Agrupamento
Validação

Aprendizado não supervisionado

Heloisa de Arruda Camargo

INFORMAÇÃO,

TECNOLOGIA

& INOVAÇÃO

Validação de agrupamento

- A maioria dos algoritmos de agrupamento impõem uma estrutura de agrupamento ao conjunto de dados X .
- Entretanto, X pode não possuir uma estrutura de agrupamento.
  - Assim torna-se necessário fazer a avaliação dos resultados obtidos pelo agrupamento.
  - Validação de Clusters: tarefa que avalia **quantitativamente** os resultados de um algoritmo de agrupamento para verificar se os clusters são significativos

Ressalvas....

- “However, it must be emphasized that the results obtained by these methods are ***only*** tools **at the disposal of the expert** in order to evaluate the resulting clustering.” (Theodoridis & Koutroumbas, 2009).

- “These index can be useful, but we should keep in mind their limited role and treat the findings implied by them as only useful guidelines.” (Pedrycz, 2005).

Abordagens para validação de agrupamento

- Critérios externos
  - Exigem validação estatística para verificar se o agrupamento obtido não é aleatório
  - Usa uma medida externa que mede o grau de correspondência entre um agrupamento C produzido por um algoritmo com uma partição P construída independentemente de C

- Critérios internos
  - Exigem validação estatística para verificar se o agrupamento obtido não é aleatório
  - Usa uma medida interna que avalia o agrupamento C produzido por um algoritmo com base nos dados e na matriz de proximidade

Abordagens para validação de agrupamento

- Critérios relativos:
  - O agrupamento é avaliado por comparação com outras estruturas de agrupamento, resultantes da aplicação:
    - do mesmo algoritmo de agrupamento com diferentes parâmetros ou
    - de outros algoritmos de agrupamento

Índices de validação

- Um índice de validação é uma estatística pela qual a validade de um agrupamento é testada

- Índices podem ser:
- Internos – avaliam o agrupamento com base apenas na matriz de dados ou na matriz de similaridade

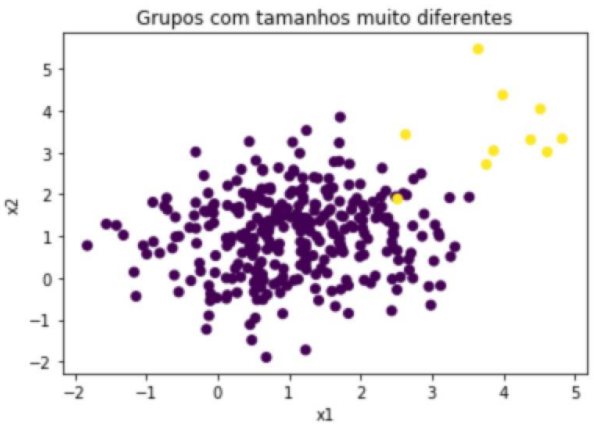
- Externos – avaliam o agrupamento comparando a partição resultante de um algoritmo com uma partição já conhecida

- OBS- A validação relativa pode utilizar os dois tipos de índices

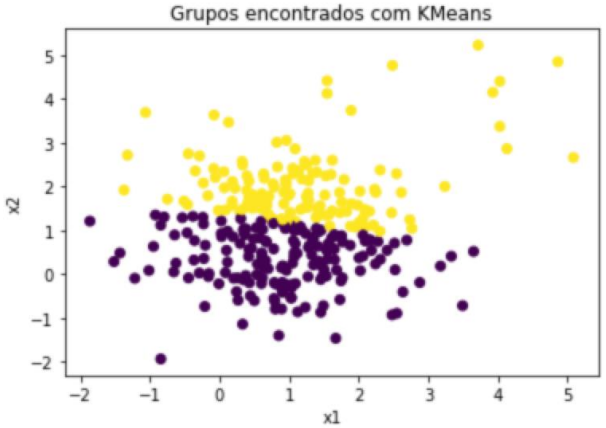
Índices de validação externos

- Agrupamento obtido C = {C_1, C_2, ..., C_m}
- Agrupamento conhecido P = {P_1, P_2, ..., P_s},
  - O número de grupos em C não precisa ser igual ao número de grupos em P

Agrupamento conhecido P



Agrupamento obtido C



Índices de validação externos

- Considere um par de objetos (x_i, x_j)

- Esse par é identificado por:

  - SS se os dois objetos pertencem ao mesmo grupo em C e ao mesmo grupo em P
  - SD se os dois objetos pertencem ao mesmo grupo em C e a grupos diferentes em P
  - DS se os dois objetos pertencem a grupos diferentes em C e ao mesmo grupo em P
  - DD se os dois objetos pertencem a grupos diferentes em C e a grupos diferentes em P

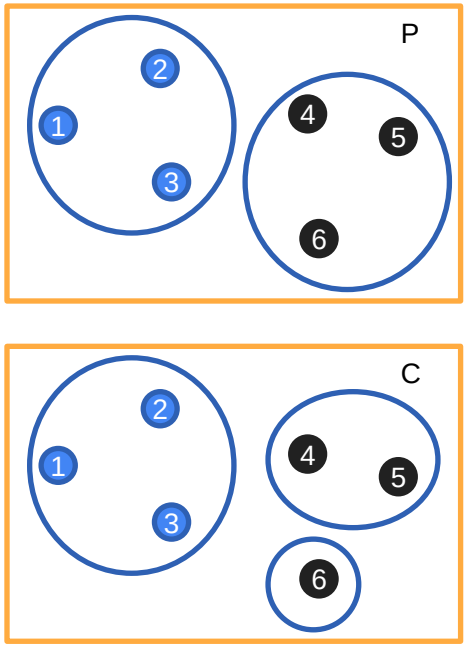
Índices de validação externos

$X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$

$C = \{\{x_1, x_2, x_3\}, \{x_4, x_5\}, \{x_6\}\}$

$P = \{\{x_1, x_2, x_3\}, \{x_4, x_5, x_6\}\}$

	x_1	x_2	x_3	x_4	x_5	x_6
x_1		SS	SS	DD	DD	DD
x_2			SS	DD	DD	DD
x_3				DD	DD	DD
x_4					SS	DS
x_5						DS
x_6						



Índices de validação externos

- Sejam:
- a: o número de pares de vetores de X do tipo SS
- b: o número de pares de vetores de X do tipo SD
- c: o número de pares de vetores de X do tipo DS
- d: o número de pares de vetores de X do tipo DD

- Definimos:

- M - número total de possíveis pares de vetores em X

$$a+b+c+d = M$$

$$M = N(N-1)/2$$

- $m_1 = a+b$ número de pares de objetos que pertencem ao mesmo cluster em C
- $m_2 = a+c$ número de pares de objetos que pertencem ao mesmo cluster em P

Índices de validação externos

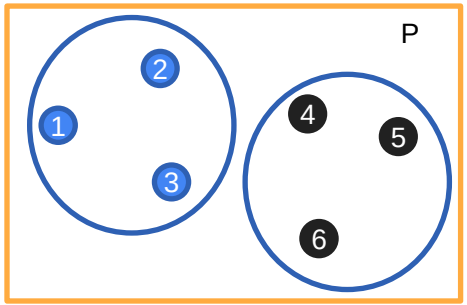
$X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$

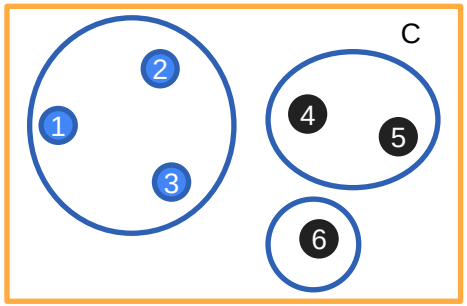
$C = \{\{x_1, x_2, x_3\}, \{x_4, x_5\}, \{x_6\}\}$

$P = \{\{x_1, x_2, x_3\}, \{x_4, x_5, x_6\}\}$

	x_1	x_2	x_3	x_4	x_5	x_6
x_1		*SS*	*SS*	*DD*	*DD*	*DD*
x_2			*SS*	*DD*	*DD*	*DD*
x_3				*DD*	*DD*	*DD*
x_4					*SS*	*DS*
x_5						*DS*
x_6						

a=4
b=0
c=2
d=9





Índice Rand

Mede a similaridade entre dois agrupamentos

$$R = \frac{a+d}{M}$$

- a: o número de pares de vetores de X do tipo SS
  - (SS - os dois objetos pertencem ao mesmo grupo em C e em P)
- d: o número de pares de vetores de X do tipo DD
  - (DD - os dois objetos pertencem a grupos diferentes em C e em P)

- M = a+b+c+d

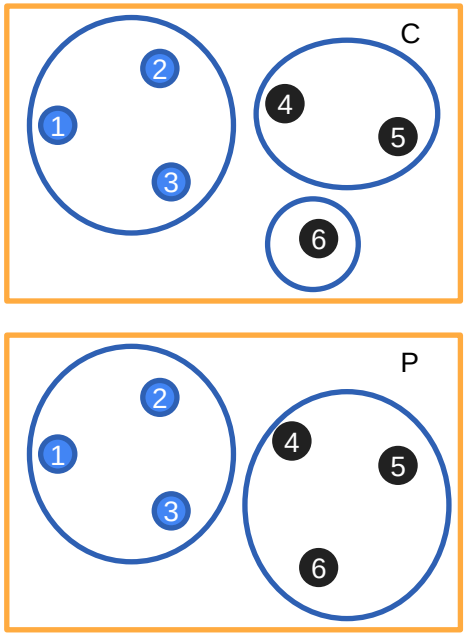
- Mede a fração do número total de pares SS ou DD
- Valores entre 0 e 1
- Para atingir o valor máximo, é necessário ter m=s.

Índice Rand

X = {x_1, x_2, x_3, x_4, x_5, x_6}
C = {{x_1, x_2, x_3}, {x_4, x_5}, {x_6}}
P = {{x_1, x_2, x_3}, {x_4, x_5, x_6}}

	x_1	x_2	x_3	x_4	x_5	x_6
x_1		*SS*	*SS*	*DD*	*DD*	*DD*
x_2			*SS*	*DD*	*DD*	*DD*
x_3				*DD*	*DD*	*DD*
x_4					*SS*	*DS*
x_5						*DS*
x_6						

a=4
b=0
c=2
d=9



R = (a+d)/M = (4+9)/15 = 13/15 = 0,87

Índice Rand

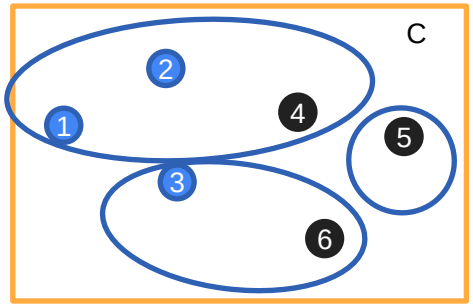
$X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$

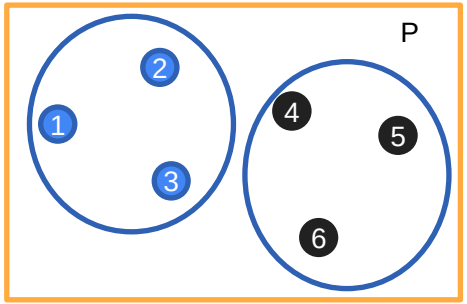
$C = \{\{x_1, x_2, x_4\}, \{x_5\}, \{x_3, x_6\}\}$

$P = \{\{x_1, x_2, x_3\}, \{x_4, x_5, x_6\}\}$

	x_1	x_2	x_3	x_4	x_5	x_6
x_1		*SS*	*DS*	*SD*	*DD*	*DD*
x_2			*DS*	*SD*	*DD*	*DD*
x_3				*DD*	*DD*	*SD*
x_4					*DS*	*DS*
x_5						*DS*
x_6						

a=1
b=3
c=5
d=6





$R = (a+d)/M = (1+6)/15 = 7/15 = 0{,}47$

Índice Rand corrigido

Ajustado para garantir um valor próximo de zero para agrupamentos aleatórios independente do número de clusters e instâncias e valor 1 para agrupamentos idênticos .

$$RC = \frac{R - E[R]}{\max(R) - E[R]}$$

- Vantagens:

- Agrupamentos aleatórios (grupos não válidos) tem um valor perto de zero independente do número de clusters ou de instâncias
- Valores entre -1 e 1
- Não faz suposições sobre a estrutura dos clusters.
  - Pode ser usado para comparar resultados do K-Means, que encontra clusters globulares com resultados do algoritmo de agrupamento spectral, que pode encontrar clusters de outros formatos

Índice Rand corrigido em Python

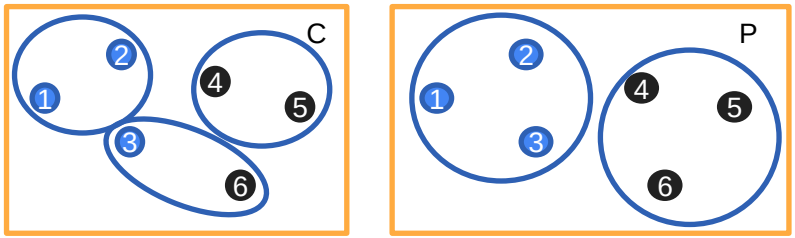
- sklearn.metrics.**adjusted_rand_score**(*labels_true*, *labels_pred*)
- Parâmetros:
  - labels_true : int array, formato = [n_samples]
  - Rótulos dos grupos conhecidos usados como referência
  - labels_pred : array, formato = [n_samples]
  - Rótulos dos clusters obtidos no agrupamento
- Retorna:
  - ari : float
  - Índice de similaridade entre -1.0 and 1.0.

Índice Rand corrigido em Python

```
#Calcular o RC para o conjunto definido
```

```
from sklearn.cluster import metrics
labels_true = [0, 0, 0, 1, 1, 1]
labels_pred = [0, 0, 1, 1, 2, 2]
metrics.adjusted_rand_score(labels_true, labels_pred)
```

```
0.24242424242424246
```



Índice Rand corrigido em Python

```
#Se permutar 0 e 1 e trocar 2 por 3 o resultado é o mesmo
```

```
labels_pred = [1, 1, 0, 0, 3, 3]
metrics.adjusted_rand_score(labels_true, labels_pred)
```

```
0.24242424242424246
```

```
#O cálculo do índice RC é simétrico
```

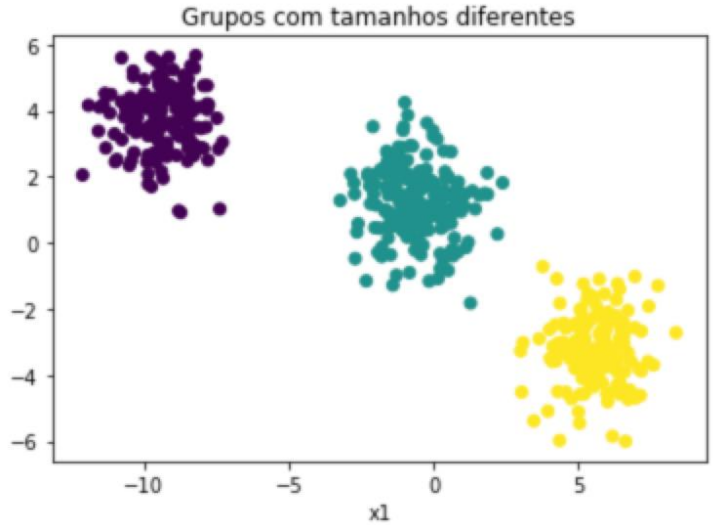
```
metrics.adjusted_rand_score(labels_pred, labels_true)
```

```
0.24242424242424246
```

Índice Rand corrigido em Python

#Calcular o RC para o conjunto definido- gerado com make_blobs

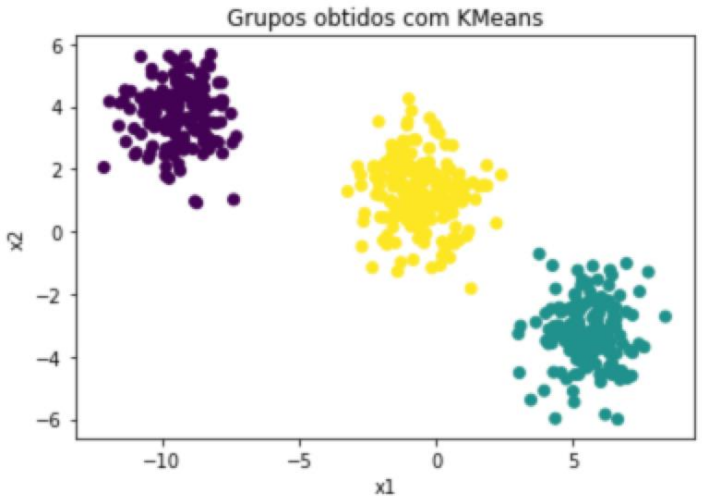
```
#gerando grupos com tamanhos diferentes
X, y = make_blobs(n_samples=[150,200,150])
plt.scatter(X[:,0], X[:,1], c=y)
plt.title("Grupos com tamanhos diferentes")
plt.xlabel("x1")
plt.ylabel("x2")
```



Índice Rand corrigido em Python

```
#Agrupar com Kmeans
```

```
km = KMeans(n_clusters = 3)
km.fit(X)
plt.scatter(X[:,0], X[:,1], c=km.labels_)
plt.title("Grupos obtidos com KMeans")
plt.xlabel("x1")
plt.ylabel("x2")
```



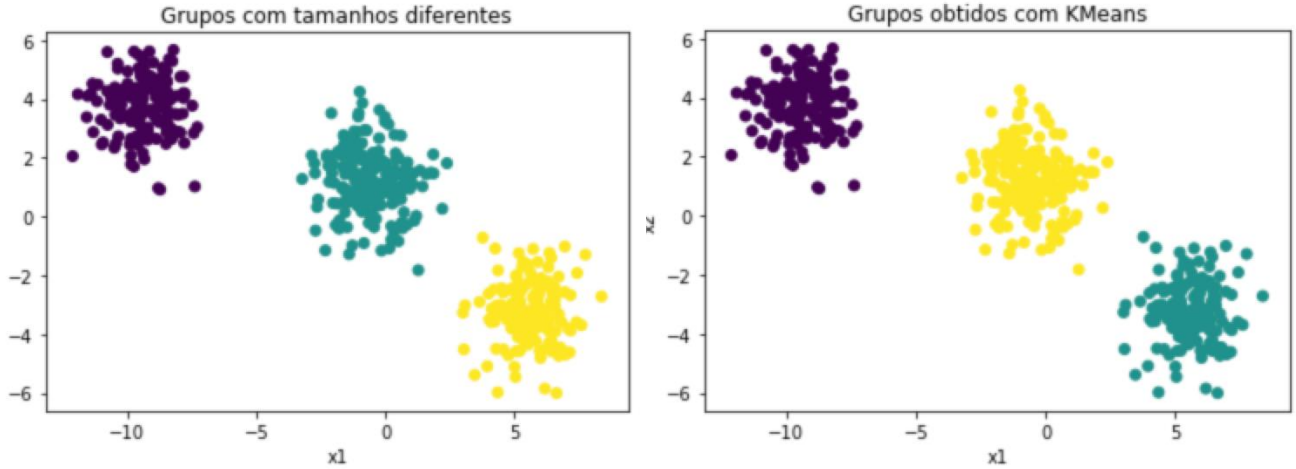
Índice Rand corrigido em Python

```
#Calcular RC
```

```
metrics.adjusted_rand_score(km.labels_,y)
```

```
1.0
```

Neste exemplo, os agrupamentos são iguais



Coeficiente de Jaccard

- Calcula a probabilidade de que dois objetos pertencentes ao mesmo cluster em uma das partições também pertençam ao mesmo cluster na outra partição

$$J = \frac{a}{a+b+c}$$

  - Agrupamentos aleatórios (grupos não válidos) tem um valor perto de zero independente do número de clusters ou de instâncias
  - Valores entre -1 e 1
  - Não faz suposições sobre a estrutura dos clusters.
    - Pode ser usado para comparar resultados do K-Means, que encontra clusters globulares com resultados do algoritmo de agrupamento spectral, que pode encontrar clusters de outros formatos

Índices baseados em informação mútua

- Calcula a concordância entre duas partições
  - MI (Mutual Information)
  - NMI (Normalized Mutual Information)
  - AMI (Adjusted Mutual Information)
- **Vantagens:**

- Para AMI, agrupamentos aleatórios (grupos não válidos) tem um valor perto de zero independente do número de clusters ou de instâncias(o que não acontece para MI ou medida V)
  - **Limite superior de 1**: Valores próximos de zero indicam agrupamentos independentes, valores próximos de 1 indicam concordância significativa entre os agrupamentos

Índice AMI em Python

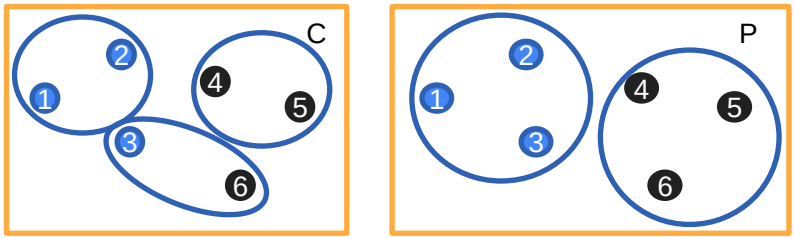
- sklearn.metrics.adjusted_mutual_info_score(*labels_true*, *labels_pred*, *average_method='arithmetic'*)
- Parâmetros:
  - labels_true : int array, formato = [n_samples]
  - Rótulos dos grupos conhecidos usados como referência
  - labels_pred : array, formato = [n_samples]
  - Rótulos dos clusters obtidos no agrupamento
  - average_method: string (optional) (default: ‘arithmetic’)
  - Como calcular o normalizador no denominador
  - Opções: ‘min’, ‘geometric’, ‘arithmetic’, ‘max’.
- Retorna:
  - ami : float
  - Índice de similaridade entre -1.0 and 1.0.

Índice AMI em Python

```
#Calcular o AMI para o conjunto definido
```

```
from sklearn.cluster import metrics
labels_true = [0, 0, 0, 1, 1, 1]
labels_pred = [0, 0, 1, 1, 2, 2]
metrics.adjusted_mutual_info_score(labels_pred, labels_true ,average_method='arithmetic')
```

```
0.29879245817089006
```



Índice AMI em Python

```
#Agrupamentos muito diferentes – índice zero
```

```
metrics.adjusted_mutual_info_score([0, 0, 0, 0], [0, 1, 2, 3], average_method ='arithmetic')
```

```
0.0
```

```
#Agrupamentos idênticos – índice 1
```

```
metrics.adjusted_mutual_info_score([0, 0, 1, 1], [0, 0, 1, 1],average_method='arithmetic'))
```

```
1.0
```

Homogeneidade, completeza e medida V

- **Homogeneidade**: um agrupamento satisfaz homogeneidade se todos os seus clusters contêm apenas dados de uma mesma classe

- **Completeza (completeness)**: um agrupamento satisfaz completeza se todos os membros de uma dada classe são atribuídos ao mesmo cluster

- **Medida-V**: média harmônica de homogeneidade e completeza
  - v = (1 + beta) * homogeneity * completeness / (beta * homogeneity + completeness)

- **Vantagens:**

  - Possui valores limitados: 0 indica agrupamentos ruins, 1 indica agrupamento perfeito

  - Não faz suposições sobre a estrutura dos clusters – pode ser usado para comparar clusters de diferentes formatos

Homogeneidade, completeza e medida V em Python

- sklearn.metrics.homogeneity_score(*labels_true*, *labels_pred*)
- Retorna homogeneity

- sklearn.metrics.completeness_score(*labels_true*, *labels_pred*)
- Retorna completeness

- sklearn.metrics.v_measure_score(*labels_true*, *labels_pred*, *beta=1.0*)
- Retorna v_measure

Homogeneidade, completeza e medida V em Python

```
#Calcular homogeneidade, completeza e medida V para o conjunto definido
```

```
from sklearn import metrics
labels_true = [0, 0, 0, 1, 1, 1]
labels_pred = [0, 0, 1, 1, 2, 2]
print("Homogeneidade: %0.3f " % metrics.homogeneity_score(labels_true, labels_pred))
print("Completeza: %0.3f " %  metrics.completeness_score(labels_true, labels_pred))
print("Medida-V %0.3f " % metrics.v_measure_score(labels_true, labels_pred))
```

```
Homogeneidade: 0.667
Completeza: 0.421
Medida-V 0.516
```

Homogeneidade, completeza e medida V em Python

As medidas de homogeneidade, completeza e medida V podem ser calculadas de uma só vez:

sklearn.metrics.homogeneity_completeness_v_measure(*labels_true*, *labels_pred*, *beta=1.0*)

```
from sklearn import metrics
labels_true = [0, 0, 0, 1, 1, 1]
labels_pred = [0, 0, 1, 1, 2, 2]
metrics.homogeneity_completeness_v_measure(labels_true, labels_pred)
```

```
(0.6666666666666669, 0.420619835714305, 0.5158037429793889)
```

m1 = a+b número de pares de objetos que pertencem ao mesmo cluster em C
m2 = a+c número de pares de objetos que pertencem ao mesmo cluster em P

Índice de Fowlkes e Mallows

- Avalia a similaridade entre duas partições

$$FM(C,P) = \frac{a}{\sqrt{(m_1)(m_2)}}$$

- **Vantagens**:

- Agrupamentos aleatórios (grupos não válidos) tem um valor perto de zero independentemente do número de clusters ou de instâncias(o que não acontece para MI ou medida V)

  - **Limite superior de 1**: Valores próximos de zero indicam agrupamentos independentes, valores próximos de 1 indicam concordância significativa entre os agrupamentos

  - Não faz suposições sobre a estrutura dos clusters – pode ser usado para comparar clusters de diferentes formatos

Índice Fowlkes e Mallows em Python

- sklearn.metrics.fowlkes_mallows_score(*labels_true*, *labels_pred*, *sparse=False*)
- Parâmetros:
  - labels_true : int array, formato = [n_samples]
  - Rótulos dos grupos conhecidos usados como referência
  - labels_pred : array, formato = [n_samples]
  - Rótulos dos clusters obtidos no agrupamento
  - sparse: booleano
  - Calcula matriz de contingência internamente
- Retorna:
  - score : float
  - Índice de similaridade entre -1.0 and 1.0.

Índice FM em Python

```
#Calcular índice FM para o conjunto definido
```

```
from sklearn.cluster import metrics
labels_true = [0, 0, 0, 1, 1, 1]
labels_pred = [0, 0, 1, 1, 2, 2]
print(sm.fowlkes_mallows_score(labels_true, labels_pred))
print(sm.fowlkes_mallows_score([0, 0, 1, 1], [0, 0, 1, 1]))
print(sm.fowlkes_mallows_score([0, 0, 1, 1], [1, 1, 0, 0]))
```

```
0.4714045207910317
1.0
1.0
```

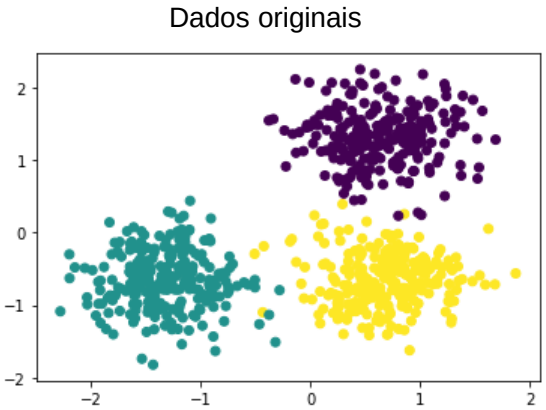
Índices externos de validação de agrupamento

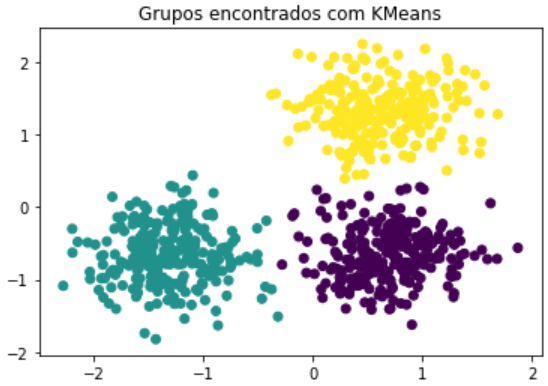
Exercícios em Python no Colab

Índices externos em Python

```
#Agrupar com KMeans
```

```
km = KMeans(n_clusters = 3, init='random').fit(X)
labels_true = y
rotulos_km = km.labels_
plt.scatter(X[:,0], X[:,1], c=rotulos_km)
plt.title("Grupos encontrados com KMeans")
```





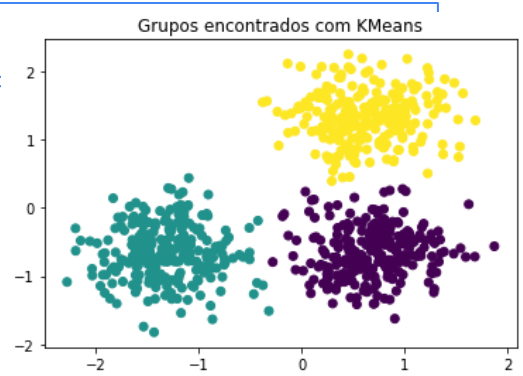
Índices externos em Python

```
#Calcular índices de validação externos do agrupamento com KMeans
```

```
h = metrics.homogeneity_score(labels_true, rotulos_km)
print("Homogeneidade: %0.3f" % h)
c = metrics.completeness_score(labels_true, rotulos_km)
print("Completeza: %0.3f" % c)
v = metrics.v_measure_score(labels_true, rotulos_km)
print("Medida V: %0.3f" % v )
ari = metrics.adjusted_rand_score(labels_true, rotulos_km)
print("Índice Rand corrigido: %0.3f" % ari)
ami = metrics.adjusted_mutual_info_score(labels_true, rotulos_km,
      average_method='arithmetic')
print("Adjusted Mutual Information: %0.3f"  % ami )
fm = metrics.fowlkes_mallows_score(labels_true, rotulos_km)
print("Índice Fowlkes-Mallows: %0.3f"  % fm )
```

Índices externos em Python

#Calcular índices de validação externos do agrupamento com KMeans




```
h = metrics.homogeneity_score(labels_true, rotulos_km)
print("Homogeneidade: %0.3f" % h)
c = metrics.completeness_score(labels_true, rotulos_km)
print("Completeza: %0.3f" % c)
v = metrics.v_measure_score(labels_true, rotulos_km)
print("Medida V: %0.3f" % v )
ari = metrics.adjusted_rand_score(labels_true, rotulos_km)
print("Índice Rand corrigido: %0.3f" % ari)
ami = metrics.adjusted_mutual_info_score(labels_true, rotulos_km, average_method='arithmetic')
print("Adjusted Mutual Information: %0.3f"  % ami )
fm = metrics.fowlkes_mallows_score(labels_true, rotulos_km)
print("Índice Fowlkes-Mallows: %0.3f"  % fm )
```

Homogeneidade: 0.945 Completeza: 0.945
Medida V: 0.945 Índice Rand corrigido: 0.968 Adjusted Mutual Information: 0.945 Índice Fowlkes-Mallows: 0.979

Índices de validação internos

- Índices que avaliam a qualidade do agrupamento com base apenas nas estruturas internas como matriz de dados ou matriz de similaridade
- Nenhuma informação externa sobre os grupos é conhecida

Matriz de dados

$$\begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 5 & 4 \\ 6 & 5 \\ 6.5 & 6 \end{bmatrix}$$

Matriz de dissimilaridade distância Euclidiana

$$\begin{bmatrix} 0 & 1 & 5 & 6.4 & 7.4 \\ 1 & 0 & 4.2 & 5.7 & 6.7 \\ 5 & 4.2 & 0 & 1.4 & 2.5 \\ 6.4 & 5.7 & 1.4 & 0 & 1.1 \\ 7.4 & 6.7 & 2.5 & 1.1 & 0 \end{bmatrix}$$

Coeficiente de silhueta

- A medida se baseia na proximidade entre os objetos de um cluster e na distância dos objetos de um cluster ao cluster mais próximo

- Avalia:
  - A adequação de cada objeto ao seu cluster
  - A qualidade de um cluster individualmente
  - A qualidade de uma partição

- Valores entre [-1, 1]

- Melhor partição tem valor 1

Coeficiente de silhueta

- **a**: Distância média entre um objeto e todos os outros do mesmo cluster
- **b**: Distância média entre um objeto e todos os outros do cluster mais próximo
- Silhueta de um objeto x_i:

$$sil(x_i) = \frac{b-a}{\max(a,b)}$$

- Silhueta de um cluster:

$$sil(C_j) = \frac{1}{|C_j|} \sum_{x_i \in C_j} sil(x_i)$$

- Silhueta de um agrupamento:

$$sil(C) = \frac{1}{n} \sum_{i=1}^{n} sil(x_i)$$

Coeficiente de silhueta

- **Vantagens**:
- Limitado entre -1 para agrupamentos incorretos e +1 para agrupamentos densos
- Índices próximos de zero indicam clusters sobrepostos
- O índice é mais alto quando os clusters são densos e separados

Interpretação:

S <= 0,25	Não foi encontrada uma estrutura
0,26 <= S <= 0,5	Estrutura fraca
0,51 <= S <= 0,7	Estrutura razoável
0,71 >= S <= 1	Estrutura forte

- **Desvantagens:**
  - Custo computacional elevado
  - É melhor para clusters convexos (obtidos por Kmeans) do que para clusters com outros formatos (densos, DBSCAN)

Coeficiente de silhueta em Python

- sklearn.metrics.silhouette_score(*X*, *labels*, *metric='euclidean'*, *sample_size=None*, *random_state=None*, ***kwds*)

- Parâmetros:

  - X : int array, formato = [n_samples, n_features]
  - Matriz de dados

  - labels: array, formato = [n_samples]
  - Rótulos dos clusters obtidos no agrupamento

  - Metric: string : string
  - Medida de distância utilizada

- Retorna:
  - silhouette : float
  - Coeficiente de silhueta entre -1.0 and 1.0.

Índice Davies Bouldin

- Calcula a similaridade média entre cada cluster e o mais parecido com ele

- **Vantagens**:
- O cálculo é mais simples do que a silhueta

- **Desvantagens:**

- O índice é maior para custers convexos do que para outros conceitos de clusters, tais como clusters baseados em densidade como os obtidos pelo DBSCAN

- O uso de centroides limita a métrica de distância para o espaço Euclidiano

- A obtenção de um valor bom por esse método não implica que o melhor agrupamento foi obtido.

Índice Davies-Bouldin em Python

- sklearn.metrics.davies_bouldin_score(*X*, *labels*)

- Parâmetros:

  - X : int array, formato = [n_samples, n_features]
  - Matriz de dados

  - labels: array, formato = [n_samples]
  - Rótulos dos clusters obtidos no agrupamento

- Retorna:
  - score : float
  - Índice Davis Bouldin

Índice Caliski-Harabasz

- Calcula a razão entre a dispersão média entre pares de clusters e a dispersão intra-clusters
- Valores mais altos indicam clusters mais bem definidos

- **Vantagens**:
    - O índice é alto quando os clusters são densos e bem separados, o que está relacionado a um conceito padrão de cluster
    - É rápido para calcular

Índice Caliski-Harabasz em Python

- sklearn.metrics.calinski_harabasz_score(*X*, *labels)*
- Parâmetros:
  - X : int array, formato = [n_samples, n_features]
  - Matriz de dados
  - labels: array, formato = [n_samples]
  - Rótulos dos clusters obtidos no agrupamento
- Retorna:
  - score : float
  - Índice Calisnke-Harabasz

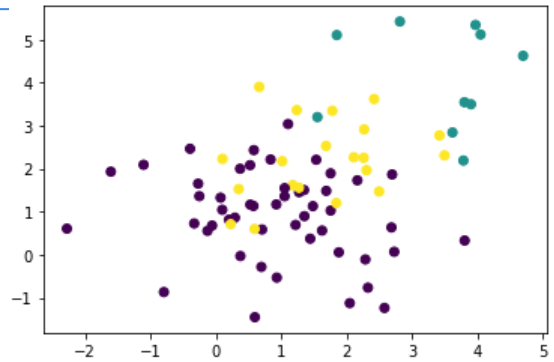
Índices internos de validação de agrupamento

Exercícios em Python no Colab

Índices de validação internos em Python

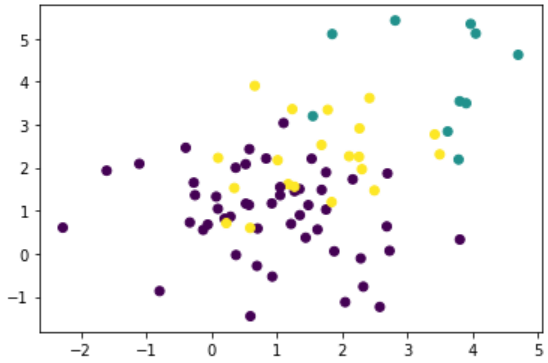
#Gerar conjunto de dados, agrupar e calcular os índices internos

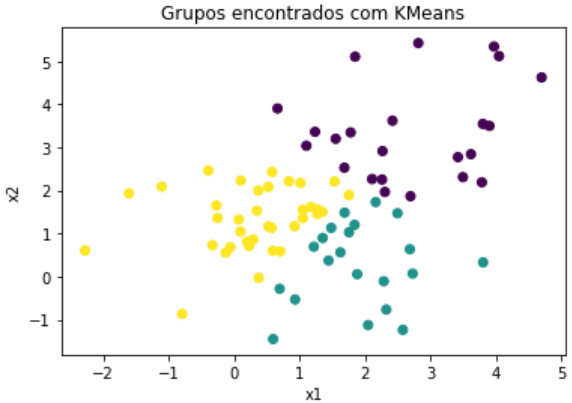
```
#gerando dados 2D
n_grupos = 3
X, y = make_blobs(n_samples=[50,10,20], centers = ([[1,1],[4,4],[2,2]]))
plt.scatter(X[:,0], X[:,1], c=y)
```



Índices de validação internos em Python

```
# Executando KMeans e mostrando o resultado
y_pred = KMeans(n_clusters=3,init='random')
y_pred.fit(X)
#Usando os rótulos dos grupos para plotar os grupos obtidos
rotulos= y_pred.labels_
plt.scatter(X[:,0], X[:,1], c=rotulos)
plt.title("Grupos encontrados com KMeans")
plt.xlabel("x1")
plt.ylabel("x2")
```





Índices de validação internos em Python

```
#Calcular índices internos para o agrupamento resultante
```

```
s= metrics.silhouette_score(X, rotulos, metric='euclidean')
print("Coeficiente de Silhueta: %0.3f" % s)
ch = metrics.calinski_harabasz_score(X, rotulos)
print("Calinski_harabasz: %0.3f" % ch)
dbs = metrics.davies_bouldin_score(X, rotulos)
print("Davies Bouldin: %0.3f" % dbs)
```

```
Coeficiente de Silhueta: 0.332
Calinski_harabasz: 58.099
Davies Bouldin: 0.983
```