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
```

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

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

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

[110]:
```
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()
```

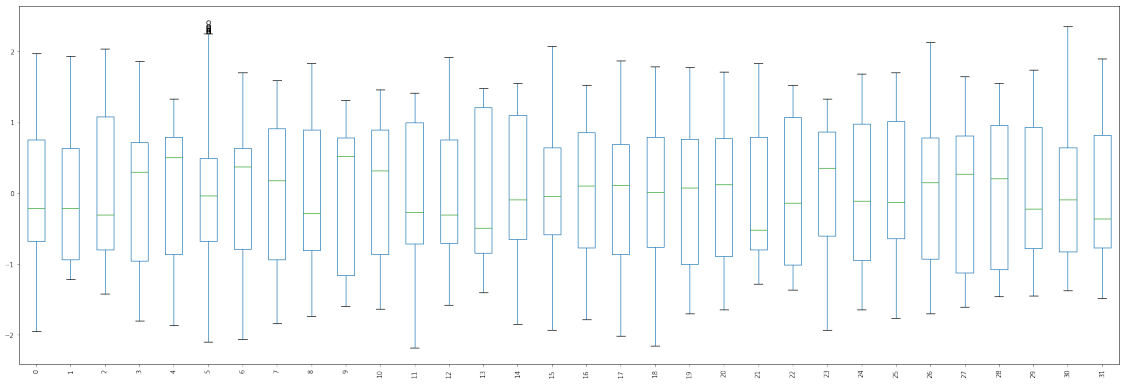
[117]:
```
          0         1         2         3         4         5         6   \
0 -0.346046  0.753920 -0.249457 -1.571379 -0.741871  1.413562  0.588274
1 -0.286521  0.701178 -0.229892 -1.488251 -0.775518  1.286234  0.632086
2 -0.375808  0.701178 -0.269023 -1.592161 -0.725048  1.439028  0.588274
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

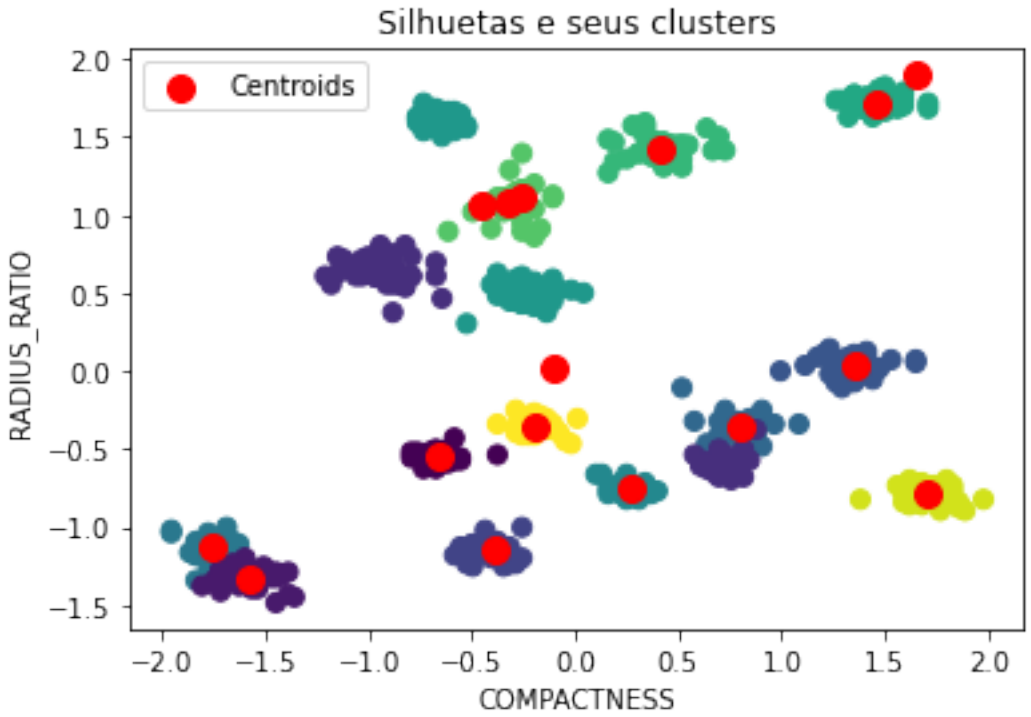
4.1.1 3.1.1 K-Means

[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

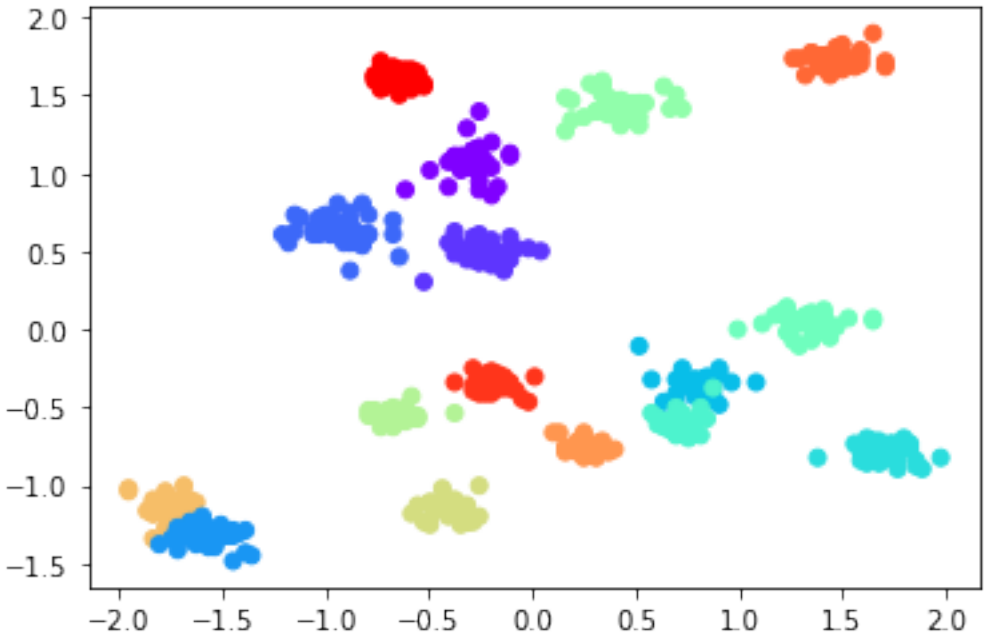
[122]:
```
data_agglo = data_scaled.copy()
```

[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

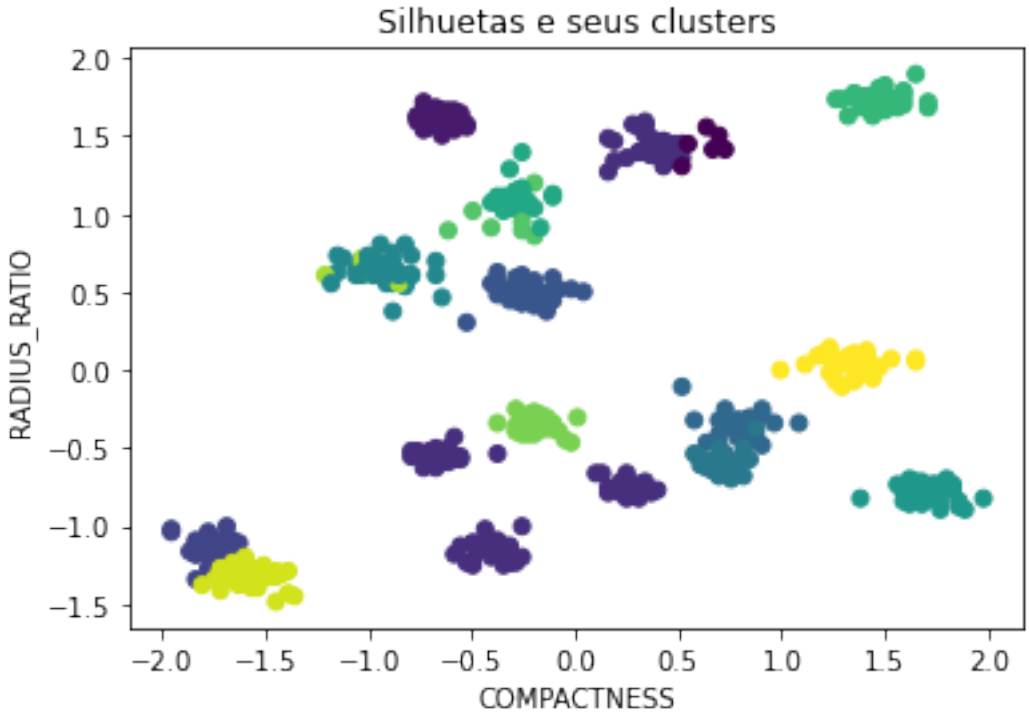
[125]:
```
data_spectral = data_scaled.copy()
```

[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

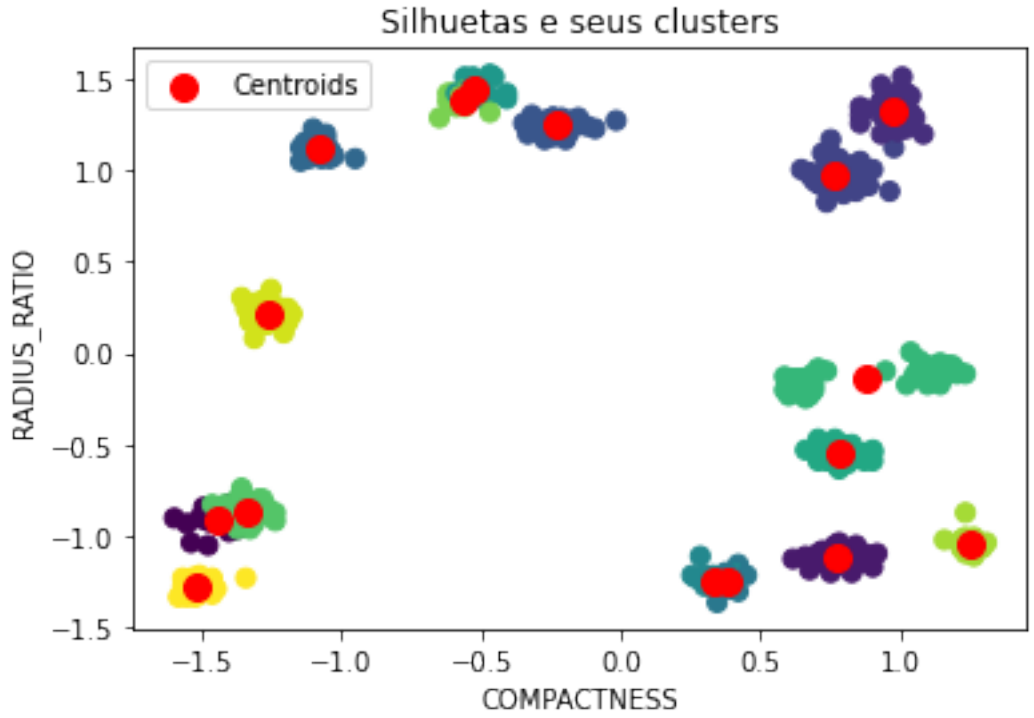
4.2.1 3.2.1 K-Means

[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

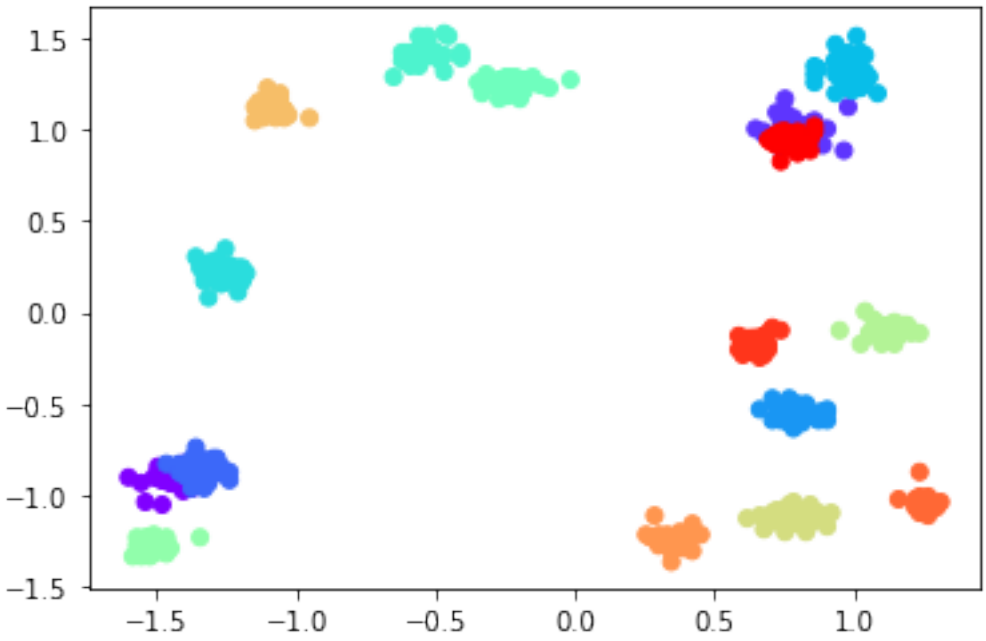
[134]:
```
data_agglo2 = data_scaled2.copy()
```

[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

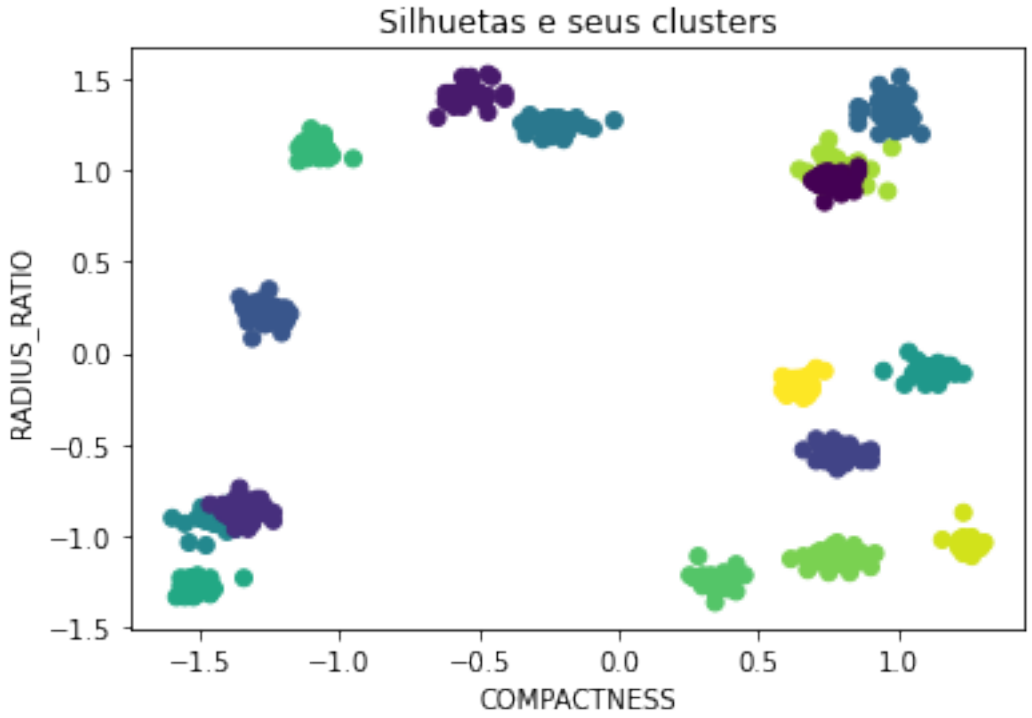
[137]:
```
data_spectral2 = data_scaled2.copy()
```

[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]:
```
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]:
```
# 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)
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  1  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 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 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 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 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  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 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 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]]

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)
```

```
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  1  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 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 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  0  0  0  0  0  0 64  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 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  3 61  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 64  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]]

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

[159]:
```
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
```