PROCESSO DE DESENVOLVIMENTO DOCUMENTAÇÃO DE PROJETO		
Nome do curso: Machine Learning	Clusterização com K-means	Responsável: Paulo Jarbas Camurça

Clusterização com K-means

Olá, seja bem-vindo!

Durante este curso, você aprendeu sobre **“clusterização”**, a técnica de Machine Learning que faz parte do aprendizado não supervisionado, também conhecida como “análise de agrupamentos”, está lembrado? Então, nesta aula, será abordado um dos mais renomados algoritmos não hierárquicos na análise de clusters, o algoritmo **K-means**. Ficou curioso para saber como funciona? Vamos lá!

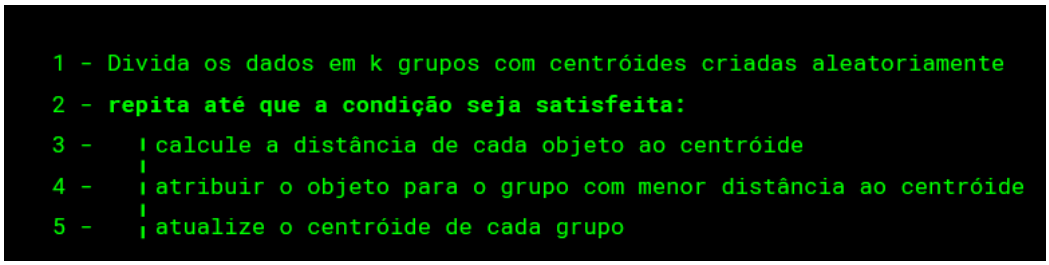
Para agrupar dados não rotulados e que possuem características semelhantes, diversos sistemas utilizam a clusterização. Para realizar essa tarefa, existem vários algoritmos, sendo um dos mais simples e populares algoritmos de clustering, o **K-means**, que você, agora, vai conhecer e entender como funciona.

Inicialmente, vamos falar sobre a definição do K-means. Ele é um algoritmo não hierárquico, que tem o objetivo de agrupar os dados em k clusters diferentes. Para entender como o K-means funciona é preciso, primeiro, definir o conceito de centróide de um cluster.

Um centróide representa o centro de um cluster e é obtido pelo valor médio dos dados que estão dentro desse cluster. Assim, o algoritmo K-means encontra, iterativamente, o centróide de cada cluster através de uma medida de distância, de

forma que cada cluster contenha os dados, cuja distância entre eles e o centróide seja mínima. O algoritmo K-means pode ser representado pelas seguintes etapas:

1 - Divida os dados em k grupos com centróides criadas aleatoriamente
2 - **repita até que a condição seja satisfeita:**
3 - | calcule a distância de cada objeto ao centróide
|
4 - | atribuir o objeto para o grupo com menor distância ao centróide
|
5 - | atualize o centróide de cada grupo



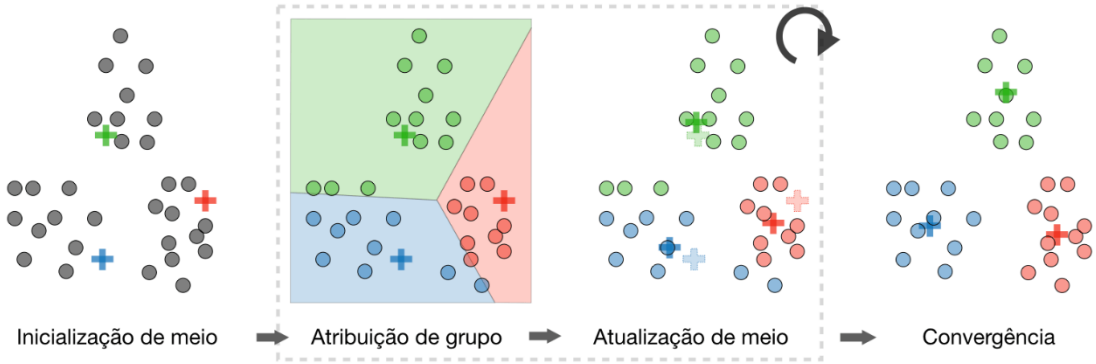
Graficamente, as etapas podem ser expressas da seguinte forma: Os passos do algoritmo K-means estão ligados por setas e cada passo é ilustrado por três grupos de círculos, representando os dados, com um símbolo de + colorido.

No primeiro passo, o da **“Inicialização de meio”**, o grupo à esquerda tem um símbolo de + azul, o da direita tem um símbolo de + em vermelho e o de cima tem um símbolo de + verde. Todos os grupos são da cor cinza.

No segundo passo, o da **“Atribuição de grupo”**, os grupos agora estão coloridos e estão divididos por três linhas. À esquerda, sobre um fundo azul, há o grupo azul com o + azul. À direita, sobre um fundo vermelho, existe o grupo vermelho com o + vermelho. Em cima, sobre um fundo verde, encontra-se o grupo verde com o + verde. Alguns círculos de um grupo cruzam parcialmente a linha do outro grupo.

No terceiro passo, o da **“Atualização de meio”**, as linhas e os fundos sumiram, mas os grupos ainda mantêm a mesma posição do passo 2, com alguns círculos de um grupo próximo de outro. Sobre esse passo, há um símbolo de atualização.

No quarto passo, o da **"Convergência"**, os grupos estão claramente separados entre si, sem nenhum círculo de um grupo se aproximando de outro.



O algoritmo realizará esses passos até que uma condição de parada seja satisfeita. Essa condição de parada é conhecida como convergência, e pode ser representada, por exemplo, pela soma do erro quadrático, do inglês **SSE (Sum of Squared Errors).** Ele representa a soma das diferenças ao quadrado, entre cada observação e a média do grupo em um conjunto de dados.

$$SSE = \sum_{i=1}^{n} (X_i - \bar{X})^2$$

$\bar{X}$ é a média

X_i é a observação

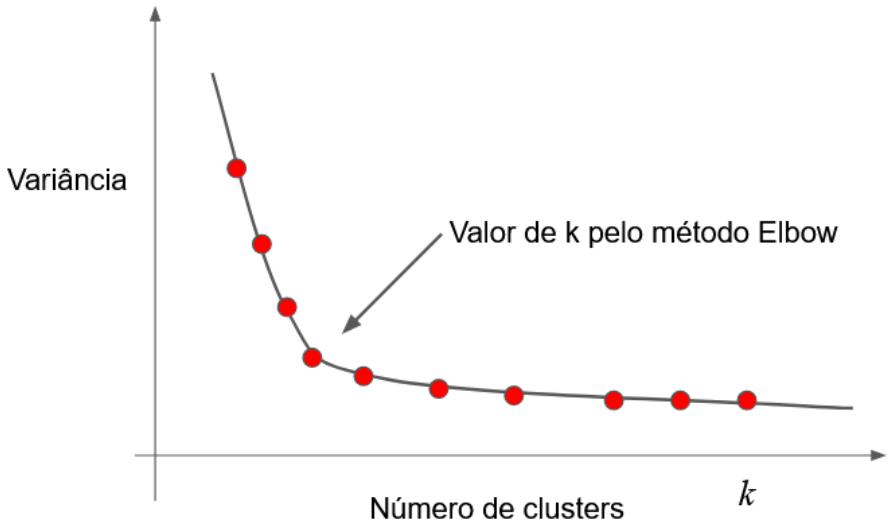
Agora que você já entendeu como o algoritmo funciona, deve haver o seguinte questionamento: como o valor de k deverá ser escolhido? Bem, vamos lá!

Na literatura, um dos métodos mais populares de encontrar o valor de k é o método **Elbow**, que consiste em plotar a variância dos dados em função do número

de clusters. Desse modo, a partir do ponto em que os valores de k crescem e os valores de variância não mudam de forma significativa, esse ponto é o melhor valor para k.

Sendo assim, o método **Elbow**, também conhecido como **método do cotovelo**, cujo apelido se deve, além da tradução literal de **Elbow** ser “cotovelo”, à curva formada por ele também lembra o formato de um cotovelo.

A representação gráfica desse método é formada por dois eixos. O eixo Y, na vertical, representa a variância. O eixo X, na horizontal, representa o número de clusters. Ao lado, há a letra k. No gráfico, existe uma linha com vários pontos vermelhos que se assemelha a um cotovelo. No quarto ponto, na “dobra” do cotovelo, encontra-se uma seta com a mensagem “Valor de k pelo método Elbow”.



Agora, de forma prática, no ambiente do jupyter, você vai aplicar o algoritmo K-means, usando a biblioteca scikit-learn em um conjunto de dados, tudo bem?

Para isso, utilize uma base de dados criada para fins didáticos, que representa informações básicas dos clientes de um shopping. Depois, use o algoritmo K-means para identificar possíveis grupos de clientes.

Inicialmente, com o ambiente do jupyter aberto, localize e importe as bibliotecas numpy, pandas e matplotlib. Para isso, escreva, na primeira linha, o comando “import numpy as np”; na segunda, “import pandas as pd”; e, na terceira, “import matplotlib.pyplot as plt”.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Em seguida, você deve fazer a leitura dos dados dos clientes usando o método pd.read_csv do pandas. Para isso, escreva o comando “df = pd.read_csv('Mall_Customers.csv')” e, para localizar as primeiras linhas do arquivo, escreva df.head().

Leitura dos dados

```
df = pd.read_csv('Mall_Customers.csv')
df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

O dataframe é constituído por uma tabela com 5 colunas. A primeira delas é a **CustomerID**, que representa uma identificação para o cliente. A segunda coluna, a **Gender**, faz referência ao gênero do cliente. A terceira é a **Age**, que representa a idade. Já a quarta coluna, a **Annual Income**, é a renda anual do cliente em dólares multiplicados por mil. E por fim, a quinta e última coluna do exemplo, a **Spending Score**, que representa um score de 1 a 100 atribuído ao cliente, de acordo com comportamento de compras.

Para saber o número de registros do dataframe, você deve usar o método shape:

```
df.shape
(200, 5)
```

```
df.shape
(200, 5)
```

Então, o dataframe possui 200 linhas e 5 colunas com os registros. Para saber se existem dados nulos, o comando utilizado é o “df.isnull().sum()”, que mostra a soma dos registros que são nulos. Conforme a demonstração, na parte superior estará escrito “Verificar dados nulos” e logo abaixo, o comando “df.isnull ().sum ()” e o resultado:

```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Logo, é possível identificar que não existem dados nulos em nossa base de dados.

Verificar dados nulos

```
df.isnull().sum()
```

```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Você também pode obter algumas informações estatísticas sobre os dados e, para isso, basta utilizar o comando “df.describe()”. Na demonstração, após a utilização do comando, foram exibidas as “Informações estatísticas” de oito clientes em forma de tabela.

A tabela tem cinco colunas, A primeira coluna apresenta os tipos estatísticos, que são: “count”, “mean”, “std”, “min”, “25%”, “50%”, “75%” e “max”. A segunda é a CustomerID, que é a identificação do cliente. Na terceira coluna denominada de Age, temos a representação da idade do cliente, Já a quarta coluna, a Annual Income (k$), traz a renda anual do cliente. E a quinta e última coluna é a Spending Score (1-100), que traz um score ou um número, que representa o comportamento de compras do cliente.

Informações estatísticas

```
df.describe()
```

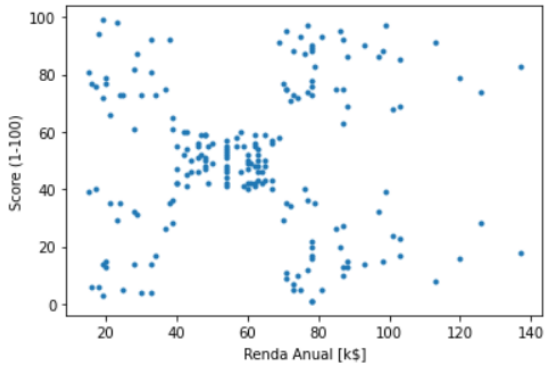
	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

Com essas informações sobre os dados, você pode saber, por exemplo, que o cliente, que possui a maior idade, tem 70 anos; e que o cliente mais novo tem 18 anos. Além disso, é possível saber que 75 % dos clientes possuem renda anual de 78 mil dólares.

Então, agora, gere um gráfico com a renda anual versus o score dos clientes, usando a biblioteca matplotlib. Para isso, escreva “plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'], marker='.')”. Em seguida, escreva os comandos para plotar o título do eixo x e do eixo y, “plt.xlabel('Renda Anual [k$]')” e “plt.ylabel('Score (1-100)')”, finalizando com “plt.show()”.

Será gerado um gráfico de dispersão que representa a renda anual versus o score dos clientes. O gráfico possui dois eixos: O eixo Y, na vertical, é o Score (1-100) dos clientes, e apresenta os valores 0, 20, 40, 60, 80, 100. O eixo X, na horizontal, é a Renda Anual (k$) e apresenta os valores 20, 40, 60, 80, 100, 120 e 140. Como resultado, há vários pontos azuis espalhados, sendo o maior agrupamento de pontos entre 60 e 40 no score e renda anual entre 40 e 70.

```
plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'], marker='.')
plt.xlabel('Renda Anual [k$]')
plt.ylabel('Score (1-100)')
plt.show()
```



Agora, selecione os dados de renda anual e de score que serão usados como entrada para a clusterização, pois você pode agrupar o cliente de acordo com o score, certo? Para isso, escreva, para a variável X, o comando “X = df[['Annual Income (k$)' e 'Spending Score (1-100)']]”, utilizando o comando “X.head()” para obter os primeiros registros.

Neste exemplo, será formada uma tabela com duas colunas. A primeira é intitulada Annual Income (k$) e representa a renda anual do cliente. A segunda é a Spending Score (1-100), que é um score representando o comportamento de compras do cliente.

Na primeira linha, o cliente 0 tem uma annual income de 15 e seu spending score é 39. Na segunda linha, o cliente 1 tem uma annual income de 15 e seu score é 81. Já na terceira, o cliente 2 tem uma annual income de 16 e seu score é 6. Na

quarta linha, o cliente 3 tem uma annual income de 16 e seu score é de 77. Na quinta e última linha, o cliente 4 tem uma annual income de 17 e seu score é 40.

Selecionando dados para agrupamento

```
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]
X.head()
```

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40

Em seguida, escolha o valor da quantidade de grupos do algoritmo K-means. Em primeiro lugar, importe a biblioteca que contém o algoritmo K-means, escrevendo “from sklearn.cluster import KMeans” .

Importando K-means

```
from sklearn.cluster import KMeans
```

O módulo Kmeans recebe, como parâmetros de entrada, as variáveis **n_clusters**, e **init**, que são, respectivamente, o número de clusters k e o tipo de inicialização. Por simplicidade, escolha o valor de **n_clusters** = 5 e **init** = ‘k-means++’.

Dando continuidade, escreva os comandos “modelo_kmeans = KMeans(n_clusters= 5, init='k-means++')” para criar o modelo com 5 grupos. Feito isso, passe os dados contidos na variável X para o método **fit_predict**, para que o modelo possa iterar sobre os dados e retornar os grupos, que são os resultados da clusterização, fazendo “y_kmeans = modelo_kmeans.fit_predict(X)”.

Nesta amostra, foi organizado um array com cinco grupos ou cinco clusters. Nele, são exibidos os números 0, 3, 2, 4 e 2 em várias combinações dentro de parênteses angulares.

Clusterizando com k = 5

```
modelo_kmeans = KMeans(n_clusters= 5, init='k-means++')
```

```
y_kmeans= modelo_kmeans.fit_predict(X)
```

```
print(y_kmeans)
```

```
[0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0
 3 0 3 0 3 0 1 0 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 4 2 4 1 4 2 4 2 4 1 4 2 4 2 4 2 4 1 4 2 4 2 4
 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2
 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4]
```

Através do comando “print(y_kmeans)”, você obterá um array com os grupos criados. Logo, foram criados 5 grupos, não é mesmo?

O primeiro grupo, pode ser exibido ao escrever “print(X[y_kmeans == 0])”, que retornará os dados agrupados na primeira categoria, como neste exemplo de visualização do primeiro grupo criado, formado por duas colunas: Annual Income (k$) e Spending Score (1-100) com 25 linhas.

Visualizando o primeiro grupo criado

```
print(X[y_kmeans == 0])
```

	Annual Income (k$)	Spending Score (1-100)
123	69	91
125	70	77
127	71	95
129	71	75
131	71	75
133	72	71
135	73	88
137	73	73
139	74	72
141	75	93
143	76	87
145	77	97
147	77	74
149	78	90
151	78	88
153	78	76
155	78	89
157	78	78
159	78	73
161	79	83
163	81	93
165	85	75
167	86	95
169	87	63
171	87	75

Para visualizar os grupos criados, é preciso plotar dentro de um loop, passando para a função de plot, cada parte dos dados correspondentes àquele grupo. Para isso, escreva “k_grupos = 5”, e cores = ['r', 'b', 'k', 'y', 'g'], para definir, respectivamente, o número de grupos e as cores de cada grupo.

Dentro do loop, escreva: cluster = X[y_kmeans == k], para obter os dados do grupo; e “plt.scatter(cluster['Annual Income (k$)'], cluster['Spending Score (1-100)'], s = 100, c = cores[i], label = f'Cluster {k}')”, para fazer o plot.

Visualizando os grupos

```
k_grupos = 5
cores = ['r', 'b', 'k', 'y', 'g']
for k in range(k_grupos):
    cluster = X[y_kmeans == k]
    plt.scatter(cluster['Annual Income (k$)'], cluster['Spending Score (1-100)'],
                s = 100, c = cores[k], label = f'Cluster {k}')


plt.title('Grupos de clientes')
plt.xlabel('Renda Anual (k$)')
plt.ylabel('Score (1-100)')
plt.grid()
plt.legend()
plt.show()
```

Para gerar o gráfico, você deve escrever as linhas de código “plt.title('Grupos de clientes')” para o título do gráfico, e “plt.xlabel('Renda Anual (k$)')”, “plt.ylabel('Score (1-100)')” para os títulos do eixo x e y, assim como “plt.legend()” para a legenda; e, por fim, “plt.show()”.

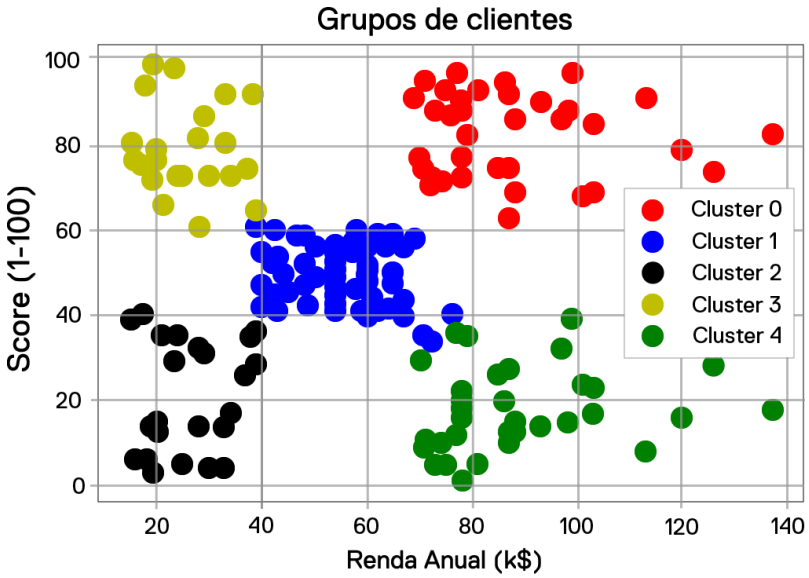
A representação gráfica da clusterização de grupos de clientes será indicada por pontos coloridos e classificada por score e renda anual, conforme o exemplo. À direita, há uma legenda mostrando a cor que representa cada cluster. O gráfico é formado por dois eixos.

No eixo Y, na vertical, temos os números 0, 20, 40, 60, 80 e 100 que representam o score. No eixo X, na horizontal, temos os números 20, 40, 60, 80, 100, 120 e 140 que representam a renda anual. No gráfico, os pontos pretos que

representam o Cluster 2, têm em maior parte um score entre 0 e 40 e uma renda anual entre 20 e 40.

Os pontos verdes, que representam o Cluster 4, têm em maior parte um score entre 0 e 40 e uma renda anual entre 70 e 100. Os pontos azuis, que representam o Cluster 1, estão concentrados entre os valores de 40 e 60 no score e entre 40 e 70 na renda anual.

Os pontos amarelos, que representam o Cluster 3, têm na maior parte um score entre 60 a 100 e uma renda anual entre 20 e 40. Os pontos vermelhos, que representam o Cluster 0, têm na maior parte um score de 70 a 100 com uma renda anual entre 70 e 100.



Desse modo, a partir do resultado da análise de cluster, você perceberá, por exemplo, que o grupo de clientes com score maior que 60 e renda anual a partir de 60 mil dólares por ano corresponde ao grupo de número 0, ou seja, o primeiro grupo.

Até aqui, você estudou o algoritmo K-means e como ele agrupa um conjunto de dados, utilizando medidas de distância e comparando-as com os centróides, assim como também aprendeu como o valor de k pode ser obtido. Vale ressaltar que, para um melhor aprendizado do funcionamento e aplicação das técnicas, é essencial que

você coloque em prática os comandos abordados. E não se esqueça de aprofundar o conteúdo e resolver os exercícios.

Bons estudos e até mais!