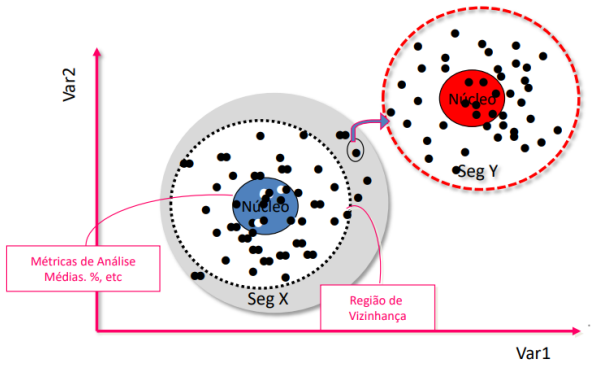
Aula 4 – Cluster Analysis

terça-feira, 24 de outubro de 2023 19:33

Arquivo e pyib

Variáveis segmentadoras - Variáveis que conseguem dividir um grupo

ANÁLISE DE AGRUPAMENTOS
CLUSTER ANALYSIS



ANÁLISE DE AGRUPAMENTOS
CLUSTER ANALYSIS

Padronização das variáveis :

Os métodos baseados em distância são afetados pela diferença de escala entre os valores das variáveis/atributos, sendo necessário normalizar os atributos.

Base Original

id	Salário	Idade
1	16.284	47
2	3.500	22
3	13.751	24
4	4.751	24
5	6.751	25
6	8.750	26

	Salário	Idade
Média	8.539,5	31,1
Desvio	4.716,4	9,3

Base com Padronização da Variáveis

id	Salário	idade
1	1,64	1,71
2	-1,07	-0,97
3	1,10	-0,76
4	-0,80	-0,76
5	-0,38	-0,65
6	0,04	-0,54

Importante padronizar as medidas. Veja nesse exemplo que a "distância" entre salários é extremamente superior à distância entre Idades

ANÁLISE DE AGRUPAMENTOS
MÉTODO HIERÁRQUICO

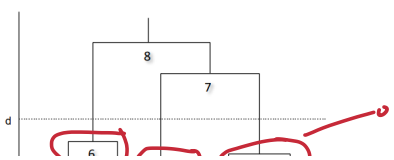
Observação	Valor
a	2
b	8
c	1
d	15
e	3
f	11
g	13
h	7
i	10

Distância=$|x_i-y_i|$

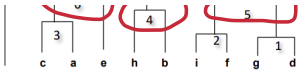
	a	b	c	d	e	f	g	h	i
a	0	6	1	13	1	9	11	5	8
b		0	7	7	5	3	5	1	2
c			0	14	2	10	12	6	9
d				0	12	4	2	8	5
e					0	8	10	4	7
f						0	2	4	1
g							0	6	3
h								0	3
i									0

Exemplo

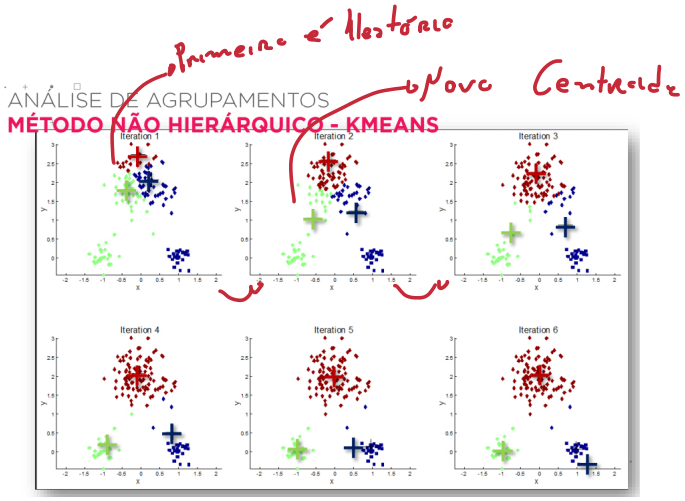
- **Dendograma** – Representação Gráfica de Agrupamento Aglomerativo



3 Clusters

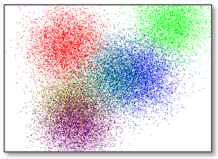


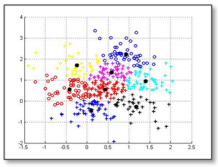
Exemplo



Exercício Python - Clusterização países por recordes em provas de corrida

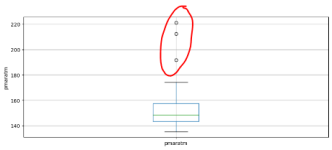
<<Arquivo e pyib.zip>>





Agrupamento (Clusterização): **Consiste em segmentar os registros do conjunto de dados em subconjuntos ou clusters, de tal forma que os elementos de um cluster compartilhem propriedades comuns que os distingam de elementos nos demais clusters. O objetivo nesta tarefa é maximizar a similaridade intracluster e minimizar a similaridade intercluster.**

Outliers



In [9]:
```
# Excluindo outliers e selecionando somente as variáveis segmentadoras
aux = dados[dados['pmaratm'] < 180]
dados_semout = aux.drop(['PAIS'], axis=1)
dados_semout.describe()
```

Out[9]:

	p100ms	p200ms	p400ms	p800mm	p1500mm	p3000mm	pmaratm
count	51.000000	51.000000	51.000000	51.000000	51.000000	51.000000	51.000000
mean	11.303137	22.987451	51.652157	2.007843	4.144510	8.939804	150.399804
std	0.326224	0.764214	2.142184	0.063853	0.186089	0.501017	9.325974
min	10.490000	21.340000	47.600000	1.890000	3.840000	8.100000	135.250000
25%	11.110000	22.520000	49.895000	1.965000	3.995000	8.535000	143.450000
50%	11.310000	22.920000	51.560000	2.000000	4.100000	8.810000	148.270000
75%	11.495000	23.365000	52.940000	2.060000	4.270000	9.250000	154.800000
max	12.130000	25.100000	56.230000	2.150000	4.540000	10.070000	174.180000

3 dados foram dropados

Correlações

É possível fazer clusterização com variáveis categóricas, mas são técnicas muito específicos

```
In [10]: dados_semout.corr()
```

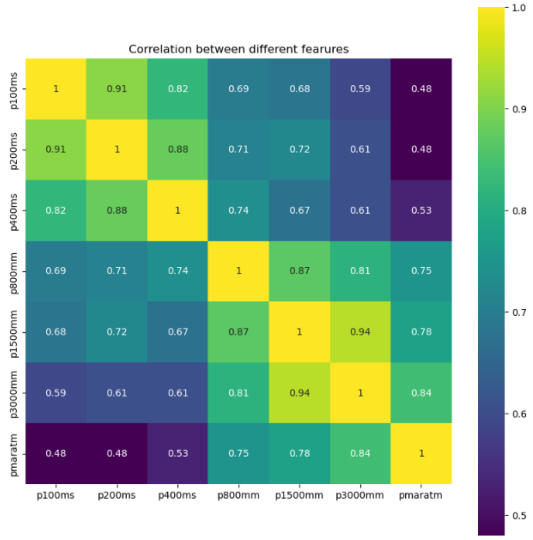
Out[10]:

	p100ms	p200ms	p400ms	p800mm	p1500mm	p3000mm	pmaratm
p100ms	1.000000	0.908808	0.822025	0.694890	0.682289	0.593126	0.480142
p200ms	0.908808	1.000000	0.878632	0.707706	0.723509	0.607023	0.480022
p400ms	0.822025	0.878632	1.000000	0.737465	0.673253	0.607387	0.530104
p800mm	0.694890	0.707706	0.737465	1.000000	0.868839	0.813762	0.748180
p1500mm	0.682289	0.723509	0.673253	0.868839	1.000000	0.944559	0.777070
p3000mm	0.593126	0.607023	0.607387	0.813762	0.944559	1.000000	0.839591
pmaratm	0.480142	0.480022	0.530104	0.748180	0.777070	0.839591	1.000000

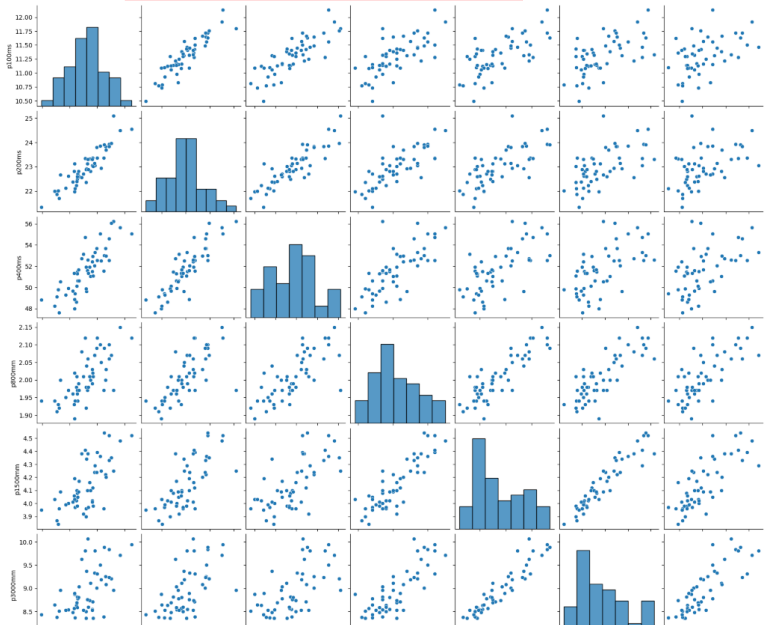
- Vemos que todas as correlações são positivas (conforme uma variável aumenta, outra umenta também.
- Porém conforme mais similares as provas são (100 e 200, 200 e 400), maior a correlação

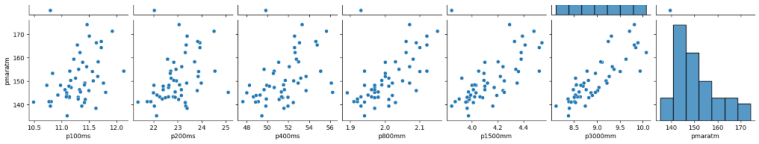
```
In [11]: correlation = dados_semout.corr()
         plt.figure(figsize=(10,10))
         sb.heatmap(correlation, vmax=1, square=True,annot=True,cmap='viridis')

         plt.title('Correlation between different fearures')
```

```
Out[11]: Text(0.5, 1.0, 'Correlation between different fearures')
```




```
In [13]: # Análise exploratória dos dados
         #sb.pairplot()
         sb.pairplot(dados_semout)
```





ANÁLISE DE AGRUPAMENTOS
CLUSTER ANALYSIS

Padronização das variáveis :

Os métodos baseados em distância são afetados pela diferença de escala entre os valores das variáveis/atributos, sendo necessário normalizar os atributos

Padronização - Transforma os valores em números de desvios padrões a partir da média. É dada por: :

$$z = \frac{X - \bar{X}}{S}$$

Onde : $\bar{X}$ = Média da variável
S = desvio padrão

Padronização das variáveis :

Os métodos baseados em distância são afetados pela diferença de escala entre os valores das variáveis/atributos, sendo necessário normalizar os atributos.

Base Original

id	Salário	Idade
1	16.284	47
2	3.500	22
3	13.751	24
4	4.751	24
5	6.751	25
6	8.750	26

	Salário	Idade
Média	8.539,5	31,1
Desvio	4.716,4	9,3

Base com Padronização da Variáveis

id	Salário	Idade
1	1,64	1,71
2	-1,07	-0,97
3	1,10	-0,76
4	-0,80	-0,76
5	-0,38	-0,65
6	0,04	-0,54

In [14]:
```
# feature Scaling
cols = dados_semout.columns

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler().fit(dados_semout)

dados_scaled = scaler.transform(dados_semout)

dados_scaled = pd.DataFrame(dados_scaled, columns=[cols])

dados_scaled.describe().round(2)
```

Out[14]:

	p100ms	p200ms	p400ms	p800mm	p1500mm	p3000mm	pmaratm
count	51.00	51.00	51.00	51.00	51.00	51.00	51.00
mean	-0.00	0.00	0.00	-0.00	0.00	0.00	-0.00
std	1.01	1.01	1.01	1.01	1.01	1.01	1.01
min	-2.52	-2.18	-1.91	-1.86	-1.65	-1.69	-1.64
25%	-0.60	-0.62	-0.83	-0.68	-0.81	-0.82	-0.75
50%	0.02	-0.09	-0.04	-0.12	-0.24	-0.26	-0.23
75%	0.59	0.50	0.61	0.82	0.68	0.63	0.48
max	2.56	2.79	2.16	2.25	2.15	2.28	2.58

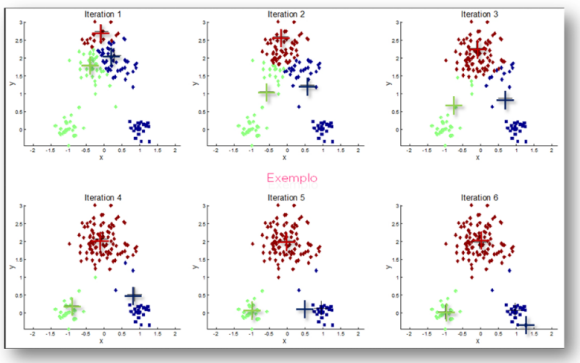
Melhor valor é 1,64 desvios padrão abaixo da média

Pior valor é 2.58 desvios padrão acima da média

Nesse caso específico porque quanto menor o tempo melhor (récorde)

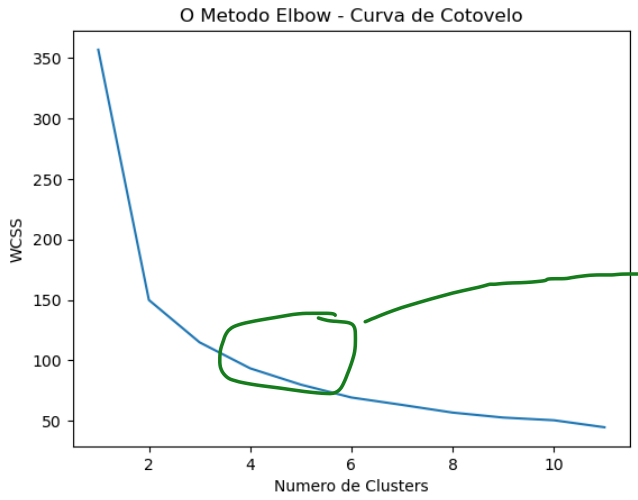
Método KMEANS

MÉTODOS NÃO HIERÁRQUICOS - KMEANS




```
# https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

from sklearn.cluster import KMeans

#opção
```

```
wcss = []

for i in range(1, 12):
    kmeans = KMeans(n_clusters = i, max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(dados_scaled)
    wcss.append(kmeans.inertia_)

# Mostra o Gráfico
plt.plot(range(1, 12), wcss)
plt.title('O Metodo Elbow - Curva de Cotovelo')
plt.xlabel('Numero de Clusters')
plt.ylabel('WCSS') #within cluster sum of squares
plt.show()

# No código acima plotamos o somatório da variância dos dados em relação ao número de clusters
# para conseguir verificar até que ponto com o aumento do número de clusters não existe ganho.

# É sempre bom lembrar que a escolha do parâmetro K é de extrema importância para a tarefa de agrupamento
# e deve ser corretamente alinhado com as regras do negócio ou problema que esteja resolvendo
```



Momento que devemos parar de aumentar o número dos clusteres

```
In [19]: Segmentos = Modelo_Kmeans.fit(dados_scaled)
         Segmentos

C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1436:
leak on Windows with MKL, when there are less chunks than available threads.
ariable OMP_NUM_THREADS=1.
  warnings.warn(

Out[19]: KMeans
         KMeans(n_clusters=5, n_init=10, random_state=0)

In [20]: Segmentos.cluster_centers_

Out[20]: array([[ 1.04731014,  1.18521569,  1.25440918,  0.57640168,  0.66555822,
                  0.46978895,  0.04828959],
                [-0.18514591, -0.12878675, -0.17253788, -0.32741033, -0.4974204 ,
                 -0.50547484, -0.4057217 ],
                [ 1.69302276,  1.62557642,  1.14344811,  1.69486903,  2.01073683,
                  1.82469529,  1.48717377],
                [-1.16644682, -1.29535867, -1.27266844, -1.1593282 , -0.99644395,
                 -0.92320818, -0.84349352],
                [ 0.32696464,  0.26933145,  0.53349896,  1.1017424 ,  1.08810712,
                  1.27286973,  1.43898277]])
```

Centróides de cada grupos

Tem que ter cuidado para não ter muitas variáveis colineares

(ticket médio/volume médio/valor total)

Técnica não supervisionada DBSCAN

Vai procurar regiões de densidade

DBSCAN

Dados de CRM – Vai com K-Means

parametros

class sklearn.cluster.DBSCAN(eps=0.5, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None

Tamanho do Raio

o numero minimo de pontos

```
In [12]: modelo = DBSCAN(eps=0.2, min_samples=5, metric='euclidean').\
             fit(dbscan_df)
```

```
modelo
```

Out[12]:

```
DBSCAN
DBSCAN(eps=0.2)
```

In [13]: # como ficou o clustering