710

          7         8         9  ...        23        24        25        26  \
0 -0.477591 -1.425007  0.766923  ... -1.496459  1.545976 -0.128504 -0.637324
1 -0.309097 -1.425007  0.857375  ... -1.263286  1.563134 -0.128504 -0.616502
2 -0.443892 -1.496230  0.842299  ... -1.496459  1.528817 -0.092595 -0.678968
3 -0.309097 -1.140117  0.676472  ... -1.429838  1.460183 -0.056686 -0.783077
4 -0.460741 -1.472489  0.857375  ... -1.463148  1.511658 -0.092595 -0.304174

         27        28        29        30        31  Label
0 -1.143062 -0.114565 -1.191569  1.003664  1.074389      9
1 -1.057987 -0.001014 -1.083076  0.940647  1.146003     11
2 -1.143062 -0.082122 -1.227734  1.024669  1.092292      9
3 -1.126047 -0.114565 -1.155405  0.793610  1.146003      9
4 -1.228137  0.080094 -1.318144  0.856626  0.859546     11

[5 rows x 33 columns]
```

[155]:
```
centroide_spectral = centroide(data_spectral)
```

[156]:
```
dataset = data.values

class Data:
    namostras = 0
    ndim = 0
    ncluster = 0

newData = Data()

newData.namostras = len(data)
newData.ndim = len(data.columns)
newData.ncluster = 16


labels_true = lista

# predict recebe os rotulos preditos pelo algoritmo de clustering
predict = rotulos(centroide_spectral, 16, dataset, newData)

# labels_predict sao as labels ja organizadas para comparacao correta com os␣
 ↪rotulos originais do conjunto de dados
labels_predict = labelmatch(labels_true,predict,newData.ncluster)


# METRICAS PARA AVALIACAO DO CLUSTERING
cft = confusion_matrix(labels_true, labels_predict)
```

```python
hbt = calinski_harabasz_score(dataset,labels_predict)
arit = adjusted_rand_score(labels_true, labels_predict)
amit = adjusted_mutual_info_score(labels_true, labels_predict)
f1t = f1_score(labels_true, labels_predict, average='macro')
accurracyt =accuracy_score(labels_true, labels_predict)
silhouettet = silhouette_score(dataset, labels_predict)

print('Confusion Matrix: \n', cft)
print('\nCalinski-Harabaz Score: ',hbt)
print('\nAdjusted-Rand Score: ',arit)
print('\nAdjusted Mutual Info Score: ',amit)
print('\nF1 Score: ',f1t)
print('\nAccuracy Score: ',accurracyt)
print('\nSilhouette Score: ',silhouettet)
```

```
Confusion Matrix:
 [[ 0  0  0 14  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 47  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 63  0  0  0  0  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  3 61  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]]

Calinski-Harabaz Score:  101.56858371190435

Adjusted-Rand Score:  0.2176591733241328

Adjusted Mutual Info Score:  0.6909315583831555

F1 Score:  0.3717555703780808

Accuracy Score:  0.484375

Silhouette Score:  0.3003507304934133
```

5.0.6 4.3.2 Spectral Clustering - Selecionado

[157]:
```python
data_spectral2['Label'] = spectral2.labels_
data_spectral2.head()
```

[157]:
```
          0         1         2         3  Label
0  0.766923  0.183892 -0.639551  1.035764     13
1  0.857375  0.199688 -0.545819  1.050677     13
2  0.842299  0.215484 -0.655172  1.035764     13
3  0.676472  0.294465 -0.592685  0.976113     13
4  0.857375  0.215484 -0.545819  1.005939     13
```

[158]:
```python
centroide_spectral2 = centroide(data_spectral2)
```

[159]:
```python
dataset = data_reduzida.values

class Data:
    namostras = 0
    ndim = 0
    ncluster = 0

newData = Data()

newData.namostras = len(data_reduzida)
newData.ndim = len(data_reduzida.columns)
newData.ncluster = 16


labels_true = lista

# predict recebe os rotulos preditos pelo algoritmo de clustering
predict = rotulos(centroide_spectral2, 16, dataset, newData)

# labels_predict sao as labels ja organizadas para comparacao correta com os␣
↪rotulos originais do conjunto de dados
labels_predict = labelmatch(labels_true,predict,newData.ncluster)


# METRICAS PARA AVALIACAO DO CLUSTERING
cft = confusion_matrix(labels_true, labels_predict)
hbt = calinski_harabasz_score(dataset,labels_predict)
arit = adjusted_rand_score(labels_true, labels_predict)
amit = adjusted_mutual_info_score(labels_true, labels_predict)
f1t = f1_score(labels_true, labels_predict, average='macro')
accurracyt =accuracy_score(labels_true, labels_predict)
silhouettet = silhouette_score(dataset, labels_predict)

print('Confusion Matrix: \n', cft)
print('\nCalinski-Harabaz Score: ',hbt)
```

```
print('\nAdjusted-Rand Score: ',arit)
print('\nAdjusted Mutual Info Score: ',amit)
print('\nF1 Score: ',f1t)
print('\nAccuracy Score: ',accurracyt)
print('\nSilhouette Score: ',silhouettet)
```

```
Confusion Matrix:
 [[ 0  0  0  0  0  0 15  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 47  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]]

Calinski-Harabaz Score:  307.79807227114503

Adjusted-Rand Score:  0.10681904679356016

Adjusted Mutual Info Score:  0.3771666990795962

F1 Score:  0.02460697197539303

Accuracy Score:  0.125

Silhouette Score:  0.2626307993848876
```

dim128-clustering

May 26, 2020

1 0. Introdução

Trabalho Clustering:
Aluno: Gabriel Luiz
Disciplina: Tópico em Aprendizado de Máquina
Objetivos :

- Escolha dois datasets rotulados.
- Realize a análise estatística, visualização e pré-processamento dos dados.
- Realize os experimentos criando duas bases de teste distintas:
-   – considerando todos os atributos do dataset ;
-   – selecionando alguns atributos e descartando outros;
- Aplique três métodos de clustering distintos nas duas bases acima.
- Para cada dataset , em cada uma das bases, analise os resultados segundo medidas de qualidade de clustering , usando índices de validação interna (SSW, SSB, silhueta, Calinski-Harabasz, Dunn e Davis-Bouldin) e externa (pureza, entropia, acurácia, F-measure , ARI, NMI).
- Proponha uma maneira adicional de comparar os resultados obtidos além das medidas acima.
- Compare e interprete os resultados dos dois experimentos em cada dataset

1.1 0.1 Dependências

Para realização da tarefa foram utilizados as seguintes bibliotecas:

[1]:
```
from datetime import datetime
import numpy as np
import pandas as pd
from sklearn.cluster import *
import seaborn as sns
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.feature_selection import SelectKBest
```

```
from sklearn.feature_selection import chi2
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import silhouette_score
from sklearn.metrics import calinski_harabasz_score
from sklearn.metrics import adjusted_rand_score
from sklearn.metrics import adjusted_mutual_info_score
from sklearn.metrics.pairwise import euclidean_distances
from scipy.stats import mode
from munkres import Munkres
```

2 1. Dados

Para realização das tarefas envolvidas neste relatório utilizou-se o arquivo **dim128.csv** que contém dados não descritos, onde foram feitos para a realização de clustering que se encontram no site: http://cs.uef.fi/sipu/datasets/

2.1 1.1 Carregamento do arquivo

[2]:
```
from clustering.labelMatch import rotulos, labelmatch
dataset = './dataset/dim128/dim128.csv'
clusters = './dataset/dim128/dim128pa.csv'
```

[3]:
```
data = pd.read_csv(
    dataset,
    header = None
    )

label = pd.read_csv(
    clusters,
    header = None
    )
```

[4]:
```
data.head()
```

[4]:
```
     0    1    2    3    4    5   6    7    8    9  ...  118  119  120  121  \
0  145  142  131  135  208  209  65  128  183  131  ...  199  218  182   53
1  149  148  137  137  213  209  71  125  183  125  ...  198  222  182   52
2  151  144  135  132  210  208  67  124  183  128  ...  198  218  182   52
3  148  141  136  135  207  209  65  127  184  130  ...  197  219  184   50
4  146  145  136  135  208  212  70  130  185  129  ...  199  217  182   52

   122  123  124  125  126  127
0  144  198   93   34   99   79
1  148  198   97   35   99   78
2  144  196   93   38  101   78
```

```
3  144  198   92   36  101   82
4  148  198   95   36   96   80

[5 rows x 128 columns]
```

[5]:
```python
data.describe()
```

[5]:
```
                 0            1            2            3            4  \
count  1024.000000  1024.000000  1024.000000  1024.000000  1024.000000
mean    125.248047   150.040039   134.053711   134.069336   118.694336
std      51.254859    48.465458    49.652222    38.661577    54.941676
min      31.000000    45.000000    42.000000    46.000000    35.000000
25%      89.500000   129.500000   104.500000   100.750000    76.500000
50%     117.000000   145.000000   142.000000   139.500000   111.000000
75%     158.500000   191.000000   174.000000   167.000000   158.000000
max     220.000000   225.000000   205.000000   195.000000   227.000000

                 5            6            7            8            9  ...  \
count  1024.000000  1024.000000  1024.000000  1024.000000  1024.000000  ...
mean    145.112305   125.099609   117.110352   108.508789   126.273438  ...
std      44.562082    51.200904    48.900247    51.715931    50.317170  ...
min      65.000000    52.000000    41.000000    31.000000    41.000000  ...
25%     111.250000    66.000000    72.000000    68.000000    89.000000  ...
50%     143.000000   130.000000   116.000000   100.000000   121.500000  ...
75%     180.000000   171.250000   152.250000   137.250000   176.000000  ...
max     218.000000   207.000000   220.000000   207.000000   218.000000  ...

               118          119          120          121          122  \
count  1024.000000  1024.000000  1024.000000  1024.000000  1024.000000
mean    145.520508   133.936523   130.793945   136.500000   136.348633
std      54.379262    55.852890    58.455433    52.589374    51.880328
min      34.000000    42.000000    41.000000    47.000000    30.000000
25%     105.750000    90.750000    83.000000   103.500000   100.250000
50%     150.500000   133.000000   111.500000   134.000000   133.000000
75%     194.000000   187.000000   195.750000   184.500000   187.000000
max     223.000000   224.000000   222.000000   218.000000   218.000000

               123          124          125          126          127
count  1024.000000  1024.000000  1024.000000  1024.000000  1024.000000
mean    117.336914   123.756836    99.931641   110.326172   151.151367
std      60.981599    46.710213    49.196389    60.645574    49.358342
min      32.000000    25.000000    27.000000    30.000000    58.000000
25%      60.750000    94.000000    63.000000    54.750000   114.750000
50%     113.500000   124.000000    87.500000    98.000000   179.500000
75%     181.250000   159.000000   128.250000   168.000000   190.000000
max     209.000000   210.000000   194.000000   215.000000   204.000000

[8 rows x 128 columns]
```

3 2. Pré-processamento

Validações efetivadas:

- 1. Dados faltantes representados por "NaN"
- 2. Dados que não possuem valores númericos

[6]:
```python
data.isna().sum()
```

[6]:
```
0      0
1      0
2      0
3      0
4      0
      ..
123    0
124    0
125    0
126    0
127    0
Length: 128, dtype: int64
```

[7]:
```python
for col in data:
    print(col, data[col].unique())
```

```
0 [145 149 151 148 146 143 153 147 150 154 144 142 152 156 215 217 213 214
 216 218 220 209 210 212  95  96  97  94  98  91  93  92  75  76  74  81
  73  77  79  78  82  83  85  84  80 112 111 113 114 119 109 116 115 110
 128 127 129 130 125 136 126 131 120 118 123 117 121 193 195 196 194 197
 198 191 199 101 100  99 104 102 103 135 138 137 141 140 139  31  35  33
  37  36  38  32  34  39  69  72  62  66  68  71  67 173 175 179 166 176
 172 177 174 181 170]
1 [142 148 144 141 145 146 147 143 149 153 140 150  57  58  59  61  56  55
  63  62 191 193 189 190 192 196 194 118 117 116 119 120 115 122 121 195
 188 220 219 216 217 218 214 223 221 225 163 164 162 165 166 160 167 207
 208 209 206 211 210 204 212 136 135 137 134 138 132 139 114 157 154 156
 155 151 152  51  52  50  49  53  54  45  48  47]
2 [131 137 135 136 121 138 134 130 139 133 126 132 153 123 124 122 125 127
 120 128 173 171 174 172 169 176 175 204 203 202 205 197 196 195 194 192
 198 193 200 191  47  46  48  45  43  49  44  50  42 114 111 115 117 108
 113 116 148 147 149 142 145 146 150 144 129  51  52  53  54  55  88  90
  86  87  89  91  94  84  85 162 166 164 165 163 161  59  60  57  58  64
  62  56  68  63 170 178 154 143]
3 [135 137 132 141 133 140 136 134 138 139 147 149 150 148 151 146 104 103
 105 106 102 107 101 124 123 119 121 122 125 126  96  97  98  99 100  95
  94 159 161 158 160 157 156 154 167 168 166 169 171 165 163 164 176 186
 188 185 187 190 184 189 127 129 128 130 131 170  92  93  91  90  50  47
  51  48  49  53  52  46  54 191 192 194 193 195 142 144 143 145 173 172
  84  85  81  86  80  83  87  82  89]
```

```
4 [208 213 210 207 205 216 206 209 211 202 212  43  40  42  44  41  46  39
   38  35 158 159 157 160 162 156 154 165 155  81  80  82  79  78  85 112
  110 113 116 111 114 108 109 104 103 105 106  95 107 102  68  69  67  71
   66  70  72 130 131 129 132 127 137 128 124 133 125 126 123  37 101  47
   48  45  49  51  50  54  52 221 220 224 219 222 227 189 191 190 192 188
  186 193 194 187]
5 [209 208 212 201 214 210 203 211 206 205 207  92  93  90  95  94  91  97
   96 147 149 145 148 144 150 152 146 126 125 127 130 129 128 124 139 140
  136 138 137 135 142 213 216 215 218 217 141 131 173 171 172 169 174 160
  159 161 151 158 157 163 156 164 175 176 168 178 177 170 189 188 190 191
  186 187 192 193  99 100 103  98 102 101  68  67  70  65  66  69  75 119
  123 121 122 120 114  73  77  72  83  76 197 195 196 199 202 200 194 198]
6 [ 65  71  67  70  74  64  73  66  63  68  69  75  95  96  97  93  90  92
   98  94 100  61  72 140 138 141 139 134 142 143 135 144 145 146 148  60
   62  58  59  56  52  99 101 161 160 162 171 163 158 159 166 155 169 164
  157 156 167 181 182 180 183 179 177 178 205 207 204 206 203 198 202 201
  199  55  57 176 175 195 172 122 120 123 121 119 113 116 117 126 118 124]
7 [128 125 124 127 130 129 126 132 123 134 131 133 150 149 147 148 151 145
  154 153 146 152 170 171 169 172 168 174 175  71  69  72  70  73  74  44
   48  46  45  47  43  41 137 138 135 136 139 140  68  75  79  88  86  92
   89  87  84 178 177 176 182 179 181 158 157 159 155 156 160 142 144  81
   85  80  82  83  78 216 218 217 220 219 214 215 213 105 104 101 102 103
  106 107 109 108  55  54  56  52  57  53  67  76  77]
8 [183 184 185 181 178 180 182 186 177 179 188  39  40  36  41  37  42  43
   38 124 126 122 121 123 129 125 115 116 114 118 117 113 104 102 101 103
  105 107 100  81  83  82  80  77  87  84  79  73  74  72  76  75  69  71
   68  63  64  62  65  61  86  88  78  85  90  89 206 203 201 204 202 197
  205 199 207 165 166 167 164 163 162 109 106 110 108 111 112  33  34  35
   32  31  66  60  59  98  99  97  96  93 198 196 193 195 194 200]
9 [131 125 128 130 129 127 126 132 145 138 133 182 183 185 181 184 180 178
   45  46  44  47  48  43  42  41  49 215 214 213 218 216 212  88  86  90
   85  87  89  84 196 197 195 199 194 198 193 200  97  96  91  93 176 175
  177 170 173 174 121 123 124  53  52  55  50  51  54  56  57  77  78  79
   76  75  80  82  74  81 150 151 149 152 147 153 148  92 116 114 115 117
  120 118 119 112 107 109 106 104 105 122 103 108 179]
10 [151 149 152 153 150 148 156 155 147 154  64  63  62  59  61  65  66  42
   39  43  41  44  38  40  45  37  47 127 126 128 125 124 123 129 142 141
  139 140 137 144 143 196 198 195 197 199 194 122 120 135 121  68 171 170
  172 178 168 174 173  46  34  36  70  69  71  67  74  72  73  53  50  55
   52  51  54 221 216 220 224 215 222 219 217 223 136 138 133 164 162 160
  161 163 165 166 159 158]
11 [160 162 158 161 159 153 163 155 165 154 157 164 119 120 121 118 116 117
  115 123  93  94  92  95  96  97 213 212 214 215 211 217 219 103 104 102
  100 109 105 101 106  98  65  62  66  63  69  64  68  58  67  61  88  89
   87  90  91  84  50  49  51  47  48  52  53  54  56 193 194 196 191 192
  195 190  79  80  78  83  81  76  74  82 122  39  37  38  36  34  35  40
   44 198 197 200 205 199 179 180 178 181 182 177 176  55]
12 [193 192 191 190 195 199 185 188 189 194 117 114 118 113 116 119 120 115
```

```
  112 123 122  96  95  94  93  98  97 196 197 198 200 205 207 206 209 210
  211 208 204 203 202 141 140 137 142 144 139 138 177 178 175 179 174 173
  180 111 110 109 107 183 184 181 182 186 187 105 106 103 104 101 108 102
  100  99  37  38  36  33  35  39  34  32 168 165 169 170 171 166 167 162
  164  56  57  59  53  54  55  58  50  52]
13 [211 212 209 216 210 206 215 213 214 208 202 207  53  54  55  52  51  56
   58  38  36  39  40  37  35  41 143 141 142 144 145 140  47  46  48  49
   45  44 146 148 147 149  50 217 218 219 220 186 185 184 183 188 187 180
  181 179 173 174 172 171 175 176 116 115 122 117 114 110 118 113  76  77
   73  78  80  72  74  75  79 163 164 162 160 166 161 157 165 151 150 152
  154 155  67 199 200 197 196 201 198 195 203]
14 [172 169 171 170 166 174 168 176 165 173 175 177 146 144 145 147 148 149
  143 142 130 131 129 132 128 134 160 158 161 159 162 154 157  42  43  41
   44  45  38  46  40  47 164 163  79  80  82  81  77  76  78  83  75  85
   68  69  70  74  67 101 102 100 105 103  98  97  99 107 104  60  61  59
   63  62  65  64  66  57 191 190 192 184 189 193 188 195 187 196 197 198
  199 194 200 126 123 127 125 124 122 167  84  86  94  34  35  36  33  32
   37  30  39]
15 [ 79  77  83  76  80  71  82  78  74  73  84  81  96  95  98  94 101  92
   97 100  91  99 163 161 162 164 160 165 114 116 115 113 117 118 122 123
  126 124 119 127 121 125 120  58  59  60  57  56  65  42  55  61  62  63
   64  34  33  35  37  32  36  31  38  75 174 173 171 176 172 175 170  40
   41  39  45  44 215 211 214 213 216 212  93  88  89  90 112 110 111]
16 [197 195 200 201 193 199 204 198 196 194 191 113 115 114 117 112 118 116
  111 176 177 178 175 179 174 213 214 215 212 211  44  43  41  47  42  45
   46 110 109 108 107 125 124 127 129 126 119 122 128 123 209 208 206 210
  207 101  98 100 102  97  99  93  96  95  94  77  75  76  79  78  73 161
  160 158 159 162 156 157 153 154 149 147 146 148 145 150 144 165 166 167
  168 163 164 170 169 172]
17 [146 151 143 148 145 147 140 149 144 137 142 150 162 161 160 164 159 163
  157 158 103 105 104 102 100 101 106  99 124 125 122 123 128 126 221 222
  224 219 220 218 223  94  93  95  96  90  97  37  39  38  34  40  41  36
   35  32  33 170 169 171 166 168 174  45  46  44  50  42  47  48  49 173
  175 177 172 179 176  89  91  85  88  92  29  31  30  27  28 127 121 120
  119 129 130 135 138 136 133 131 139 134]
18 [211 209 213 210 216 191 208 212 207 206  82  81  84  83  80  79  86  78
  203 204 199 202 205 120 119 118 121 117 122 115 164 162 163 165 158 161
  160  59  58  60  61  56  62  55  63  66  57  52  64 143 142 144 141 140
  145 172 173 177 170 171 169 174 176 178  51  50  49  48  47  54  53  46
  183 186 184 185 182 179 180 187 168 167 166 214 215 217 218]
19 [ 86  85  88  84  91  87  90  80  82  95  81  83  89  79  75  77  76  72
   78  70  73 121 122 120 119 118 184 183 182 185 186 152 151 154 153 149
  150 155 158 156 215 213 216 214 217 212 219 211 157 161 159 162 163 197
  196 195 198 194  51  50  53  47  46  54  49  52  55  45  48  64  63  65
   66  62  68  59  61 146 148 147 145 144 143 142 210 138 141 128 126 123
  116 218 221 220 223 222]
20 [162 164 160 158 161 157 163 159 165 166 169 156 167 168 170 112 111 113
  109 108 110 114 206 207 208 205 209  82  83  81  84  85  80  86 139 140
```

```
 141 138 142 137 136 151 150 153 149 152 147 143 118 119 117 116 121 120
  79  75  78  77  71 174 173 171 175 172 177 178 155 154 184 181 183 185
 186 179 182 180  46  45  47  48  41  44  49  43 198 200 197 195 204 199
 105 107 104 106 101 103]
21 [208 206 211 204 207 205 198 199 200 210 209 201 203 202  71  70  72  69
  73  74  76  44  45  43  42  86  88  87  84  85  90 218 222 221 220 217
 219 223 224 139 144 140 138 137 141 136 135  66  65  67  64  68  63  75
  56  57  58  60  59  55  52  53  54  49  51 197 195 193 196 194 192 191
  92  89  83  91  94  62  93  95  96  98  97  99 179 180 175 185 181 178
 177 182]
22 [204 206 205 209 203 210 202 200 208 211 207 201 152 153 151 156 147 154
 148 149 150  80  79  76  81  83  78 119 121 122 120 123 117 118  82  85
  84  86  66  65  63  64  59  67  68  62  60 124 125 126 146 155 159 105
 104 102 106 103  99 100 101 108 107 145 114 112 116 115 113 110 111 175
 172 174 171 176 178 173 170 196 198 193 199 197 190 192 194  58  57  56
  54  61 195]
23 [152 155 149 153 151 148 150 154 158 146  52  50  51  49  48  53  54  56
 147 145 144 141 143 142 139 140 138  79  81  80  84  78  75  82  76  94
  92  93  91  95  96  89  97 184 185 186 190 183 187 178  41  40  42  38
  43  39 170 169 172 167 168 173 164 171 165 174 175 177  99  98 101 204
 207 203 205 202 206 105 102 108 106 103 104 100 107 112  69  68  71  74
  70 208 211 210 209 213 212 215 214 216]
24 [207 206 213 203 200 208 205 209 211 214 202  84  85  86  78  82  88  87
  83  81 216 215 217 212 218 123 124 125 122 127 120 146 145 144 147 143
 142 141 148 150 149 151 152 140 139 137 138 135 154 153 155 159 156 111
 110 118 113 112 109 108 116 119 117 121 126 220 219  90  97 157 161 158
 160]
25 [ 44  47  43  48  46  53  51  45  50  49  38  83  82  85  84  81  79  76
  80  78 215 214 216 213 212 217 221 104 103 105 102 101 106 107 108 111
  63  62  64  60  65  61  66 178 179 176 180 177 181 168 173 174 175 172
 170 189 188 190 192 191 187 193 122 125 123 119 118 124 127 121 128 126
  87  88  89  86 144 146 143 145 141 142 147 148 194 195 211 208 209 207
 204 210 206 200  37  35  36  39  26  41  33  57  59  52  58  56]
26 [122 118 121 120 128 119 123 126 125 113 124 117 142 140 145 143 141 138
 139 146 147 144 149 132 131 127 134 130 133 129  96  98  95  97  94 100
 101  99  84  83  82  88  81  85  86  80  87  43  40  42  45  44  41  47
 102 103 148 150 181 182 184 183 180 179 187 185 157 156 155 153 164 159
 158 162 154 116 115 108 112 114 176 172 174 175 177 173 178 105 106 107
 104 111 109  78  77  74  75  76  79]
27 [219 220 218 221 222 223 224 216 225  81  82  84  83  75  79  86  80 147
 146 149 148 145 144 206 207 210 202 205 208 204 209 100  99 102 101 103
  97 105  98 168 165 169 166 167 173 170 172 203 212 200  96  95  94 151
 150 152 155 154 153 156  46  47  45  49  44  54  48  41  43 183 182 184
 177 181 180  56  55  53  52  50  51  57 171  58 157 158 159]
28 [196 195 194 200 201 184 197 189 199 193 192 198 190 203 150 149 151 148
 152 153 147 133 134 136 132 135  44  45  46  42  47  43  40 119 122 120
 123 121 118 116 113 114 112 111 109 115 180 181 178 182 174 179 183 188
 187 191 186 185 172 175 173 177 168 170 171 154 155 142 143 141 140 144
```

```
 138 145 139 146 161 164 162 160 163 167  79  77  76  75  73  78  74  71
  80  83  70 176  64  65  63  61  62  67  58  66  69  68]
29 [ 79  82  87  81  86  80  83  75  89  85  78  84  73 174 173 175 172 171
 177 176 148 147 149 146 145 141 150 207 205 209 208 206 132 133 134 135
 136 139 131 129 130 128 127 156 155 157 153 159 165 158 154 123 124 122
 119 125 121 151 152  70  68  69  67  71  74  72  42  41  44  43  39  40
  56  55  57  58  59  53  62  54 189 188 190 191 192 196 195 164  52  50
  51 142 144 143 140]
30 [139 138 142 136 131 141 137 144 140 134  40  41  39  42  43  44  36  37
  45 219 218 220 221 217 223 214 222  38 105 106 107 104 108 109  70  71
  69  68  67  73  61 199 198 200 209 197 204 206 196 195 194 193  72  64
  74  75  77 190 192 191 189 184 185 183 186 188 180  98  95  97  96  99
 100  91  92  93  76  46  49 102 101 103 132 133 135 126 130]
31 [189 191 195 190 193 196 192 184 197 194 188 187 211 209 210 212 208 213
 207 216 215 214  65  66  68  63  64  62  95  94  96  98  97  92  31  32
  30  33  29  34  36  35 134 138 132 135 133 136 145 144 146 143 142 147
 148 150 149 141 206 204 205 203 201 202 198 199  38  39  37 172 171 176
 173 170 178 165 175 174 152 151 153 154 155 167 169  48  49  46  51  50
  47  53  52  56 218 217 219]
32 [ 32  34  38  36  24  27  30  33  31  28  35  39  72  74  73  65  70  75
  76 115 114 117 118 119 113 116 150 149 151 152 154 147 148 167 168 169
 166 165 170 164 162 163  82  81  80  83  87  79  89 135 136 137 141 131
 138 132 134 133 145 146 142 140 144 160 161 205 204 206 208 203 207 128
 153 155 157 156 171 172 173 174 178 201 202 199 197 143]
33 [153 157 156 158 159 155 150 154 152 160 149  53  52  51  54  57  50  56
  55 129 130 128 127 131  49 215 216 214 219 213 217 218  91  93  92  95
  90  89  94 109 108 106 112 105 107 111 110 115  48  47  75  74  76  73
  77  72  79 124 126 125 132 134 133 168 167 166 170 169 163  46  45  44
  43 173 172 174 171 175 178 177 179 176 180 181  87  86  97  88 101 103
 104 102 100]
34 [123 120 121 122 124 110 119 116 118 126 127 128  54  53  52  51  50  58
  55  59  96  95  97  91  94  98  92  37  35  36  38  33  34  32  41 173
 174 172 171 170 169 177 176 161 162 164 160 156 158 153 163 166 159 168
 167 165 125  61  62  64  63  57  60 109 113 111 108 112 115 114 107 147
 148 146 149 145 150 155 154 157 151 152 136 135 134 138 137 139  89  90
  86  88  87  93]
35 [111 113 117 115 112 116 123 106 114 118 110 119  57  55  54  56  58  59
  62  61 215 216 218 213 214 211 199 198 200 202 206 197 201 174 175 176
 178 177 173 180 172  45  47  44  42  46  49  53  63  60 171 170 168 165
 169 203 204 205 207 225 220 222 221 223 219 224 217  92  91  93  89  86
  94  90  88  96 125 124 126 122 127  40  37  39  35  34  38  36  33  50
  51 154 153 156 157 155 148 152 160]
36 [ 61  59  58  56  63  45  57  60  54  55 158 157 156 160 161 153 159 155
 181 183 184 180 182 179 185 177 211 212 213 209 210  68  69  67  64  71
  70  66  92  94  91  93  90  88  89 133 132 134 138 131 125 130 135 169
 168 171 170 166  62  65 137 136 140  39  40  41  38  42  37  43  44 189
 186 187 190  48  51  49  50  47 123 121 120 124 122 119 162 167 163 126
 128 127 129]
```

```
37 [199 204 202 198 200 197 201 194 195 196  92  91  93  96  90  94  89  95
  87  88 127 129 125 128 124 130 126 120  97  98 101  99 203 205 209  69
  70  68  67  73  75  71  55  54  56  57  60  58  51  50  53  52 176 174
 177 178 170 172 175 179 173 192 193 191 156 154 155 157 152 158 160 153
 159 161 213 214 212 216 215 210 167 168 166 169 164 171 165 163 135 131
 132 133 136 137 142 139 134  72  74  76  66  64]
38 [131 138 136 137 141 135 133 132 134 123 130 147 115 113 112 114 116 110
 111 119 118  74  72  75  73  76  71  88  89  85  87  90  86  91 120 117
 122 121  77  80  78  81  79 187 185 186 189 181 188 182 201 194 191 140
 143 144 139 142 206 207 209 208 205 204 203 103 101 102 105 104 100  99
 106  48  49  50  52  51  47  64  68  65  63  67  66  62  69  70  61 184
 145 146 148 149 126 127 128 125 150]
39 [158 159 154 160 152 157 156 150 161 155 151 163 166 171 173 174 172 170
 169 168 181 180 184 183 177 182 179 178 213 214 212 211 210 215 216 222
 221 220 219 223 218  85  86  87  82  84  83 194 193 196 195 198 199 192
 113 114 108 110 115 112 175 167 176  68  66  65  67  70  72  63  69  71
  62  64  73 202 200 201 197 208 204 209 207 205 206 203 126 111 117  95
  96  97  98  94  93  92  91 100]
40 [211 208 210 209 207 214 212 205 204 213 215 117 116 115 113 114 118 188
 191 189 187 186 185 190 206 201 200 197 203 199 202 198 216 217 161 162
 159 163 160 164 156 157 155 222 223 226 218 221 219 224 225 122 123 124
 121 120 119  45  47  46  41  44  48  50  49  88  87  85  83  89  86  91
 183 182 184  90  92  93  94 154 151 153 149 152 168  65  64  66  61  59
 181 177]
41 [148 149 152 150 151 147 153 141 154 155 146 158 145 144 143 203 202 204
 205 199 201 200 172 171 174 170 173 120 119 118 123 121 117 111 108 114
 110 112 113 109 194 193 195 186 192 196 136 137 135 133 138 134 214 215
 216 213 212 211 210 209  90  87  89  86  91  92  88  93 177 178 176 182
 175 179 181 180  97  94 106 105 107 104 125 116 223 208 206]
42 [ 36  37  39  35  38  41  34  32  42  31  40  46 105 107 104 103 106 102
 100 111 137 136 138 135 140 133 134  48  49  47  50  44  52  51  54  53
 127 129 128 126 125 123 124  98  99  95 101  96  97  94  93  87  92  89
  90  88 116 117 120 119 118 115 112 114  84  83  85  82  80  86 199 200
 201 198 197 196 194 205 181 182 179 178 180 183 177 186 185 187  91 153
 154 155 157 156 148 151 150 152]
43 [ 99 100  98 101 111 110 103  95 102  97 219 220 218 222 221 217 212 216
 224 215 137 134 138 136 140 135 141  86  85  87  83  88  89  84  38  40
  37  39  41  43  96 214 213 104 106  90  92  93 105 199 198 200 196 205
 197 202 203  42  44  45  47  36 180 181 179 178 177 109 108 107 112 124
 121 122 131 123 125  72  70  69  67  71  76  68  65  64  73  74]
44 [128 130 131 122 129 133 132 127 125 137 126 123 135 149 150 148 151 147
 146 143 145  99  96 100 102  97 101  98 104  93  94  92  91  89  90  95
  61  59  58  60  62  57 153 134 161 160 159 156 162 163  40  39  41  43
  38  42  35  48 209 211 210 207 206 208 213 204 212 169 168 167 166 165
 170 171 114 115 118 113 116 111 110 112 117  72  73  74  71  75  70  67
 152 155  51  50  47  55  49  52  37  34  33  45  36]
45 [175 172 173 171 181 174 177 176 167 178 179 149 148 147 150 152 146 154
 144 193 192 194 188 190 196 195 134 135 137 133 136 132 121 122 119 123
```

```
 126 120 124 114 110 115 112 113 117 116 168 170 169 166 164 165 163 187
 189 186 145 140 143 209 211 213 212 214 216 210 215 208 219 153 151 159
 217 218 222 220 221 197 198 200 199 162 160 161 158  76  73  75  77  82
  70  78  79  74 131 129 130]
46 [ 80  82  81  77  90  75  83  79  84  72  86  78  85 105 107 108 106 109
 104 113 103 116 119 117 118 115  91  92  89  88  93  53  54  58  52  55
  49  56  51 164 166 165 163 162 161 208 209 207 206 204 202 210 213 211
 205 212  50  48 190 192 191 183 189 188 194 193 187  32  33  34  37  31
  36  35  42 185 186 158 159 156 160 157 125 124 122 126 123 121 127 154
 155 120 111 110 114 112]
47 [117 121 120 116 129 128 119 124 118 122 115 114 123  91  90  89  92  95
  93  86  87  94  98 179 180 178 184 177 176 174 173 172 175  88 162 161
 163 160 159 158 164  96 107  97 102 151 152 154 150 153 143 148 181 183
 182 186 185  61  60  59  66  58  62  57  65  63  56  55 113 112 155 156
  42  47  49  48  46  44  43  45  54 167 166 165 169 168 209 208 207 206
 210 205 211 218 212 202 213]
48 [ 41  43  39  46  40  37  45  42  44  38  36  60  57  59  58  61  62  56
  64  94  95  93  92  96  97 174 173 175 176 171 172  90  91 190 187 189
 191 195 186 188 194 192 197 196 202 198 200 199 208 170 169 167 168 112
 111 113 108 110 114 109 115 117 129 128 130 132 126 127 182 181 183 180
 177 185 193  49  47  50  48  51  55  52 184 179 178 141 142 143 144 139
 138 140 146 136 145]
49 [ 77  78  80  75  71  81  72  76  79  74  83 123 125 122 124 121 120 127
 111 110 108 109 113 112 130 131 132 129 128 133 126  49  53  48  50  51
  45  46  47 158 157 159 156 151 161 160 148 154 107 114 106  97  98  96
  92  94  99  95 175 177 176 178 171 179 181 173 174  35  34  36  37  33
  38 119 162 163  73  70 193 191 195 189 190 192 186 187 185 184 188]
50 [157 156 158 151 155 153 160 159 163 154 147  34  33  35  31  38  36  32
  37  30 161 149 148 150 152 108 107 111 109 106 110 104 114 113 115 118
 112 116  72  71  68  73  70  74  92  91  93  89  95  90  87  98  88  39
  40  46  41 199 198 200 201 197 195 194 202 206 208 207 209 203 205 211
 204  59  58  61  60  62  56  57  65  64 124 125 123 126 122 121 120 128
 117 119]
51 [178 175 176 174 177 181 187 179 173 170 172 130 129 128 132 123 131 127
 126 125  91  92  90  94  89  93  95  86 154 153 155 152 156 150 151 157
 136 135 138 134 137 133  46  42  48  45  47  49  43  44 100  99 101 102
  97  81 105  98 103 171 124 117 122 121 120 118 221 220 222 219 218 223
 200 198 199 202 201 196 195 197 203 148 149 194 193 192 190 191  83  88]
52 [ 41  34  38  36  39  29  37  43  40  42  35  44  45  78  79  80  77  81
  75  76  84  82  83  62  63  58  64  60  61  57  59  55  54  56 174 175
 173 179 176 177 172 178  86 114 112 115 116 118 113 208 207 209 206 210
 204 211 192 191 189 188 190 198 184 196 187  74  73 134 136 135 131 132
 137 133 139 138 181 180 182  67  71  69  68  70  65  72  66 157 159 161
 155 160 158 152 156 154]
53 [149 148 150 152 155 153 157 145 142 151 187 186 185 190 191 189 188 184
 182 206 207 205 204 208 203  35  36  34  38  33  37 172 177 168 173 171
 174 176 170 175 146 154  98  97 100 103  99  96  95  92  46  44  45  42
  48  43  47 114 116 115 113 117 112 111 156 158 124 122 125 127 123 126
```

```
 119 128 129 108 107 106 105 109 101 104 121 120 192 202 193 201 200 209
 210]
54 [182 178 177 183 179 180 176 185 175 181 189 190 184 135 142 136 139 137
 133 138 140 134 170 169 172 171 168 174 163 165 123 122 126 125 124 121
 173 110 109 111 108 107 115 112 199 198 200 194 195 208 201 197 117 118
 116 119  63  64  62  66  61  65  60  45  43  46  40  42  44  47  50  48
  41 132 160 159 161 157 162 154 158 156 164  69  67  59  68 106 104 105
 114 128 131 129 130]
55 [ 97  95  98 102  96  91  94  99 103 101 100  47  44  45  48  46  50  51
  42  49  43 200 199 198 202 203 197 201 144 145 143 146 142 104 169 167
 168 170 166 172 165 164 192 194 193 191 187 195 186 183 190 189 112 113
 114 115 111 188 182 185 184  66  64  63  65  71  67  62  70  68 196 117
 118 119 116 120 121 123  40  52  41 150 151 147 149 141 140 136 139 148
 138]
56 [130 127 128 129 137 123 125 126 122 132 131 158 159 157 160 162 161 163
 154 167  88  91  89  87  85  83  84 114 115 112 116 113 111 117  58  55
  57  53  56  60  54  59 140 141 139 138 142 135 143  97  96  98  99 103
 101 102  95 214 213 215 211 212 148 147 150 146 149 145 144 120 121 124
 110 109  82  80  81  86  79  73  69  72  71  74  70  76 198 199 202 197
 196 204 200 195 179 181 180 178 177 187 175 184  50  51  52  49  48]
57 [151 148 147 149 152 153 146 150 154 155 141 145  63  62  60  66  59  64
  61 130 129 132 131 128 133  71  74  75  73  76  77  72  79 115 117 114
 116 118 121 113 111 198 197 193 195 199 196 192 201 190 194 200  80  81
  82  85  78  84 143 144 142 139  91  89  93  90  88  92  86  87 103 102
 104 106 105 107 100 101  83 177 179 182 180 176 174 178 172 175 171 205
 206 203 204 207 202]
58 [198 203 200 202 201 196 199 205 204 207 194 210 169 168 172 167 170 171
 166 164 165  49  50  48  51  52  46 186 188 185 187 183  94  95  93  98
  96 101 103 100 102 107  99 105  97 109 146 147 145 148 143 150 141 140
 144  36  38  39  37  31  34  35  41  42  40  81  82  83  73  84  80  85
 209 211 208 206 212 190 192 191 189 193 184 182 195  66  68  67  64  65
  69  62  70  63  60  61]
59 [136 138 139 135 137 146 143 134 140 142 131 147 145 148 149 144 141 150
  72  71  73  70  76  69 127 124 129 128 126 125 215 213 214 218 212 216
 211 210  92  93  91  94  95  90  96  89  47  46  44  48  51  43  45  41
  40 204 205 206 207 208 200 123 121 122 119  80  82  75  81  77  84  79
  78  83  49 195 194 201 196 197 198 192 193  52  50  53  54  35  34  33
  36  32  37 118 117 116 115 203 202 199]
60 [ 52  53  45  55  44  51  50  49  54  43  56  81  82  86  80  79  83  78
 177 176 179 173 180 178 212 213 211 210 214 207 209 181 182 183 132 133
 131 130 138 127 136 134 202 203 200 204 199 196 201 205 198 129 128 125
 117 115 118 114 116 113 120  57  58  48 187 186 185 184 191  59  62  60
  61  71  73  72  74  69  70  76 107 104 112 106 102 105 109 103 159 162
 163 161 158 164 160 166 169]
61 [ 76  73  74  79  77  71  78  81  80  82  68  75  84 188 189 190 187 191
 192 137 139 138 136 134 135 142 140 198 197 200 199 202 201 196 156 157
 155 154 153 158 159  72  70  69  67 212 213 214 211 215 216 210 209 208
 218  83  89  85  86  87 102 103 104  98 101 100 105 147 145 146 144 148
```

```
 150 143 149 141  59  62  60  57  58  61  63  64  92  93  91  90  94  96
  88  95  97 160 162 161 164 163]
62 [ 56  50  55  54  53  59  58  57  66  60  64 153 151 152 154 156 157 150
 148 158 147 149 146 145 121 120 119 117 122 125  40  39  37  35  42  38
  41  34  46 199 200 195 198 202 201 197 196 204 188 190 187 192 186 178
 191 185 189 193 155 144 143 138 142 209 207 210 212 211 208 205 206 213
 203  52  51  65  63  68  61  67  44  43  45  47 100  99  97  98 102 101
 103 104  93  96  91 129]
63 [109 107 105 108 110 104 106 112 113 100 111 114 103 164 165 162 163 166
 160 168  81  82  85  80  83  77  84  79  78 161 159 181 178 182 180 179
 183 177 145 146 144 147 143 140 142 148 150  75  76  74  72  73  88  86
  87  93  89  90 141 151 152 102 101 176 175  46  44  41  47  48  45  49
  43  42  38 149 154 155 115 116 117]
64 [131 126 130 127 128 118 135 134 132 129 133  63  64  65  61  62  67  69
  68 204 203 205 207 206 208 202  54  56  53  55  52  57  51  66 117 115
 116 121 120 122 113 119 114 220 219 222 217 215 214 221 218 225 216 226
 209 195 196 194 197 193  42  40  43  41  46  48  45  39  47  37 187 189
 186 188 192 185 184 190 191  44  38  36 182 181 183 179 167 171 169 168
 170 175 172 174 164 173 211 212 213 210 180 177 178]
65 [ 62  64  65  66  63  70  60  58  56  57  71  72  59  67 160 161 159 157
 162 158 156 164 174 175 173 176 177 180 172  99 101 100  96  98  97  95
 125 124 122 120 123 119 127 121  86  90  85  88  87  89  84 140 141 139
 142 146 143 144 185 186 184 188 182 189 181 187 131 132 130 135 133 129
  94  92  91  93  55  53 183 190  50  52  51  48  49  54 154 153 152 150
 155 148 151 149 221 220 222 217 224 229 218 219 145 147]
66 [182 181 183 177 180 166 178 184 185 175 179 126 128 125 127 124 130 129
 119 138 139 141 137 140 142  67  68  66  65  64  69  70 211 210 208 213
 209 212 214 198 200 199 197 196 202 201 151 152 153 146 150 154 149 165
 162 167 168 164 169 189 186 131 132 133 134 136  58  57  56  55  59  53
  60  54  50 107 105 108 104 109 106 112 110 115  72 103  99 102 100 101
  98  93 143 144 145 207 203 216 206 205 204]
67 [104 107  99 106 105 108 103  97 101 115  98 110 109 221 222 220 223 224
 226 219 217 214 213 218 170 173 171 172 169 167 168 195 196 194 190 193
 191 197  65  66  63  64  62  67 134 133 136 130 135 131 132 138 139 140
 142 141 149 144 137 143  70  59 100 111 102 164 160 161 158 165 156 166
 162 163 157 174 175 145 147 146  34  33  35  36  37 187 188 186 189 185
 182 184 179 127 129 128 126 125 180 178 176 177 181 183]
68 [155 159 157 152 156 158 151 165 160 162 154 163 134 135 132 131 136 130
 133 140 142 137 138 100  96 103 101 107  98  99 102 104 190 191 189 192
 188 193 196 187 121 120 123 119 118 115 122 114 161 164 148 153 149 171
 172 173 170 174 168 169 175 176 180 177 179 178 144 143 146 145 139 147
 141  42  45  44  41  43  46  40  39  47 150 167 198 197 199 195 200 186
 184 194 183 185 182 181]
69 [117 118 116 119 109 115 120 122 124 123 121 112 125 205 206 203 204 208
 201 209 207 181 183 182 186 185 180 184 179  50  51  49  48  53  52  55
 210 211 212 213 214 215 150 152 151 149 154 153 148 147 132 131 126 130
 128 137 133 129 136 196 194 195 197 198 193  68  67  72  69  70  66  65
  71 202 155 146 114 113 110 111 176 177 175 178 174 170  42  40  39  43
```

41 38 44 46 34 35 45 36 173 172 171]
70 [123 119 124 120 121 114 122 125 126 130 128 118 111 183 181 182 187 184
180 177 219 220 217 218 221 159 160 158 157 161 156 127 58 57 59 56
55 60 61 190 189 191 185 193 197 195 194 186 141 139 144 142 140 136
143 137 173 174 176 172 175 171 198 199 201 200 196 204 202 68 67 65
69 129 110 109 108 112 107 214 216 209 215 213 212 211 132 134 133 135
131 92 91 90 93 89 86 95 87 88 85]
71 [215 218 216 210 217 212 221 220 214 222 219 69 68 71 70 67 73 72
65 58 56 57 59 60 182 181 184 183 178 180 179 209 208 211 207 205
192 194 193 191 189 190 196 175 176 173 172 177 174 188 187 162 163 164
159 161 166 160 165 167 186 92 93 94 87 95 91 90 96 97 119 118
117 116 120 115 121 123 124 122 230 224 34 33 31 35 32 36 125 126
127 129 128 130]
72 [78 79 75 77 73 76 81 80 82 74 43 44 42 46 48 41 45 47
51 157 158 156 159 160 154 101 100 99 103 102 97 104 105 106 217 220
218 215 219 216 214 152 151 150 153 148 149 143 126 125 124 127 123 128
90 91 92 88 85 94 89 86 87 84 83 178 176 177 174 179 181 175
180 183 171 167 169 168 170 166 136 134 133 137 135 131 132 129 138 155
107 108 110 118 109]
73 [88 92 87 89 81 86 91 85 94 90 84 93 205 202 204 206 207 208
209 211 203 201 49 48 51 50 42 52 47 62 61 58 60 63 64 107
108 104 106 109 111 196 195 193 197 199 198 194 45 46 41 54 38 69
68 70 66 71 67 158 157 159 152 156 162 161 174 170 172 171 169 168
173 175 177 99 101 100 98 95 103 102 96 97 65 154 155 160 151 82
83 79 74 80]
74 [161 162 165 163 166 158 164 159 160 170 157 205 206 207 208 204 209 203
210 186 185 187 190 189 184 188 191 179 154 156 198 197 196 195 199 194
201 202 200 60 61 59 62 71 64 57 63 56 58 65 192 214 213 216
221 215 218 212 217 40 41 42 39 43 36 44 181 182 34 35 33 32
31 66 67 73 219 138 136 135 134 137 133 132 139 141]
75 [62 59 63 61 72 60 65 67 66 64 57 182 183 181 179 180 184 176
202 201 200 204 205 203 206 199 207 174 173 175 172 177 198 208 178 187
186 185 188 190 55 56 54 53 52 58 50 51 161 160 158 159 168 164
162 163 157 165 141 142 134 144 145 140 143 137 48 49 46 47 34 33
35 36 37 39 38 155 156 152 154 189 191 193 68]
76 [36 38 39 37 34 40 42 30 35 33 41 177 178 175 176 173 179 174
113 114 116 111 112 115 110 63 64 62 61 59 60 65 189 192 191 190
188 197 186 187 193 85 84 86 87 91 88 90 83 219 218 220 215 216
217 221 161 160 159 158 163 162 157 164 156 165 138 139 137 133 132 136
134 140 141 135 142 155 154 153 152 150 149 98 96 99 97 100 95 94
54 55 56 53 57 58 52 184 181 182 183 185 180 209 212 211 210 213
214 208]
77 [84 86 81 87 75 85 89 83 88 80 79 82 178 177 179 180 175 181
176 174 96 94 98 95 93 97 92 36 34 33 35 32 31 99 100 101
104 102 90 203 202 200 204 206 201 197 196 198 195 199 194 47 49 48
46 50 45 56 44 91 184 78 76 186 185 183 189 187 191 193]
78 [156 157 155 158 154 163 159 160 161 166 162 149 202 203 198 201 204 205
207 206 200 199 214 212 215 213 216 217 129 130 133 128 131 127 164 59

```
 61  58  56  60  57  71  72  73  66  70  78  65  75 107 106 112 108 105
 110 109 103  47  48  50  46  49  52  51 113 111 114 117 115  88  90  87
  85  89  86  92  91  83  84  40  41  38  39  42  43 143 146 139 145 142
 144 148 141 147 135 177 175 181 176 174 180 172  67  69  68  64]
79 [ 36  38  40  37  39  32  35  29  34  31 148 147 150 149 152 145 146 144
 151 143  56  55  54  57  53  60 189 191 188 190 187  33 192 194 186 162
 164 166 163 167 161 158 156 165 160  64  65  62  66  61  67  69  74  68
  70  63  72 200 204 202 201 197 203 206 205 199 207 208 210 214 215 217
 218 212 213 216 219 211 193 195 168 169 170 159  41  42  79  78  84  77
  80  82  81  75  76]
80 [223 220 226 222 233 224 221 225 218 216 219 228 227 217  76  75  73  74
  77  78  71  82  79  95  97  96  94 101  93  98 167 168 166 165 164 169
 156 155 153 154 159 157 152 162  44  45  43  41  40  42  38  46  47  39
 148 149 145 144 147 138 150 127 124 122 126 125 120 123  99 100 102  62
  63  66  61  70  64  60  57  65  59 131 132 128 130 133 129 134 104 103
 189 190 188 191 192 193 195 187 110 109 105 107 106 113 170 163 171]
81 [152 151 149 150 153 148 147 155 145 170 169 164 168 166 172 167 165 171
 112 111 113 115 110 114  73  74  75  71  72  69  70 123 124 122 126 121
 120 125  81  78  85  82  80  83  79  86  54  55  56  53  52  50  58  51
  57 205 206 204 208 202 207 203  63  64  60  59  62  61  67  65  66  68
 109 104 108 107 215 216 213 212 210 214 218 211 209 219 217 177 176 175
 179 173 180 174 178 181  92  94  93  89  90  91 201 200 198 199 197]
82 [123 119 128 126 125 124 134 127 117 122 129 121  98  99  94  97  93 100
  96 101 102  54  55  53  57  56 190 189 191 192 188 118 115 120 116  58
  59  79  80  81  77  78  82  85  75 112 111 110 107 113 109 105 104 103
 106 133 130 135 132 137 131 141 136  52  51  50  48 108 114 152 154 151
 153 147 159 155 149  61  60]
83 [ 94  96  93  83  92  97  91  89  88  90  95  86 164 162 163 166 165 167
 161 168 169  43  44  42  45  41  46  47 218 217 214 216 219 220 221  72
  73  74  71  75 189 186 190 188 185 187 191 195 177 179 178 176 175 180
 174 181 182 183 171 170 192 193  98  99 100 101 116 115 108 114 120 117
 113 118 121 150 151 149 154 148 152 147 144 146 173 172 153 156 157 155
 158]
84 [121 119 126 118 123 122 124 115 116 120 125 127  73  74  72  70  79  69
  71  76  75 117 114  45  44  43  46  47 108 107 103 106 105 109 110 104
  99  98 100  97  96 162 161 163 160 158 165 159 164 112 113 199 200 198
 194 201 196 203 197 195 202 128 130 129 134  41  42  39  40  38  78  77
  94  91  95  89  92  93 111]
85 [213 214 215 218 209 212 208 211 210 217 216 102 100 101 104 103  99 108
 106 115 113 114 116 112  78  75  77  80  79  76 107 105  48  50  49  47
  51  46  52 183 182 184 181 180 186 192 178 189  70  69  71  68  67  53
  55  54  56  57  64  61  63  62  60  65  45 111  81  82  83  84 135 138
 141 137 139 134 136 140 142 133  40  41  38  39  42  36  43  44  37]
86 [ 73  71  72  57  74  70  68  75  69  77  64 145 140 144 142 143 146 141
 201 197 202 203 204 200 205 198 207 147 148  65  66  67  62 170 171 169
 168 166 172 221 220 223 219 217 222  80  79  78  88  83  81  82  76 162
 164 165 161 167 160 163 209 210 208 211 212 214 216 218 215 101  97 100
  98 102  99 157 154 156 158 153 159 155  43  44  45  46  39  42  36  40
```

```
  50  47]
87 [ 65  66  59  58  63  62  60  64  61  67  55 177 178 173 180 175 174 179
 176 182  56 181 183 185 184 109 110 111 108 113 112 107 171 170 169 172
  71  68  32  31  30  33  34  69  70  72 220 221 215 217 222 219 223 226
 224 133 132 134 135 131  50  51  48  53  49  52  54  46 218 213 216 214
  57 137 138 136 140]
88 [185 180 186 192 184 190 183 191 193 181 187 182  84  83  85  82  81  80
  86 114 110 112 113 115 116 119 111 173 171 174 172 170 175 169 109 107
 108 106 194 188 198 195 176 101 102 105 103 100 104  99  52  51  48  54
  50  49  53  56 199 200 197 201 204  76  78  75  73  77  79  74  71  95
  61 155 154 157 156 158 152 159 162 153 160 134 132 131 135 133 130 127
 138 136 128]
89 [131 128 130 129 133 120 132 136 122 134 135 210 208 211 209 212 207 213
 162 161 163 164 160 106 105 107 103 109 108 104 110  45  43  44  46  40
  47  48  42  49  51  50  54  53  52  55  99 102  91  90  92  88  89  85
  87  94 139 137 140  76  77  79  70  80  75 200 201 206 199 202 196 198
 193 197 203 205 217 223 218 216 222 219 215 220 221  97  93 101  95 191
 190 194 188 192 189  61  59  60  58  63  62  65  56  57  64]
90 [193 188 191 187 194 189 190 195 192 185 203 186 183 210 209 211 212 207
 214 213 206 208 105 106 108 104 103 100 107 101 162 166 163 164 167 165
 161 160 129 128 130 125 127 126 124 134  50  46  51  49  45  48  52  53
  47 204 205 201 197 198 202 200 139 138 141 137 136 140 143  66  65  63
  62  64  67  69  72  68  60  37  36  40  35  38  39  34  42  33  32  41
 135 133 132  76  74  75  78  73  79  80  77 176 177 178 175 174 180 173
  44  43 179 114 115 116 113 112 118 117 111]
91 [151 148 153 152 150 144 145 143 149 146 167 166 165 163 164 168 169 162
  74  73  75  72  76  78  77 201 199 203 200 198 202 134 133 135 136 130
 132 138 137  84  85  83  81  86  71  87  82  88  45  44  43  46  91  89
  90  93 124 122 125 120 123 121 114 127 116 161 160 159 158  94  92  56
  60  57  58  55  59 139 131 141 155 154]
92 [ 73  70  74  72  71  82  69  68  76  61  67  75  64 116 117 115 119 114
 120 118  62  60  63  58 121 113 122  86  88  87  89  85  84  92 153 156
 154 155 152 150 157 123 124 126 128 125 127  45  46  44  47  43  42 138
 139 140 137 133 142 135 136 143 141 134 220 219 218 212 217 221 216 215
 222  77  78  79  81  80  83 106 107 103 109 105 108 104 110 102 112 111
 100  96  98 101  99  95  93 191 192 193 196 198 195 189 194 188 190 186]
93 [150 148 153 154 149 152 151 157 156 147 145 181 179 182 180 178 183 177
 128 127 130 126 123 129 131  78  76  77  79  81  80 120 119 117 118 146
 143 144 170 172 171 173 169 168 175 160 167 159 158 161  82  73  84  83
  75 121 122 125 124 116 186 187 185 189 184 188 190 191 162 165  60  59
  61  64  58  62  34  36  33  30  35  32  37  28 195 193 196 194 199 206
 197 200 198 203 192]
94 [141 136 140 137 144 142 139 138 148 134 143 147 146 149 145 151 153 186
 185 184 187 188 182 160 162 161 158 159 164 157 112 109 110 111 107 108
 183 190 189 199 198 200 203 201 197 195 193 196  72  71  73  76  70  74
  98  99  97  96 102  95 155  69  78  66  67  68  77 178 176 175 177 173
 174 172 179 131 132 125 133 129 130 128 123 127 120 124 126 122 169 168
 171 170 165 167 166]
```

```
95 [201 197 193 200 196 199 208 198 202 203 194 204 195 216 218 215 214 217
 213 219 212  41  40  39  38  42  43 182 180 181 185 183 179 178 184  87
  90  88  91  89  92  86 153 152 154 155 151 156 157 162  94  77  93  67
  66  68  70  64  65  99 100  98  97 101  95 103 102 115 113 109 117 105
 112 114 111 116 118  54  55  53  50  57  58  52  51  33  35  32  31  30
  36  27  34 205 206 207 209 190 192 191 189 186  45  46  47  49  44  48]
96 [153 150 151 149 161 146 156 152 147 148 155 154 145  57  58  56  54  53
  59  60  63  52  62  55  64  79  76  80  78  83  82  75  84  77 208 209
 207 210 204 203 211 206 186 187 183 188 185 184 194 190 193 195 198 192
 196 191 197 199 180 181 179 176 178 182 177 202 205 201  66  61  65  94
  95  93  96 104  92  97  90 101 140 139 143 141 137 138 158 157 159 160
 117 114 116 115 112 118 120  51  49  50  47  48  46  45 165 166 164 162
 171 163 169 167 168 170]
97 [126 129 124 122 125 118 132 123 128 120 133  80  81  75  79  78  77  82
  83  74 201 202 200 204 199 207 157 156 159 155 158 154  53  51  52  56
  54  58  55  57 152 148 153 150 146 151  66  69  65  67  63  70  62  68
 111 110 109 108 113 106 149  71  72  73  76 116 117 114 115 193 192 195
 194 196 191 190  59  60 143 145 144 147 142 141 127  38  37  35  34  32
  39  43  36  42  33  41]
98 [119 118 120 121 122 125 129 123 117 104 105 103 106 107 108 100  99  97
 102 101  98  94 168 169 170 167 166 114 116 113 115 111 112 178 177 182
 180 179 176 207 206 205 204 210 208 209 128 127 126 130 165 163 164 162
 161 160 157 154 158 159  71  70  74  73  69  72 109 110  62  61  63  60
  64  59  65  66 172 171 173 175 174  76  77  75  78 124]
99 [ 72  66  71  69  70  68  74  73  67  77  78  79  75  58  56  57  59  60
  55  54 208 209 207 211 210 216 217 220 215 214 213 219 218  76 188 187
 186 189 185 183 190 184 193  80  83  81  82 140 141 139 138 143 136 146
 144 191 181 197  84  85 130 129 127 135 131 132 128 124 133 126  86  87
  88 203 202 199 201 200 205 198 196 206 204  63  64  65  62 122 123 119
 125 121 120]
100 [153 154 155 162 163 151 158 156 152 150 144 145 146 143 142 140 148 141
 147  69  67  71  70  73  65  68  64  94  93  92  96  95  79  78  81  77
  80  76  75 109 108 110 111 116 112 106 174 175 171 176 172 178 180 173
 170 179 168 202 201 199 204 200 198 107 101 105 113  91  90  88  89 117
 118 114 115 122 123 121 119 120 127 124 126  56  53  57  58  52  54  55
  60 128 125 129 218 217 219 214 223 220 216 213 104 103 102  99  98]
101 [108 110 111 109 102 104 113 115 112 100 107 105  41  42  40  38  43  39
 103 106 114  99  98  97 101  95  68  66  71  65  64  72  69  67  70 175
 174 176 173 178 177  63  55  74  73  76  75 142 143 151 138 141 145 140
 146 139 144 196 199 198 197 201 202 200 193 160 159 157 163 158 161 164
 162 165 166 131 130 132 133 134 129  96  92  93  79  80  77  81  78  91
  94  89]
102 [126 122 124 121 123 118 120 125 132  45  46  44  42  47  41  49  43 179
 181 180 178 177 183 182 186 176 175 208 209 210 205 211 212 104 106 105
 103 101 107 100 108 206 213 207 215 214 194 195 193 192 191 190 196 189
 111 112 114 110 109 113 116 115 170 171 172 169 166 188 187 184 141 142
 138 136 139 140 137 143 148 145 147 144 131 134 130 129 133 128 127]
103 [132 126 123 129 127 124 128 130 133 125 122 134  73  72  74  70  71  75
```

```
 78  77  68  69  51  53  52  50  49  82  83  81  80  86  85  79 194 196
195 191 197 199 198 193  43  40  42  44  41  45  46  39  76  37  38  36
 32  35 150 149 151 141 148 147 202 201 205 200 203 156 157 154 158 155
159 161  47 183 186 184 185 182 188 187 103 104 102 105 101 106 108 107
 97 119 115 118 117 121 120  65  62  61  60  59  63  64  57  67]
104 [220 223 221 225 222 224 219 226 217 218 215 216 214 213 212  32  33  34
 31  36  30 151 150 155 153 152 154 149 186 183 187 184 182 188 185 202
204 203 200 201 199 205 112 111 108 113 115 106 110 116 114 109  35  84
 85  87  86  83  81  82  80  88  54  55  53  52  57  60  56  58  50  51
 41  43  42  46  44  40 196 198 197 194 195 103 104 105 102 107  79  78
 77  76 120 117 118]
105 [ 42  39  44  43  40  41  49  38  46  45  36  47  48 121 122 119 120 123
117 118 124 115  95  98  97  93  94  96  92 152 151 150 153 148 155 149
146  70  71  69  72  74  73  67 180 181 178 179 182 183 171 176 184 207
209 210 211 215 208 136 139 135 132 133 137 134 140  76  75  68  66  78
 77  79  81 109 110 112 108 113 111 107 103 106 105 216 217 218 219 214
192 191 189 186 190 193 188 195 157 156 154 159  34  37  33  35]
106 [180 171 178 179 177 174 176 181 182 185  98 101  95  96  97 100  99  94
124 128 122 123 125 120 126 129 119  32  33  31  30  34 113 112 114 117
115 111 116  39  35  37  36 155 156 157 159 158 151 154 148  61  63  62
 60  64  67  66 143 141 142 140 133 145 139 144  82  80  79  81  78  86
 83  77  76  72  70  73  71  74  75  69 110 107 108 186 189 188 187 183
184  41  38  40  44  43  42  85 190 191 192 193 195]
107 [205 203 202 214 204 207 206 199 194 209 201 196 200  69  68  67  65  70
 66  72  62  73  84  87  85  83  82  81  79  56  55  57  54  51  52  60
 53  58 104 107 108  99 103 105 101 106 102  71  64  92  91  90  97  89
 95  98  94 208 210 211 221 223 220 219 222 218 224 225 226 181 182 180
178 179 183 109 100 142 141 139 143 136 140 111 110]
108 [ 95  98  96 108  92  97  99  94  93  90  91  56  58  59  57  54  60  55
 63  53 122 118 124 123 121 119 120 116  78  79  77  80  81  82  76  61
 62 113 115 117 114 110  50  51  46  47  48  49  52 211 210 209 207 212
208 214 215 216 213 220 217  73  83  75 107 109 111 112 106 103 105  64
 65  66 151 150 152 148 149 147  89 202 201 203 206 200 198 204 154 155
156 153 159 158]
109 [194 195 198 196 199 192 193 189 191 197 185 188 187  68  66  67  65  70
 64  71  54  53  55  56  57  52  50 190 113 111 112 115 116 114 110 139
140 138 136 137 141 119 118 117 120 121 129 123  51  48  49  47 169 168
170 167 171 172 174  42  41  36  39  40  45  44  43  38 135 133 134  88
 89  90  87  86  92  85  91 161 162 160 163 164 159 178 179 177 181 180
176 173 175 165 211 210 213 212 207 214 215 208 209]
110 [221 225 219 222 220 223 226 218 217 224 215 230  98  97  96  99 100  95
 94 210 209 213 212 211 207 208  70  68  67  72  71  69  74 164 166 161
163 165 162 167 118 120 119 117 122 121 116 160 159 158 157 155 214 216
151 153 154 156 169 170 168 171 176 114 111 109 113 112 115 108 106 185
186 184 183 188 187 190  75  73  76 199 200 195 203 194 201 198 196 197
204 202]
111 [219 224 220 221 222 223 218 225 217 213 175 174 178 172 176 173 177 180
182 167 168 169 166 165 170 164 161 171 153 154 155 152 156 151  36  37
```

```
  35  38  34  39  33 184 183 181 179 188 185 150 149 148 147  77  78  79
  89  76  80  73  75  74  81 195 193 194 191 198 196 192 199  72  70  71
  69  64  65  63  66  67  62  61  58  68 124 123 122 126 128 120 121 118
  90  88  86  85  87  91  84 141 137 143 135 138 139 140 144 117 116 115
 111 119 114 112]
112 [217 219 220 218 225 221 212 213 214 216 215 113 112 114 117 111 110 108
 115 109 181 182 183 185 180 184 178 179 177 168 169 167 170 166 171  59
  60  61  58  62  57  63 160 161 159 162 158 157 165 164 163  51  50  49
  52  48  47  53  55  65  64  66  56 143 144 142 145 139 146 140 141 150
  83  84  85  78  87  82 135 136 134 138 137 132 131  73  74  77  72  70
  76  75  98  94  96  95  91  92  93  97  89 100]
113 [117 118 119 121 114 116 126 120 113 115 107 104 106 110 105 109 102 108
 173 174 172 175 170 176  76  75  77  74  78  79  73  72 194 193 195 192
 198 196 199 191 133 134 132 130 129 137 135 128 141 203 204 205 206 202
 201  98  97 101 100  99  96  95  94 103 207 209 208 158 159 157 161 156
 160 163 124 125 127 122 145 144 142 143 147 146 213 212 210 211 216 215
 217 149 148 140  93  92]
114 [ 58  59  60  63  57  62  68  61  64  70  69  38  37  36  39  35  40  42
  32  33  41 195 191 196 194 198 197  48  47  49  51  50 133 132 134 135
 129 130 131 136 147 143 148 152 146 149 141 150  95  94  96  97  99  93
 100  98  91 172 171 170 173 174 168 169 125 126 124 119 123 127 122 128
  46  43  44  45 160 161 162 164 157 159 165 163 145 144 101 206 208 205
 207 204 211 203 210 202 139 121 120 116 117]
115 [195 193 194 190 192 189 191 200 201 169 168 167 170 166 171 165  46  48
  45  47 152 150 154 153 149 151 155  33  35  36  34  37  32  29  38  39
 141 139 140 142 143 136 138 135 146 133  40  41  44 210 209 212 208 205
 211 207 178 182 180 181 173 183 179 177  82  83  84  85  86  81  79  80
  30  31 198 199 197 202  69  68  67  70  71  66  76  75  72 215 217 214
 226 216 213  87  89  88  90  91]
116 [ 95  96  97  93  98  94  91 101 100  99 165 166 164 168 167 163 162 171
 161 169 203 204 202 201 206 199 200 205  56  57  54  55  58  60  59  61
 135 134 137 136 133 139 132 138 141 129 128 130 131 127 122 126 125 153
 154 152 151 150 155 149 123 124  77  78  76  67  79  82  75  80 190 187
 188 184 191 189 192 185 182 186  92 207 209 208 212 210  81  85  83 196
 183 193]
117 [215 219 224 217 218 222 216 220 221 226 212 122 121 117 123 120 116 124
 125 118 119 115 113 114 112 147 148 146 151 145  94  96  95  98  92  97
  93  91 126 127 128 130  74  75  76  73  71  70  79  77 150 149 153 152
  57  58  60  56  61  62  55  59  63 177 178 174 176 173 175 179 172 180
  46  48  47  45 196 197 198 193 199 190 195 194 201 170 169 171 181  78
  80  81  82  83]
118 [199 198 197 203 207 195 200 201 204 209 196 202 188 115 116 117 118 114
 122 113 112 119  78  77  76  79  75  81  80 220 219 216 218 222 221 215
 217 194 193 191 192 189 190 187 186 184  71  69  70  73  72  63  67  74
 121 120 123  42  41  34  43  44  45  36  40  51  39  48 140 137 139 138
 143 126 134 136 132 135 133 141 214 223 213  84  83  85  82  87 161 164
 167 163 165 162 166 158 160 174 168]
119 [218 222 219 217 212 220 221 214 215 224 116 117 114 119 120 115 118  45
```

```
  42  43  46  44  47 113 101  99  98 102 100 105 103 104 197 193 198 199
 196 194 195 201 200 187 185 188 189 192 186 180 184 190 182 183 150 151
 149 148 146 152 147 168 167 162 165 170 166 171 172 169  96  97  94  91
  93  95  92 153 154 155 157 211 210 213 206 207 204 209  87  86  85  84
  88  90  83  82  61  59  57  56  58  60  53  62  55  54  52 175 177]
120 [182 184 178 181 192 185 183 180 186 189 176 179 133 135 134 132 136 131
 129 130 127 137 128 100 101  99  97 103  98 102  81  83  80  85  82  79
  78  87  84  88  86  89 210 208 209 211 207 212 214 215 217 216 219 213
 156 157 149 155 153 154 151 159 162 158 218 220 221 222 105 106 108 107
 104 110  44  43  45  46  48  50  51  41  42  47  63  62  64  66  61  65
  59 118 119 117 125 120 114 126 113 121 122 116 123  95  94  93  96  70
  69  67  72  71  68  76  73]
121 [ 53  52  50  47  51  64  54  48  57  59  55  56  49 204 205 206 202 203
 201 207 166 165 167 162 163 164 169  58  60  63  61 212 209 213 211 214
 210 215 218 117 116 115 118 114 176 177 173 175 170 179 174 171 172 178
 112 113 123 109 111 108 110  87  85  86  82  88  90  89 142 141 144 146
 143 138 140 120 119 121 122  62  65  68  66  71 134 133 136 135 131 137
 129 132 125 130 139]
122 [144 148 139 157 146 147 149 151 145 143 150  34  35  32  33  30  37  31
 129 130 127 128 131 134 194 196 191 195 193 192 189 214 216 213 218 215
 212 210 217 190 188 184 185 187 198 132 133 183 186 182 181 177 170 173
 172 171 169 175 176 174 107 108 106 105 110 109  62  60  64  66  63  65
  57  61  67  76  75  74  73  80  77  72  78  79 136 137 135  83  81  82
  86  84  85]
123 [198 196 195 197 182 200 199 194 201 192 202 206 205  41  40  39  37  42
  43  38 135 134 131 136 137 133 138  66  67  63  62  68  65 203 204 207
  34  35  33  32  36 193 191 189 183  58  59  57  60  56  61 177 178 179
 181 180 176 173 139 140 147 141  47  46  52  48  45  44  51  49 208 209
  87  88  89  92  90  91  86  64  69  72 109 110 108 113 114 106 111 119
 123 117 120 118 124 112 115 122 121 116]
124 [ 93  97  92  95  94  99  89  90  96  85  91 101  86 205 204 207 203 206
 208 210 202 201 185 186 188 184 187 170 171 169 166 168 167 165  36  37
  35  34  39  25  38  33 116 115 118 117 114  88  87  83 105 106 109 104
 107 103 113 108 102 133 134 131 135 130 132 137 136 111 112 110 138 139
 141 140  40  32  41  28 149 148 151 146 145 147 150 152 144 157]
125 [ 34  35  38  36  40  37  39  32  41  27  29  33  44  95  98  92  93  94
  96  97  89  91  68  69  67  66  70  63  65  60  61  62  58  59 116 117
 115 114 118 113  75  73  74  79  81  77  76  71 186 184 185 183 143 142
 135 144 146 141 145 140 123 121 122 124 120 126 119  72 192 191 190 193
 194 189 173 174 172 176 171 175  99  86  80  78  53  55  51  52  54  50
  56  57]
126 [ 99 101  96  98 115 100  97  95 102  58  56  57  59  62  54  55  60  61
  33  32  31  30  34  36 210 209 211 208  67  64  66  65  68  69  63  70
 104 103  87  80  92  52  51  53  50  49 168 170 169 166 167 161 171 179
 180 178 176 177 181 182 183 185 188 175 164 165 172 212 213 214 215  37
  35  38  40  39  41  42  43  44  47 151 149 150 143 153 156  94  93  89]
127 [ 79  78  82  80  71  81  75  83  84  76  77  74 195 196 193 191 197 200
 198 190 194 134 135 133 137 132 136 187 188 186 189 185 192 176 175 177
```

```
 178 179 174  87  88  86  91  89  93 201 202 203 204 199 126 125 123 127
 124 130 122 181 184 182 183 180  62  64  63  65  61  66  60  58  67 164
 162 163 165 161 158 166 159 160]
```

2.1 Conclusão:

- Os dados não possuem a necessidade de pré-processamento visto que já estão todos com valores validos

3.0.1 2.3 Análise estatística

[8]:
```
data.corr()
```

[8]:
```
            0         1         2         3         4         5         6    \
0    1.000000 -0.298556  0.099976  0.022622  0.147617  0.121319 -0.126459
1   -0.298556  1.000000 -0.185263  0.432802 -0.292705  0.061192 -0.019780
2    0.099976 -0.185263  1.000000  0.067625  0.362365 -0.028879  0.080121
3    0.022622  0.432802  0.067625  1.000000  0.099326 -0.144059 -0.112927
4    0.147617 -0.292705  0.362365  0.099326  1.000000  0.293487 -0.131999
..        ...       ...       ...       ...       ...       ...       ...
123  0.034045  0.016683 -0.044062 -0.358739  0.033921  0.005289  0.050475
124  0.463972 -0.548772  0.567061 -0.241791  0.189920 -0.258597  0.129884
125 -0.023602  0.217301 -0.184789  0.213103 -0.551513 -0.268762  0.541192
126 -0.188079  0.429549 -0.021832 -0.361639  0.109259  0.058695  0.386794
127  0.311024 -0.073918  0.222362 -0.177917 -0.318805  0.066725  0.368015

            7         8         9    ...       118       119       120  \
0    0.467975  0.279398 -0.138266  ... -0.357315 -0.012444  0.125286
1    0.016283 -0.213111  0.143765  ... -0.173967  0.247016  0.633308
2    0.012888  0.007419  0.235775  ...  0.223319 -0.231223 -0.192837
3    0.266907 -0.429601 -0.028823  ... -0.416790 -0.262267  0.586108
4   -0.170449  0.577666 -0.063625  ...  0.227436 -0.035686  0.014784
..        ...       ...       ...  ...       ...       ...       ...
123 -0.220242  0.284786 -0.384904  ... -0.014952  0.285386 -0.044914
124  0.046233  0.049035  0.115393  ...  0.125422 -0.340187 -0.507953
125  0.318943 -0.447063  0.044297  ... -0.288177  0.100666  0.075201
126 -0.197502  0.433910  0.145782  ...  0.329098  0.320879 -0.103570
127  0.154835  0.012821  0.257718  ...  0.127201  0.133552 -0.133041

          121       122       123       124       125       126       127
0    0.218881 -0.253469  0.034045  0.463972 -0.023602 -0.188079  0.311024
1    0.059660  0.505386  0.016683 -0.548772  0.217301  0.429549 -0.073918
2    0.184547  0.445270 -0.044062  0.567061 -0.184789 -0.021832  0.222362
3   -0.148362  0.295559 -0.358739 -0.241791  0.213103 -0.361639 -0.177917
4   -0.217203  0.237352  0.033921  0.189920 -0.551513  0.109259 -0.318805
..        ...       ...       ...       ...       ...       ...       ...
123  0.278613  0.218825  1.000000 -0.225108 -0.049457  0.195657 -0.273943
124  0.413947 -0.233779 -0.225108  1.000000 -0.204440 -0.045870  0.515130
```

```
125 -0.131423 -0.175185 -0.049457 -0.204440  1.000000  0.001906  0.355830
126  0.085155  0.183157  0.195657 -0.045870  0.001906  1.000000  0.143927
127  0.203967 -0.040931 -0.273943  0.515130  0.355830  0.143927  1.000000

[128 rows x 128 columns]
```

3.0.2 2.4 Escalonando

Para aplicação dos algoritmos escalona-se os dados afim de parametriza-los num certo intervalor (-1 a 1)

[9]:
```
scaler = preprocessing.StandardScaler()
data_scaler = scaler.fit_transform(X = data)
```

[10]:
```
data_scaler
```

[10]:
```
array([[ 0.38555573, -0.16597321, -0.06153205, ..., -1.34082722,
        -0.18685133, -1.46250099],
       [ 0.46363525, -0.04211321,  0.05936751, ..., -1.3204906 ,
        -0.18685133, -1.48277088],
       [ 0.502675  , -0.12468654,  0.01906766, ..., -1.25948071,
        -0.15385672, -1.48277088],
       ...,
       [ 1.04923161, -2.12708994,  0.30116663, ..., -0.93409467,
        -0.28583517,  0.159091  ],
       [ 0.99067197, -2.06515993,  0.2810167 , ..., -0.87308479,
        -0.20334864,  0.2401706 ],
       [ 0.91259246, -1.96194326,  0.22056692, ..., -0.93409467,
        -0.3518244 ,  0.2199007 ]])
```

[11]:
```
data_scaled = pd.DataFrame(data_scaler)
data_scaled.head()
```

[11]:
```
          0         1         2         3         4         5         6    \
0  0.385556 -0.165973 -0.061532  0.024084  1.626257  1.434379 -1.174373
1  0.463635 -0.042113  0.059368  0.075840  1.717307  1.434379 -1.057131
2  0.502675 -0.124687  0.019068 -0.053551  1.662677  1.411927 -1.135292
3  0.444115 -0.186617  0.039218  0.024084  1.608047  1.434379 -1.174373
4  0.405076 -0.104043  0.039218  0.024084  1.626257  1.501734 -1.076671

          7         8         9    ...       118       119       120       121  \
0  0.222800  1.441096  0.093981  ...  0.983934  1.505823  0.876413 -1.588549
1  0.161421  1.441096 -0.025321  ...  0.965536  1.577474  0.876413 -1.607574
2  0.140961  1.441096  0.034330  ...  0.965536  1.505823  0.876413 -1.607574
3  0.202340  1.460442  0.074098  ...  0.947138  1.523736  0.910643 -1.645623
4  0.263719  1.479787  0.054214  ...  0.983934  1.487910  0.876413 -1.607574

        122       123       124       125       126       127
0  0.147553  1.323391 -0.658782 -1.340827 -0.186851 -1.462501
1  0.224691  1.323391 -0.573106 -1.320491 -0.186851 -1.482771
```

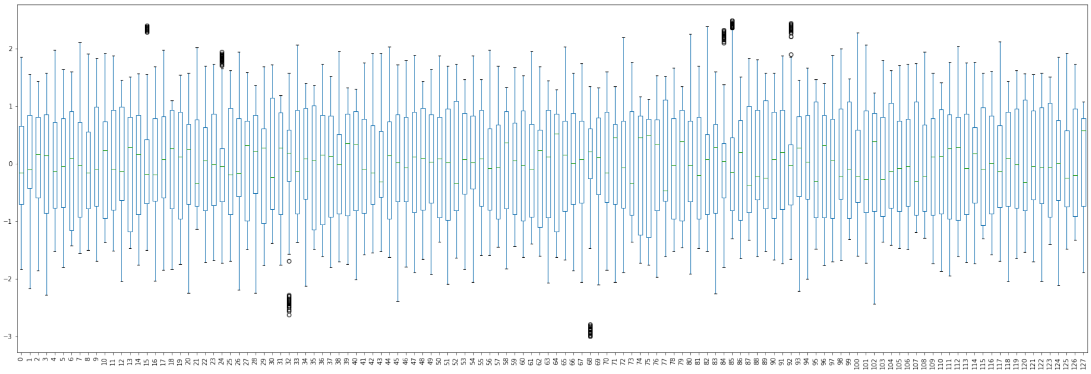
```
2  0.147553  1.290578 -0.658782 -1.259481 -0.153857 -1.482771
3  0.147553  1.323391 -0.680201 -1.300154 -0.153857 -1.401691
4  0.224691  1.323391 -0.615944 -1.300154 -0.236343 -1.442231

[5 rows x 128 columns]
```

3.0.3 2.5 Plotando boxsplot

Pelo boxsplot é possivel visualizar que há alguns outliers.

[12]:
```
data_scaled.plot(kind = 'box', figsize=(30,10), rot=90, )
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f78b6ef0470>
```



4 3. Clustering

4.1 3.1 Dataset Completo

4.1.1 3.1.1 K-Means

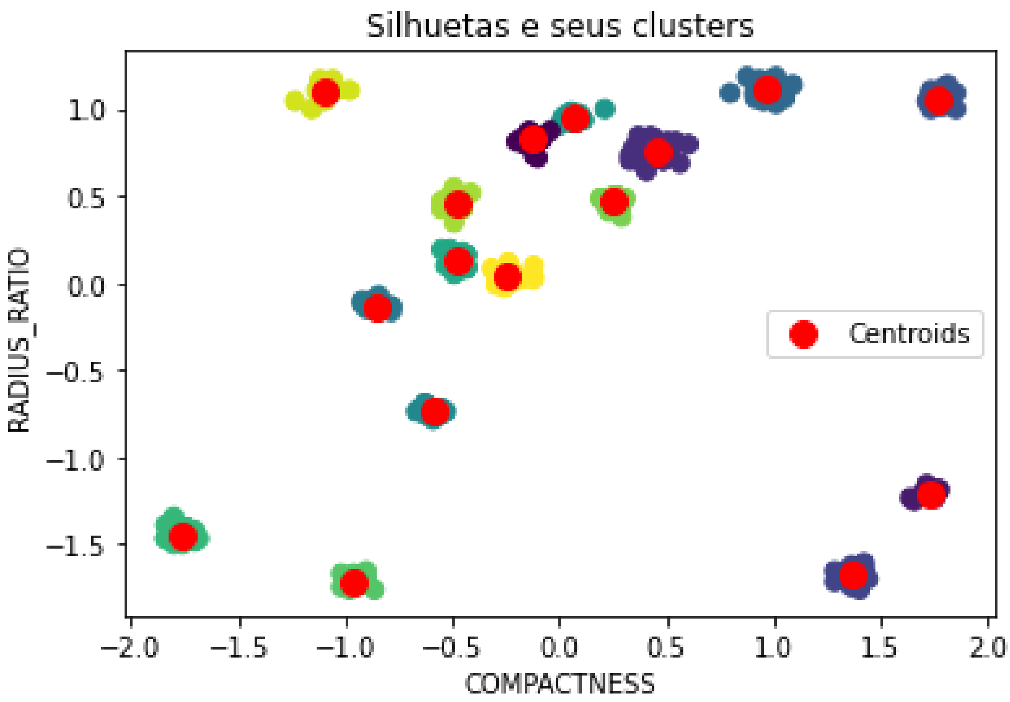
[13]:
```
data_kmeans = data_scaled.copy()
```

[14]:
```
kmeans = KMeans(n_clusters = 16, init = 'random')
kmeans.fit(data_kmeans)
```

[14]:
```
KMeans(algorithm='auto', copy_x=True, init='random', max_iter=300,
       n_clusters=16, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

[15]:
```
plt.scatter(data_scaler[:,0], data_scaler[:,31], s = 50, c = kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 31], s =␣
 ↪100, c = 'red',label = 'Centroids')
plt.title('Silhuetas e seus clusters')
plt.xlabel('COMPACTNESS')
plt.ylabel('RADIUS_RATIO')
plt.legend()
```

```
plt.show()
```



4.1.2 3.1.2 Agglomerative Clustering

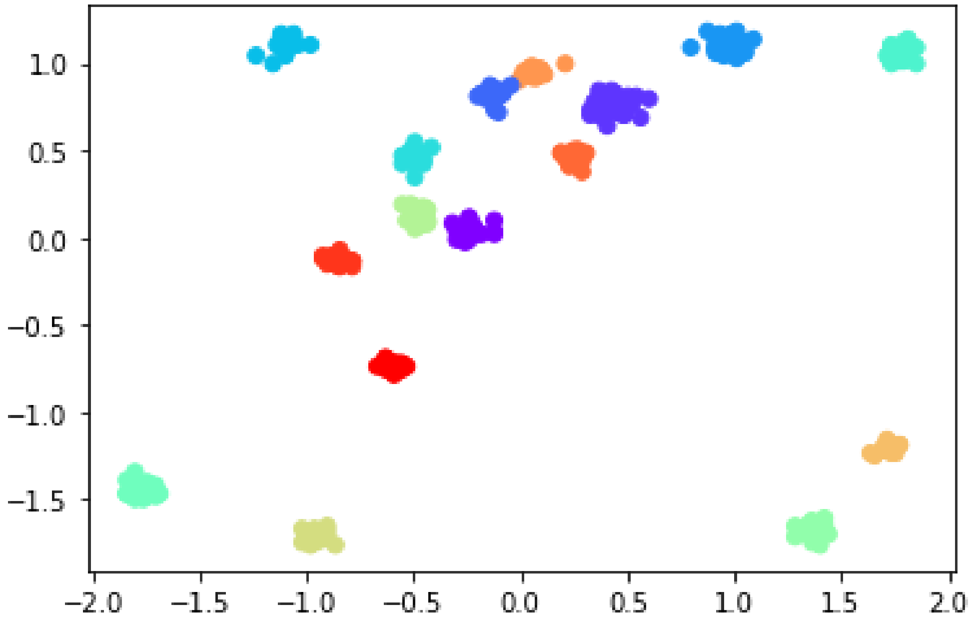
[16]:
```
data_agglo = data_scaled.copy()
```

[17]:
```
agglo = AgglomerativeClustering(n_clusters=16, linkage='ward')
agglo.fit(data_agglo)
```

[17]:
```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
                        connectivity=None, distance_threshold=None,
                        linkage='ward', memory=None, n_clusters=16)
```

[18]:
```
plt.scatter(data_scaler[:,0],data_scaler[:,31], c=agglo.labels_, cmap='rainbow')
```

[18]:
```
<matplotlib.collections.PathCollection at 0x7f78b3da7f28>
```



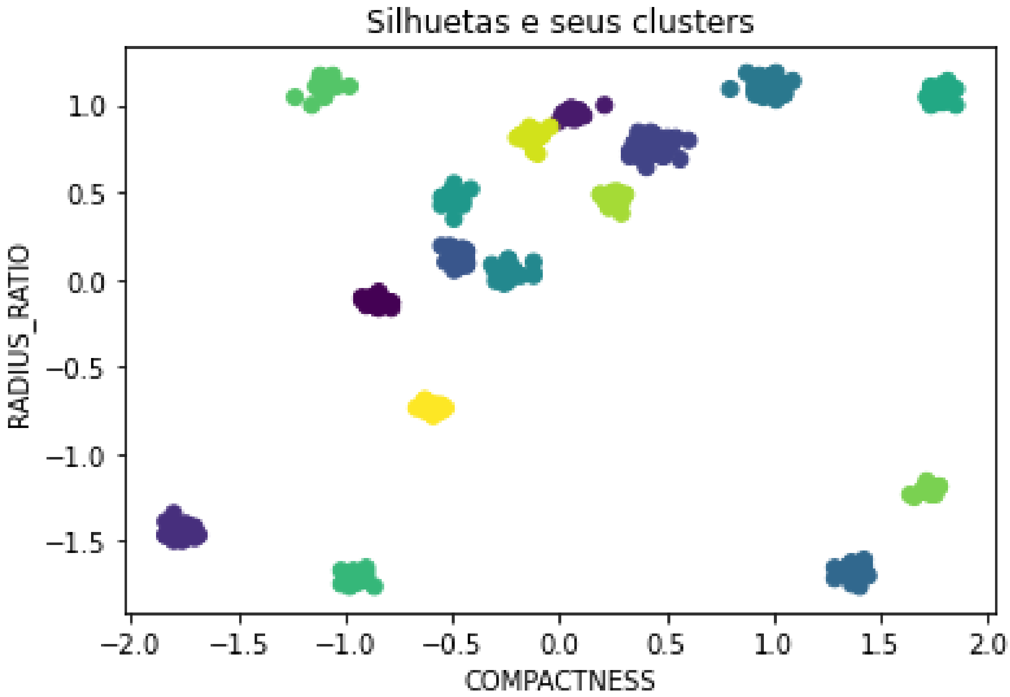
4.1.3 3.1.3 Spectral Clustering

[19]:
```
data_spectral = data_scaled.copy()
```

[20]:
```
spectral = SpectralClustering(n_clusters=16)
spectral.fit(data_spectral)
```

[20]:
```
SpectralClustering(affinity='rbf', assign_labels='kmeans', coef0=1, degree=3,
                   eigen_solver=None, eigen_tol=0.0, gamma=1.0,
                   kernel_params=None, n_clusters=16, n_components=None,
                   n_init=10, n_jobs=None, n_neighbors=10, random_state=None)
```

[21]:
```
plt.scatter(data_scaler[:,0], data_scaler[:,31], c = spectral.labels_)
plt.title('Silhuetas e seus clusters')
plt.xlabel('COMPACTNESS')
plt.ylabel('RADIUS_RATIO')
plt.show()
```



4.2 3.2 Dataset com atributos selecionados

[22]:
```
data_reduzida = pd.DataFrame(SelectKBest(chi2, k=4).fit_transform(data, label))
data_reduzida.shape

data_scaler2 = scaler.fit_transform(X = data_reduzida)
```

[23]:
```
data_scaler2
```

[23]:
```
array([[-1.35614195, -0.7651429 ,  1.18344795,  1.12525029],
       [-1.32671589, -0.74905183,  1.12225264,  1.12525029],
       [-1.29728983, -0.7651429 ,  1.06105733,  1.09601708],
       ...,
       [-0.72348165, -0.95823573, -1.15727261, -0.48257639],
       [-0.70876862, -0.92605359, -1.18787027, -0.48257639],
       [-0.76762074, -0.90996252, -1.15727261, -0.39487675]])
```

[24]:
```
data_scaled2 = pd.DataFrame(data_scaler2)
data_scaled2.head()
```

[24]:

	0	1	2	3
0	-1.356142	-0.765143	1.183448	1.125250
1	-1.326716	-0.749052	1.122253	1.125250
2	-1.297290	-0.765143	1.061057	1.096017
3	-1.341429	-0.861689	1.168149	1.125250

```
4 -1.312003 -0.877780  1.183448  1.110634
```

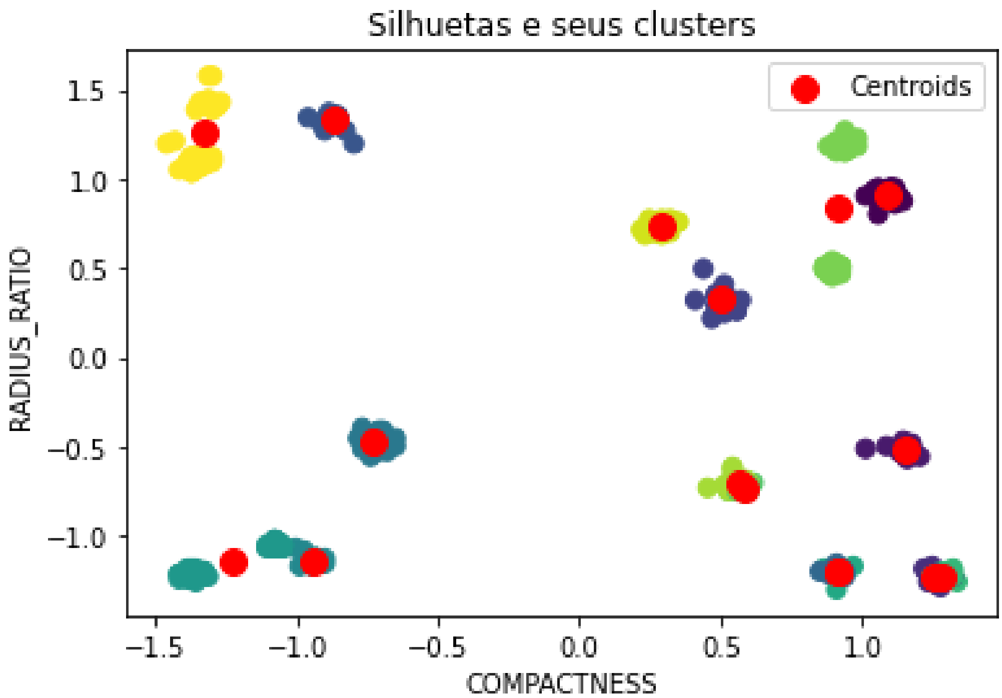
4.2.1 3.2.1 K-Means

[25]:
```python
data_kmeans2 = data_scaled2.copy()
```

[26]:
```python
kmeans2 = KMeans(n_clusters = 16, init = 'random')
kmeans2.fit(data_kmeans2)
```

[26]:
```
KMeans(algorithm='auto', copy_x=True, init='random', max_iter=300,
       n_clusters=16, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

[27]:
```python
plt.scatter(data_scaler2[:,0], data_scaler2[:,3], s = 50, c = kmeans2.labels_)
plt.scatter(kmeans2.cluster_centers_[:, 0], kmeans2.cluster_centers_[:, 3], s =␣
 ↪100, c = 'red',label = 'Centroids')
plt.title('Silhuetas e seus clusters')
plt.xlabel('COMPACTNESS')
plt.ylabel('RADIUS_RATIO')
plt.legend()
plt.show()
```



4.2.2 3.2.2 Agglomerative Clustering

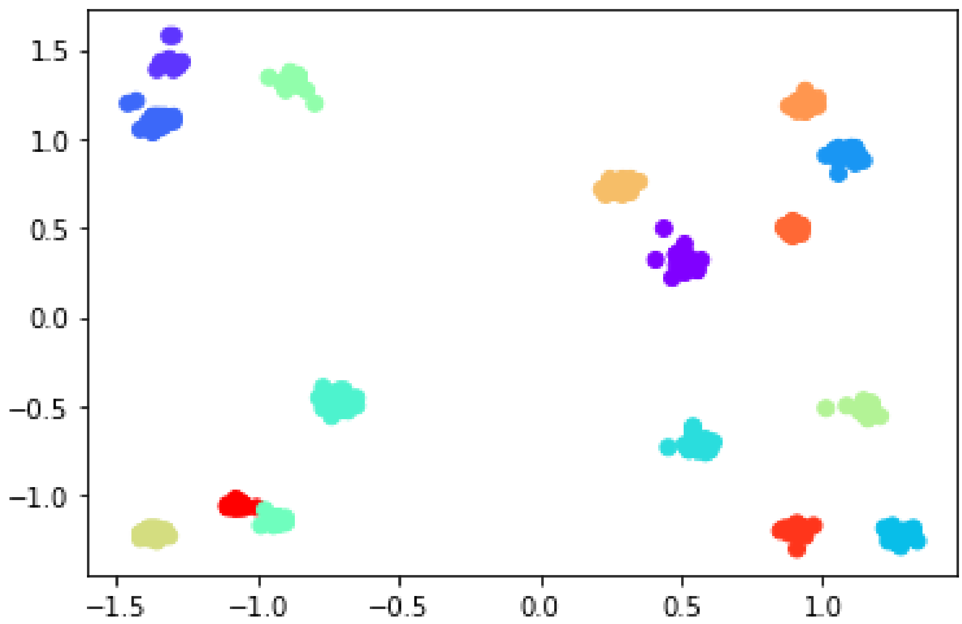
[28]:
```
data_agglo2 = data_scaled2.copy()
```

[29]:
```
agglo2 = AgglomerativeClustering(n_clusters=16, linkage='ward')
agglo2.fit(data_agglo2)
```

[29]:
```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
                        connectivity=None, distance_threshold=None,
                        linkage='ward', memory=None, n_clusters=16)
```

[30]:
```
plt.scatter(data_scaler2[:,0],data_scaler2[:,3], c=agglo2.labels_,␣
↪cmap='rainbow')
```

[30]:
```
<matplotlib.collections.PathCollection at 0x7f78b3d6e1d0>
```



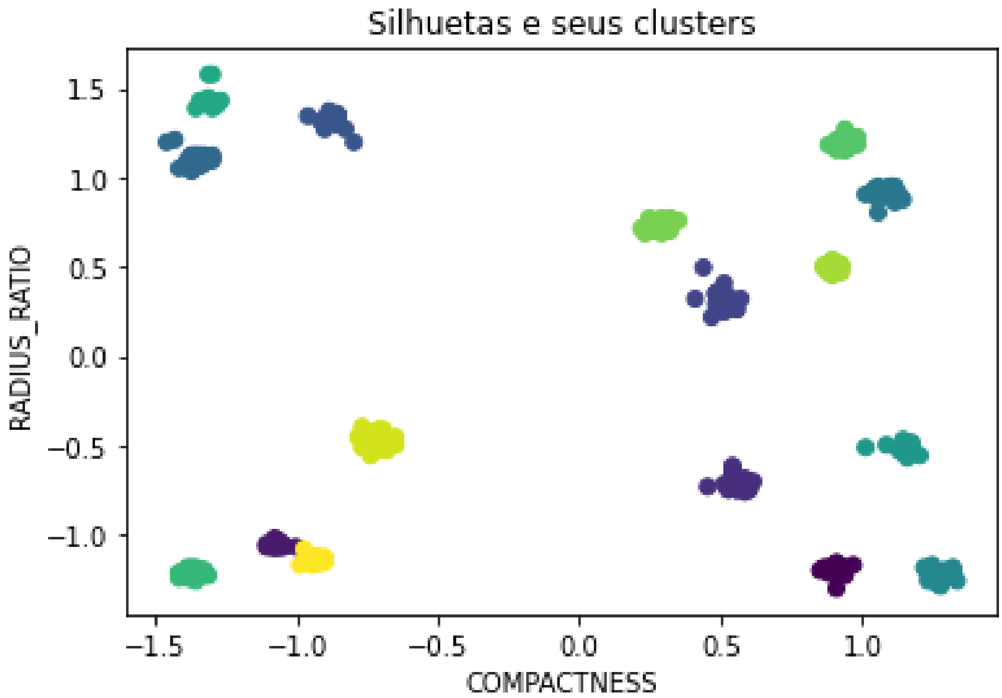
4.2.3 3.2.3

[31]:
```
data_spectral2 = data_scaled2.copy()
```

[32]:
```
spectral2 = SpectralClustering(n_clusters=16)
spectral2.fit(data_spectral2)
```

[32]:
```
SpectralClustering(affinity='rbf', assign_labels='kmeans', coef0=1, degree=3,
                   eigen_solver=None, eigen_tol=0.0, gamma=1.0,
                   kernel_params=None, n_clusters=16, n_components=None,
                   n_init=10, n_jobs=None, n_neighbors=10, random_state=None)
```

[33]:
```
plt.scatter(data_scaler2[:,0], data_scaler2[:,3], c = spectral2.labels_)
plt.title('Silhuetas e seus clusters')
plt.xlabel('COMPACTNESS')
plt.ylabel('RADIUS_RATIO')
plt.show()
```



5 4. Avaliação

[34]:
```
lista = np.array(label[0].tolist())
```

[35]:
```
for i in lista:
    lista[i] = lista[i] - 1
```

5.0.1 4.1.1 KMeans - Completo

[36]:
```
dataset = data.values

class Data:
    namostras = 0
    ndim = 0
    ncluster = 0
```

```python
newData = Data()

newData.namostras = len(data)
newData.ndim = len(data.columns)
newData.ncluster = 16


labels_true = lista

# predict recebe os rotulos preditos pelo algoritmo de clustering
predict = rotulos(kmeans.cluster_centers_, 16, dataset, newData)
```

[37]:
```python
# labels_predict sao as labels ja organizadas para comparacao correta com os␣
↪rotulos originais do conjunto de dados
labels_predict = labelmatch(labels_true,predict,newData.ncluster)
```

[38]:
```python
# METRICAS PARA AVALIACAO DO CLUSTERING
cft = confusion_matrix(labels_true, labels_predict)
hbt = calinski_harabasz_score(dataset,labels_predict)
arit = adjusted_rand_score(labels_true, labels_predict)
amit = adjusted_mutual_info_score(labels_true, labels_predict)
f1t = f1_score(labels_true, labels_predict, average='macro')
accurracyt =accuracy_score(labels_true, labels_predict)
silhouettet = silhouette_score(dataset, labels_predict)

print('Confusion Matrix: \n', cft)
print('\nCalinski-Harabaz Score: ',hbt)
print('\nAdjusted-Rand Score: ',arit)
print('\nAdjusted Mutual Info Score: ',amit)
print('\nF1 Score: ',f1t)
print('\nAccuracy Score: ',accurracyt)
print('\nSilhouette Score: ',silhouettet)
```

```
Confusion Matrix:
 [[ 0  0  0 15  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 47  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0]
```

```
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0]
 [ 0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]]

Calinski-Harabaz Score:  86413.56951870107

Adjusted-Rand Score:  0.9861230349382982

Adjusted Mutual Info Score:  0.9918227743739396

F1 Score:  0.7814129919393078

Accuracy Score:  0.9208984375

Silhouette Score:  0.9746405449945033
```

5.0.2 4.1.2 KMeans - Selecionado

[39]:
```
dataset = data_reduzida.values

class Data:
    namostras = 0
    ndim = 0
    ncluster = 0

newData = Data()

newData.namostras = len(data_reduzida)
newData.ndim = len(data_reduzida.columns)
newData.ncluster = 16


labels_true = lista
```

[40]:
```
# predict recebe os rotulos preditos pelo algoritmo de clustering
predict = rotulos(kmeans2.cluster_centers_, 16, dataset, newData)

# labels_predict sao as labels ja organizadas para comparacao correta com os␣
 ↪rotulos originais do conjunto de dados
labels_predict = labelmatch(labels_true,predict,newData.ncluster)

# METRICAS PARA AVALIACAO DO CLUSTERING
cft = confusion_matrix(labels_true, labels_predict)
hbt = calinski_harabasz_score(dataset,labels_predict)
arit = adjusted_rand_score(labels_true, labels_predict)
amit = adjusted_mutual_info_score(labels_true, labels_predict)
```

```python
f1t = f1_score(labels_true, labels_predict, average='macro')
accurracyt =accuracy_score(labels_true, labels_predict)
silhouettet = silhouette_score(dataset, labels_predict)

print('Confusion Matrix: \n', cft)
print('\nCalinski-Harabaz Score: ',hbt)
print('\nAdjusted-Rand Score: ',arit)
print('\nAdjusted Mutual Info Score: ',amit)
print('\nF1 Score: ',f1t)
print('\nAccuracy Score: ',accurracyt)
print('\nSilhouette Score: ',silhouettet)
```

```
Confusion Matrix:
 [[ 0  0  0 14  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 47  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]]

Calinski-Harabaz Score:  135.0780266306347

Adjusted-Rand Score:  0.058022033649180176

Adjusted Mutual Info Score:  0.3420899138357844

F1 Score:  0.08757421068633882

Accuracy Score:  0.1708984375

Silhouette Score:  0.11062698140197669
```

5.0.3 4.2.1 Agglomerative Clustering - Completo

[41]:
```
def centroide(data):
    array2 = []
    for valor in range(0,16):
        df_aux = data.loc[data.Label == valor]
        array = []
        for coluna in df_aux:
            array.append(df_aux[coluna].mean())

        array2.append(array)

    return np.array(array2)
```

[42]:
```
data_agglo['Label'] = agglo.labels_
```

[43]:
```
centroide_hieraquico = centroide(data_agglo)
```

[44]:
```
dataset = data.values

class Data:
    namostras = 0
    ndim = 0
    ncluster = 0

newData = Data()

newData.namostras = len(data)
newData.ndim = len(data.columns)
newData.ncluster = 16


labels_true = lista

# predict recebe os rotulos preditos pelo algoritmo de clustering
predict = rotulos(centroide_hieraquico, 16, dataset, newData)

# labels_predict sao as labels ja organizadas para comparacao correta com os␣
↪rotulos originais do conjunto de dados
labels_predict = labelmatch(labels_true,predict,newData.ncluster)


# METRICAS PARA AVALIACAO DO CLUSTERING
cft = confusion_matrix(labels_true, labels_predict)
hbt = calinski_harabasz_score(dataset,labels_predict)
arit = adjusted_rand_score(labels_true, labels_predict)
amit = adjusted_mutual_info_score(labels_true, labels_predict)
f1t = f1_score(labels_true, labels_predict, average='macro')
accuracyt =accuracy_score(labels_true, labels_predict)
```

```python
silhouettet = silhouette_score(dataset, labels_predict)

print('Confusion Matrix: \n', cft)
print('\nCalinski-Harabaz Score: ',hbt)
print('\nAdjusted-Rand Score: ',arit)
print('\nAdjusted Mutual Info Score: ',amit)
print('\nF1 Score: ',f1t)
print('\nAccuracy Score: ',accurracyt)
print('\nSilhouette Score: ',silhouettet)
```

```
Confusion Matrix:
 [[ 0  0  0 15  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 47  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0]
 [ 0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]]

Calinski-Harabaz Score:  86413.56951870107

Adjusted-Rand Score:  0.9861230349382982

Adjusted Mutual Info Score:  0.9918227743739396

F1 Score:  0.7814129919393078

Accuracy Score:  0.9208984375

Silhouette Score:  0.9746405449945033
```

5.0.4 4.2.2 Agglomerative Clustering - Selecionado

[45]:
```
data_agglo2['Label'] = agglo2.labels_
data_agglo2.head()
```

[45]:
```
          0         1         2         3  Label
0 -1.356142 -0.765143  1.183448  1.125250      2
1 -1.326716 -0.749052  1.122253  1.125250      2
2 -1.297290 -0.765143  1.061057  1.096017      2
3 -1.341429 -0.861689  1.168149  1.125250      2
4 -1.312003 -0.877780  1.183448  1.110634      2
```

[46]:
```
centroide_hieraquico2 = centroide(data_agglo2)
```

[50]:
```
dataset = data_reduzida.values

class Data:
    namostras = 0
    ndim = 0
    ncluster = 0

newData = Data()

newData.namostras = len(data_reduzida)
newData.ndim = len(data_reduzida.columns)
newData.ncluster = 16


labels_true = lista

# predict recebe os rotulos preditos pelo algoritmo de clustering
predict = rotulos(centroide_hieraquico2, 16, dataset, newData)

# labels_predict sao as labels ja organizadas para comparacao correta com os␣
↪rotulos originais do conjunto de dados
labels_predict = labelmatch(labels_true,predict,newData.ncluster)


# METRICAS PARA AVALIACAO DO CLUSTERING
cft = confusion_matrix(labels_true, labels_predict)
# hbt = calinski_harabasz_score(dataset,labels_predict)
arit = adjusted_rand_score(labels_true, labels_predict)
amit = adjusted_mutual_info_score(labels_true, labels_predict)
f1t = f1_score(labels_true, labels_predict, average='macro')
accurracyt =accuracy_score(labels_true, labels_predict)
# silhouettet = silhouette_score(dataset, labels_predict)

print('Confusion Matrix: \n', cft)
# print('\nCalinski-Harabaz Score: ',hbt)
```

```
print('\nAdjusted-Rand Score: ',arit)
print('\nAdjusted Mutual Info Score: ',amit)
print('\nF1 Score: ',f1t)
print('\nAccuracy Score: ',accurracyt)
# print('\nSilhouette Score: ',silhouettet)
```

```
Confusion Matrix:
 [[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 15  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 47  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]]

Adjusted-Rand Score:  0.0

Adjusted Mutual Info Score:  0.0

F1 Score:  0.006191950464396284

Accuracy Score:  0.0625
```

5.0.5 4.3.1 Spectral Clustering - Completo

[51]:
```
data_spectral['Label'] = spectral.labels_
data_spectral.head()
```

[51]:
```
          0         1         2         3         4         5         6  \
0  0.385556 -0.165973 -0.061532  0.024084  1.626257  1.434379 -1.174373
1  0.463635 -0.042113  0.059368  0.075840  1.717307  1.434379 -1.057131
2  0.502675 -0.124687  0.019068 -0.053551  1.662677  1.411927 -1.135292
3  0.444115 -0.186617  0.039218  0.024084  1.608047  1.434379 -1.174373
4  0.405076 -0.104043  0.039218  0.024084  1.626257  1.501734 -1.076671
```

```
          7         8         9  ...       119       120       121       122  \
0  0.222800  1.441096  0.093981  ...  1.505823  0.876413 -1.588549  0.147553
1  0.161421  1.441096 -0.025321  ...  1.577474  0.876413 -1.607574  0.224691
2  0.140961  1.441096  0.034330  ...  1.505823  0.876413 -1.607574  0.147553
3  0.202340  1.460442  0.074098  ...  1.523736  0.910643 -1.645623  0.147553
4  0.263719  1.479787  0.054214  ...  1.487910  0.876413 -1.607574  0.224691

        123       124       125       126       127  Label
0  1.323391 -0.658782 -1.340827 -0.186851 -1.462501      3
1  1.323391 -0.573106 -1.320491 -0.186851 -1.482771      3
2  1.290578 -0.658782 -1.259481 -0.153857 -1.482771      3
3  1.323391 -0.680201 -1.300154 -0.153857 -1.401691      3
4  1.323391 -0.615944 -1.300154 -0.236343 -1.442231      3

[5 rows x 129 columns]
```

[52]:
```python
centroide_spectral = centroide(data_spectral)
```

[53]:
```python
dataset = data.values

class Data:
    namostras = 0
    ndim = 0
    ncluster = 0

newData = Data()

newData.namostras = len(data)
newData.ndim = len(data.columns)
newData.ncluster = 16


labels_true = lista

# predict recebe os rotulos preditos pelo algoritmo de clustering
predict = rotulos(centroide_spectral, 16, dataset, newData)

# labels_predict sao as labels ja organizadas para comparacao correta com os␣
 ↪rotulos originais do conjunto de dados
labels_predict = labelmatch(labels_true,predict,newData.ncluster)


# METRICAS PARA AVALIACAO DO CLUSTERING
cft = confusion_matrix(labels_true, labels_predict)
hbt = calinski_harabasz_score(dataset,labels_predict)
arit = adjusted_rand_score(labels_true, labels_predict)
amit = adjusted_mutual_info_score(labels_true, labels_predict)
f1t = f1_score(labels_true, labels_predict, average='macro')
```

```python
accurracyt =accuracy_score(labels_true, labels_predict)
silhouettet = silhouette_score(dataset, labels_predict)

print('Confusion Matrix: \n', cft)
print('\nCalinski-Harabaz Score: ',hbt)
print('\nAdjusted-Rand Score: ',arit)
print('\nAdjusted Mutual Info Score: ',amit)
print('\nF1 Score: ',f1t)
print('\nAccuracy Score: ',accurracyt)
print('\nSilhouette Score: ',silhouettet)
```

```
Confusion Matrix:
 [[ 0  0  0 15  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 47  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 64  0]
 [ 0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]]

Calinski-Harabaz Score:  86413.56951870107

Adjusted-Rand Score:  0.9861230349382982

Adjusted Mutual Info Score:  0.9918227743739396

F1 Score:  0.7814129919393078

Accuracy Score:  0.9208984375

Silhouette Score:  0.9746405449945033
```

5.0.6 4.3.2 Spectral Clustering - Selecionado

[54]:
```
data_spectral2['Label'] = spectral2.labels_
data_spectral2.head()
```

[54]:
```
          0         1         2         3  Label
0 -1.356142 -0.765143  1.183448  1.125250      5
1 -1.326716 -0.749052  1.122253  1.125250      5
2 -1.297290 -0.765143  1.061057  1.096017      5
3 -1.341429 -0.861689  1.168149  1.125250      5
4 -1.312003 -0.877780  1.183448  1.110634      5
```

[55]:
```
centroide_spectral2 = centroide(data_spectral2)
```

[57]:
```
dataset = data_reduzida.values

class Data:
    namostras = 0
    ndim = 0
    ncluster = 0

newData = Data()

newData.namostras = len(data_reduzida)
newData.ndim = len(data_reduzida.columns)
newData.ncluster = 16


labels_true = lista

# predict recebe os rotulos preditos pelo algoritmo de clustering
predict = rotulos(centroide_spectral2, 16, dataset, newData)

# labels_predict sao as labels ja organizadas para comparacao correta com os␣
 ↪rotulos originais do conjunto de dados
labels_predict = labelmatch(labels_true,predict,newData.ncluster)


# METRICAS PARA AVALIACAO DO CLUSTERING
cft = confusion_matrix(labels_true, labels_predict)
# hbt = calinski_harabasz_score(dataset,labels_predict)
arit = adjusted_rand_score(labels_true, labels_predict)
amit = adjusted_mutual_info_score(labels_true, labels_predict)
f1t = f1_score(labels_true, labels_predict, average='macro')
accurracyt =accuracy_score(labels_true, labels_predict)
# silhouettet = silhouette_score(dataset, labels_predict)

print('Confusion Matrix: \n', cft)
# print('\nCalinski-Harabaz Score: ',hbt)
```

```python
print('\nAdjusted-Rand Score: ',arit)
print('\nAdjusted Mutual Info Score: ',amit)
print('\nF1 Score: ',f1t)
print('\nAccuracy Score: ',accurracyt)
# print('\nSilhouette Score: ',silhouettet)
```

```
Confusion Matrix:
 [[ 0  0  0  0  0 15  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 47  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  0  0  0  0  0  0  0  0  0  0  0  0]]

Adjusted-Rand Score:  0.0

Adjusted Mutual Info Score:  0.0

F1 Score:  0.006191950464396284

Accuracy Score:  0.0625
```