ANÁLISE COM ALGORITMOS DE CLUSTERING

Que bom que você chegou até aqui!

Eu sei que nos últimos textos temos aumentado o grau de complexidade e de autonomia que esperamos de vocês.

Temos enfatizado ainda que compartilhe e ajude a criar um senso de comunidade com os seus colegas de curso. Isso acontece, entre outras coisas, porque a cada novo assunto que vamos avançando, os materiais se tornam mais escassos, mais complexos, com menos exemplos de aplicações práticas… Esses elementos podem ser compensados pelo poder do conhecimento coletivo. Pessoas que testaram ou estudaram por horas, dias, semanas, meses e anos, dão um grande ganho à comunidade ao documentar e compartilhar uma implementação de um algoritmo, por exemplo.

Neste texto vamos abordar um assunto que começamos lá no início da nossa jornada. Quando falávamos de algoritmos de aprendizado não supervisionado, nos referíamos principalmente à clusterização e redução de dimensionalidade. O segundo, abordamos no outro texto deste mesmo bloco de estudos. Mas aqui vamos fazer de cluster!

MAS… O QUE É UM CLUSTER?

É um grupo de elementos ou observações com características similares entre si. Sua identificação é possível, graças a combinação de fatores que se repete em cada conjunto de observações similares. Mas, diferente dos algoritmos supervisionados, em que criávamos labels para indicar o padrão que estávamos procurando, os algoritmos não supervisionados são capazes de aprender os padrões sem a necessidade de labels. Como isso acontece? Bem cada algoritmo tem uma forma particular de aprender e identificar os padrões nos dados, mas eles se dividem em 4 grupos ou métodos.

OS QUATRO PRINCIPAIS MÉTODOS DE CLUSTERIZAÇÃO

Assista a esse vídeo: **https://youtu.be/Se28XHI2_xE**

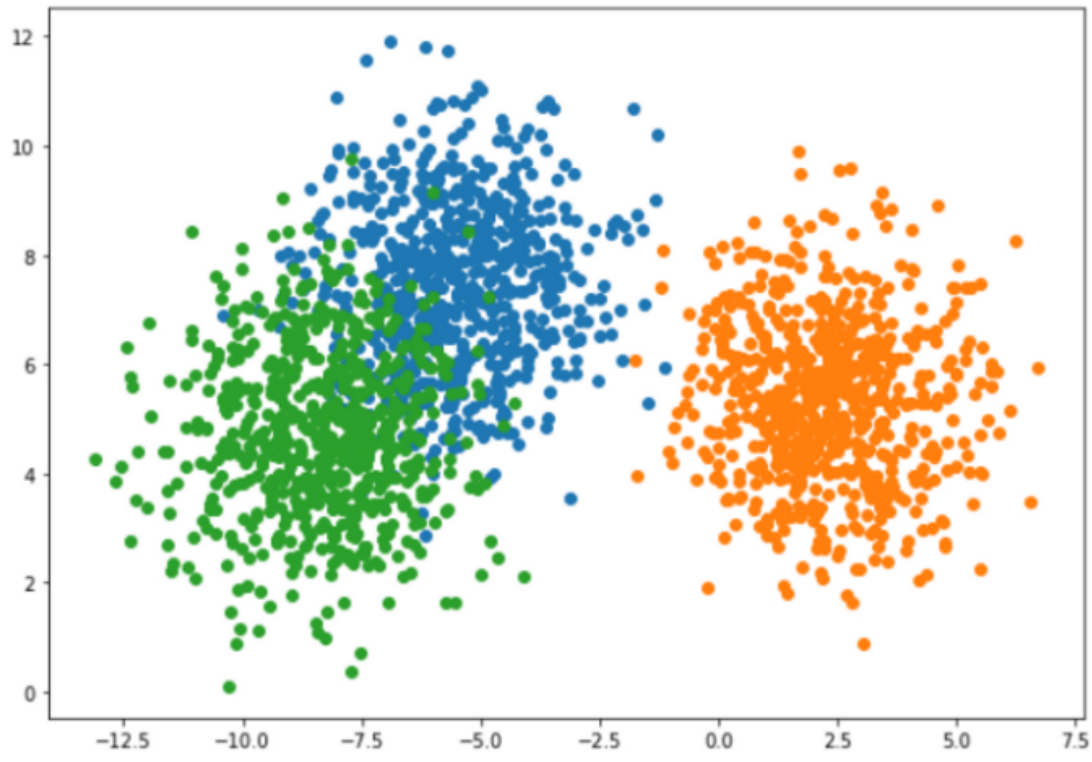
ANÁLISE COM ALGORITMOS DE CLUSTERING

clusterização (Clusterização por centróides, Clusterização por densidade, Clusterização por distribuição e Clusterização por conecitividade).

GERAÇÃO DE DADOS SINTÉTICOS

Para acompanhar um pouco do efeito da clusterização, vamos criar um dataset sintético que vai servir de base para compararmos os diferentes algoritmos em ação.

```
# synthetic classification dataset
from numpy import where
from sklearn.datasets import make_blobs
from matplotlib import pyplot
fig= plt.figure(figsize=(10,7))
# define dataset
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=2000, centers=3,
n_features=5,cluster_std=1.5, random_state=5)
# create scatter plot for samples from each class
for class_value in range(3):
# get row indexes for samples with this class
    row_ix = where(y == class_value)
# create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```



Dados sintéticos gerados no python

ANÁLISE COM ALGORITMOS DE CLUSTERING

Baixe aqui!

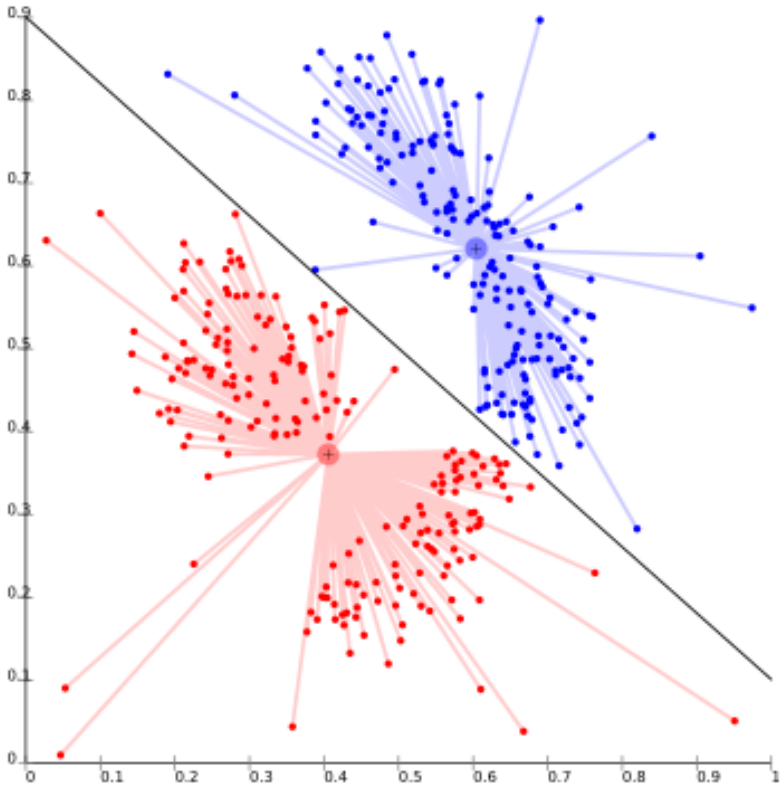
AVANÇAR

ANÁLISE COM ALGORITMOS DE CLUSTERING

CLUSTERIZAÇÃO BASEADA EM CENTROIDS

K-Means Clustering

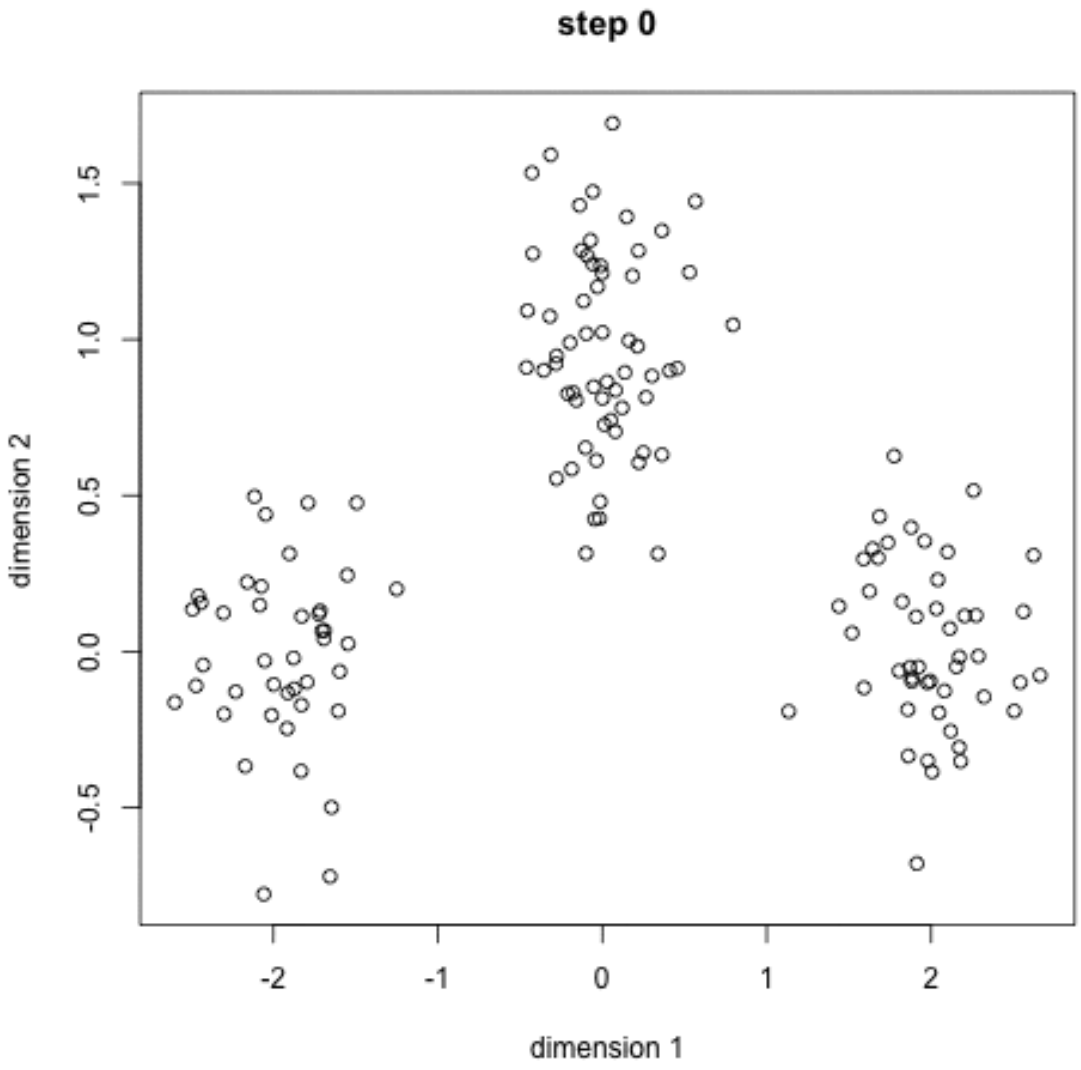
É um método de clusterização baseado em centróides (K-means), em que cada elemento do cluster tem como base a sua distância média de um dos centros determinados no momento de instanciar o modelo. Por exemplo n_clusters=2. Este número k, definido de antemão, será calculado por uma série de tentativas aleatórias o que minimiza a distância entre pontos no conjunto de dados e cada centro de cluster. O problema é conhecido como NP-hard e, portanto, as soluções são comumente aproximadas ao longo de uma série de tentativas. Porém o output do K-means será sempre catogórico: pertence a determinado cluster ou não sem a possibilidade de ambiguidades ou probabilidades.



K-Means Clustering Fonte:https://knowm.org/introduction-to-clustering/

A maior desvantagem dos algoritmos do tipo k-means é que eles exigem que o número de clusters k seja especificado com antecedência. Isso causa problemas para dados de clustering quando o número de clusters não pode

aquelas que não são lineares separáveis.



PASSO A PASSO DO ALGORITMO

01 Selecione a quantidade de classes que o algoritmo deve buscar, ele inicializa aleatoriamente os centroides. O algoritmo reinicia a busca randômica por centroides a cada round. Os pontos centrais, denotados como X no gráfico, são vetores com o mesmo comprimento de cada vetor de ponto de dados.

02 O K-means classifica cada data point calculando a distância entre os pontos específicos e o centro de cada grupo. O próximo passo é classificar os pontos que pertencem ao grupo cujo centro é o mais próximo deles.

ANÁLISE COM ALGORITMOS DE CLUSTERING

04 Repita o procedimento por um número de vezes e certifique-se de que os centros do grupo não variem muito entre as iterações.

PRÓS

- K-means é um método rápido porque não realiza muitos cálculos.
- Não realiza predições ambíguas.

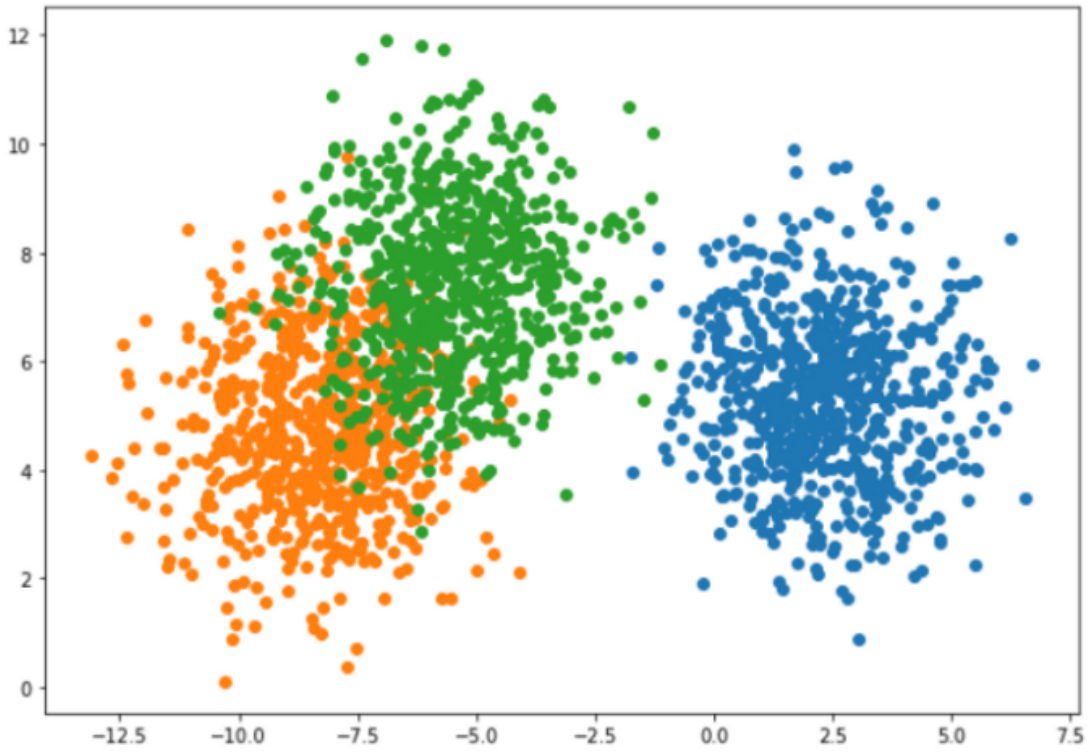
CONTRAS

- Identificar e classificar os grupos pode ser um aspecto desafiador.
- Como começa com uma escolha aleatória de centros de cluster, os resultados podem ser inconsistentes.

IMPLEMENTAÇÃO BÁSICA

ANÁLISE COM ALGORITMOS DE CLUSTERING

```
from numpy import where
fig= plt.figure(figsize=(10,7))
# define the model
model = KMeans(n_clusters=3)
# fit the model
model.fit(X)
# assign a cluster to each example
yhat = model.predict(X)
# retrieve unique clusters
clusters = unique(yhat)
# create scatter plot for samples from each cluster
for cluster in clusters:
    # get row indexes for samples with this cluster
    row_ix = where(yhat == cluster)
    # create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```

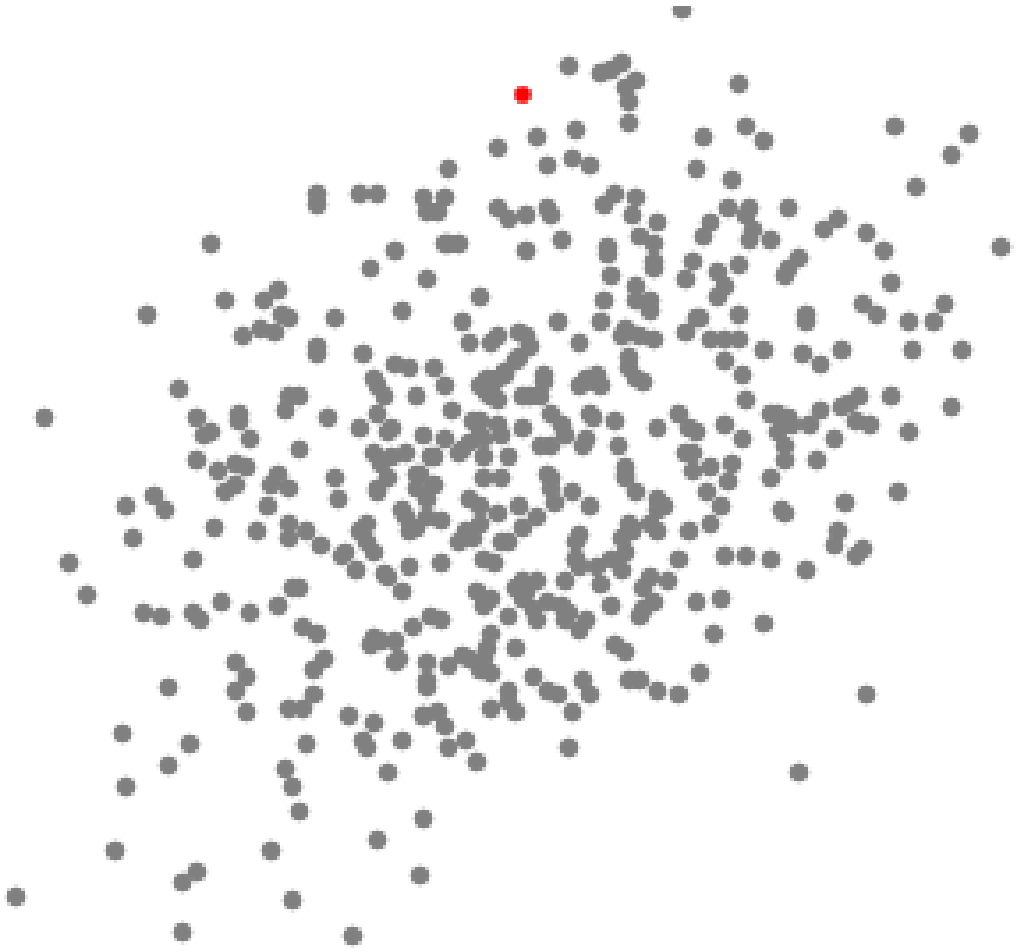


AVANÇAR

MEAN-SHIFT CLUSTERING

O Mean-Shift Clustering ou agrupamento de deslocamento médio (em uma tradução livre) é um algoritmo baseado em foco deslizante que tenta encontrar áreas densas nos data points. Também é um algoritmo baseado em centróides, o que significa que o objetivo é localizar os pontos centrais de cada grupo / classe. Para fazer isso, ele fica atualizando candidatos a pontos centrais a serem a média dos pontos dentro da desse foco ou janela deslizante. Essas janelas candidatas são filtradas em um estágio de pós-processamento para eliminar as similares, formando o conjunto final de pontos centrais e seus grupos correspondentes. Confira o gráfico abaixo para ver uma ilustração.

ANÁLISE COM ALGORITMOS DE CLUSTERING



MEAN-SHIFT CLUSTERING PELA PERSPECTIVA DE UMA ÚNICA JANELA

01 Para explicar o mean-shift ou a mudança de média, consideraremos um conjunto de pontos no espaço bidimensional como na ilustração acima. Começamos com uma janela deslizante circular centrada em um ponto C (selecionado aleatoriamente) e tendo o raio r como o núcleo. A mudança média é um algoritmo de escalada que envolve a mudança desse kernel iterativamente para uma região de densidade mais alta em cada etapa até a convergência.

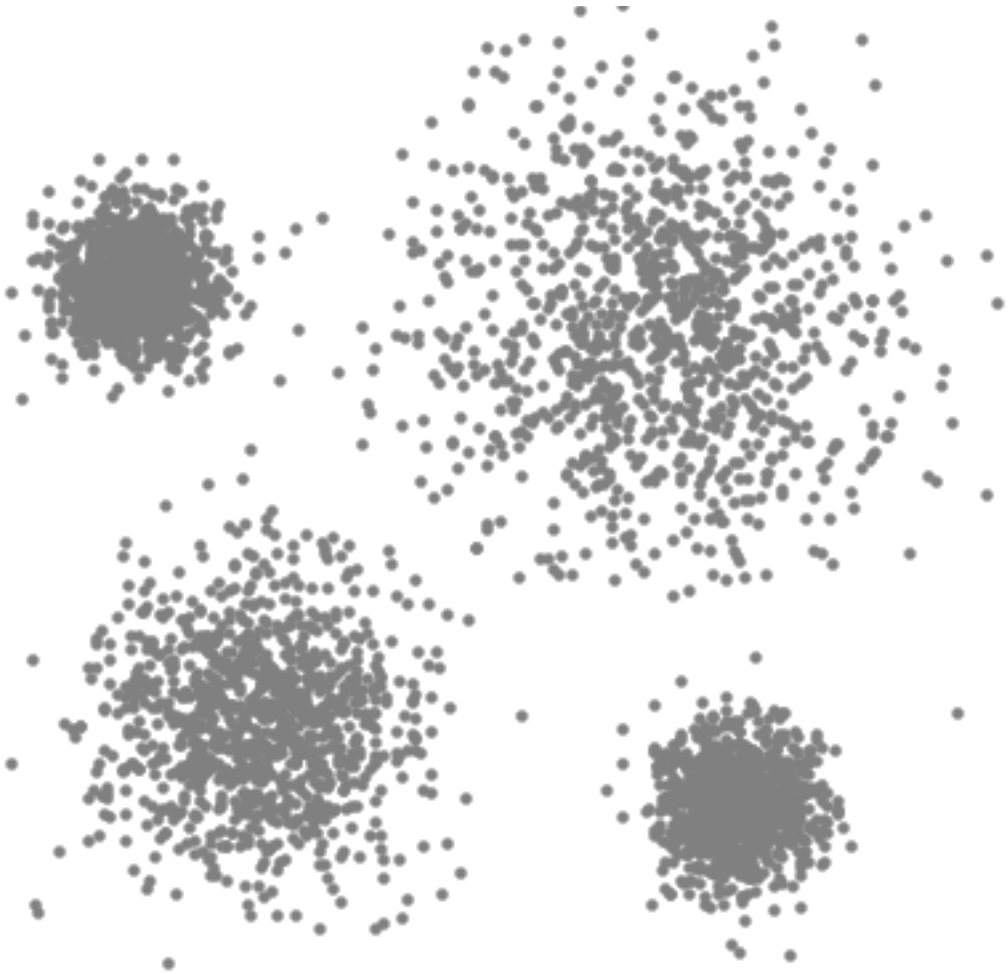
02 A cada iteração, a janela deslizante é deslocada para regiões de maior densidade, deslocando o ponto central para a média dos pontos dentro da janela (daí o nome). A densidade dentro da janela deslizante é proporcional ao número de pontos dentro dela. Naturalmente, ao mudar para a média dos pontos na janela,

03 Continuamos deslocando a janela deslizante de acordo com a média até que não haja uma direção na qual uma mudança possa acomodar mais pontos dentro do kernel. Confira o gráfico acima; continuamos movendo o círculo até que não aumentemos mais a densidade (ou seja, o número de pontos na janela).

04 Este processo das etapas 1 a 3 é feito com muitas janelas deslizantes até que todos os pontos fiquem dentro de uma janela. Quando várias janelas deslizantes se sobrepõem, a janela que contém a maioria dos pontos é preservada. Os pontos de dados são então agrupados de acordo com a janela deslizante na qual residem.

Uma ilustração do processo inteiro, de ponta a ponta com todas as janelas deslizantes é mostrada abaixo. Cada ponto preto representa o centroide de uma janela deslizante e cada ponto cinza é um ponto de dados.

ANÁLISE COM ALGORITMOS DE CLUSTERING



Uma visão gerado de todo o processo de Mean-Shift Clustering

Em contraste com o K-means, não há necessidade de selecionar o número de clusters, pois o Means-Shift descobre isso automaticamente. Essa é uma de suas maiores vantagens. O fato de os centros do cluster convergirem para os pontos de densidade máxima também é bastante desejável, pois é bastante intuitivo e se encaixa bem, no sentido data-driven. Sua maior desvantagem é que definir seleção do tamanho/raio da janela “r”, escolher este valor pode não ser trivial.

Implementação básica

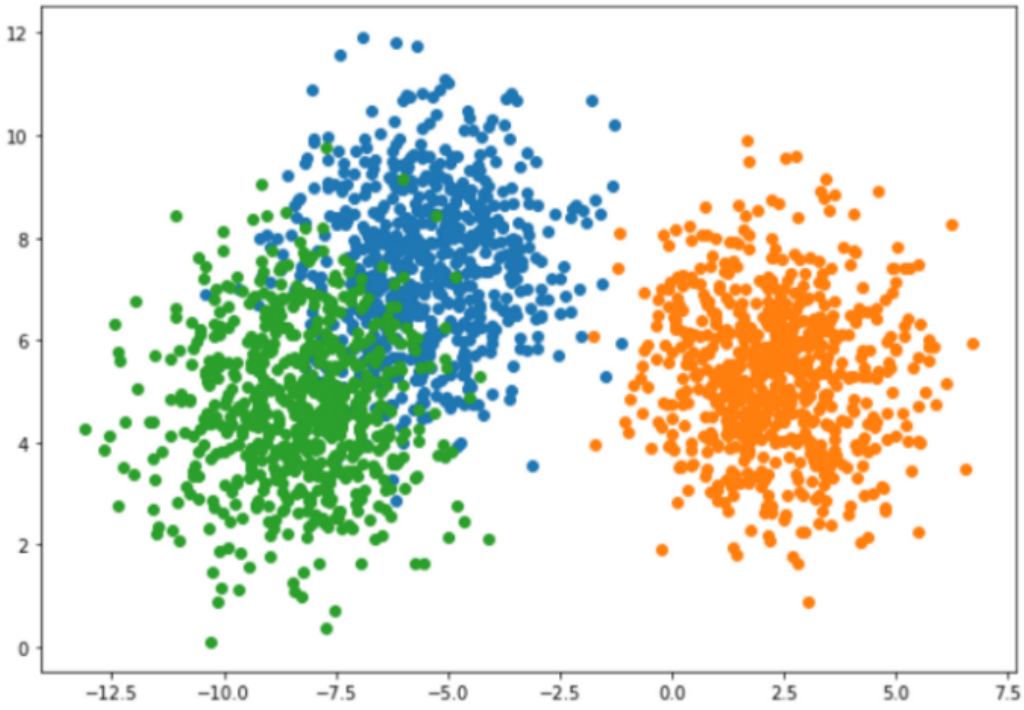
ANÁLISE COM ALGORITMOS DE CLUSTERING

```
# fit model and predict clusters
yhat = model.fit_predict(X)

# retrieve unique clusters
clusters = unique(yhat)

# create scatter plot for samples from each cluster
for cluster in clusters:
    # get row indexes for samples with this cluster
    row_ix = where(yhat == cluster)
    # create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])

# show the plot
pyplot.show()
```



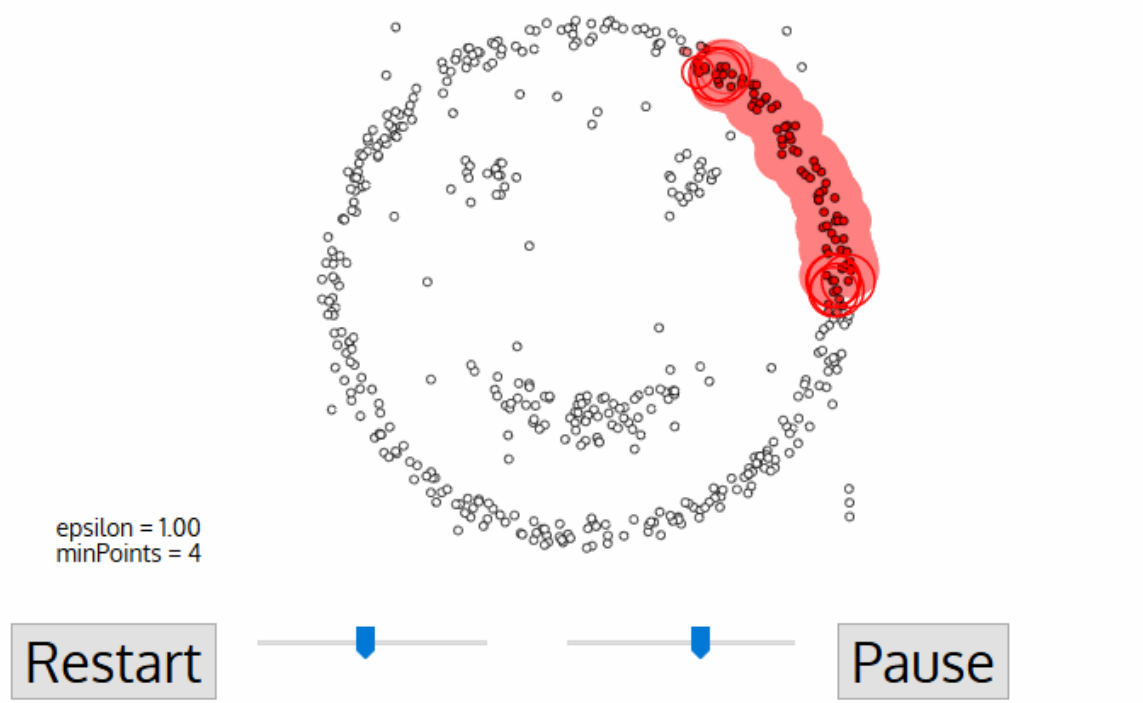
AVANÇAR

ANÁLISE COM ALGORITMOS DE CLUSTERING

CLUSTERIZAÇÃO BASEADA EM DENSIDADE

DBSCAN — Density-Based Spatial Clustering of Applications with Noise

O DBSCAN, um algoritmo de agrupamento baseado em densidade, é uma melhoria em relação ao agrupamento Mean-Shift, pois tem vantagens específicas. O gráfico a seguir pode esclarecer o assunto para você.



01 Ele começa com um ponto de dados inicial aleatório não visitado. A partir daí, todos os pontos dentro de uma distância 'Epsilon — Ɛ, são classificados como neighborhood points.

02 É preciso um número mínimo de pontos na vizinhança para iniciar o processo de clustering. Quando atingidas essas circuntâncias, o data point atual torna-se o primeiro ponto no

03 Todos os pontos dentro da distância Ɛ tornam-se parte do mesmo cluster. Repita o procedimento para todos os novos pontos adicionados ao grupo de cluster.

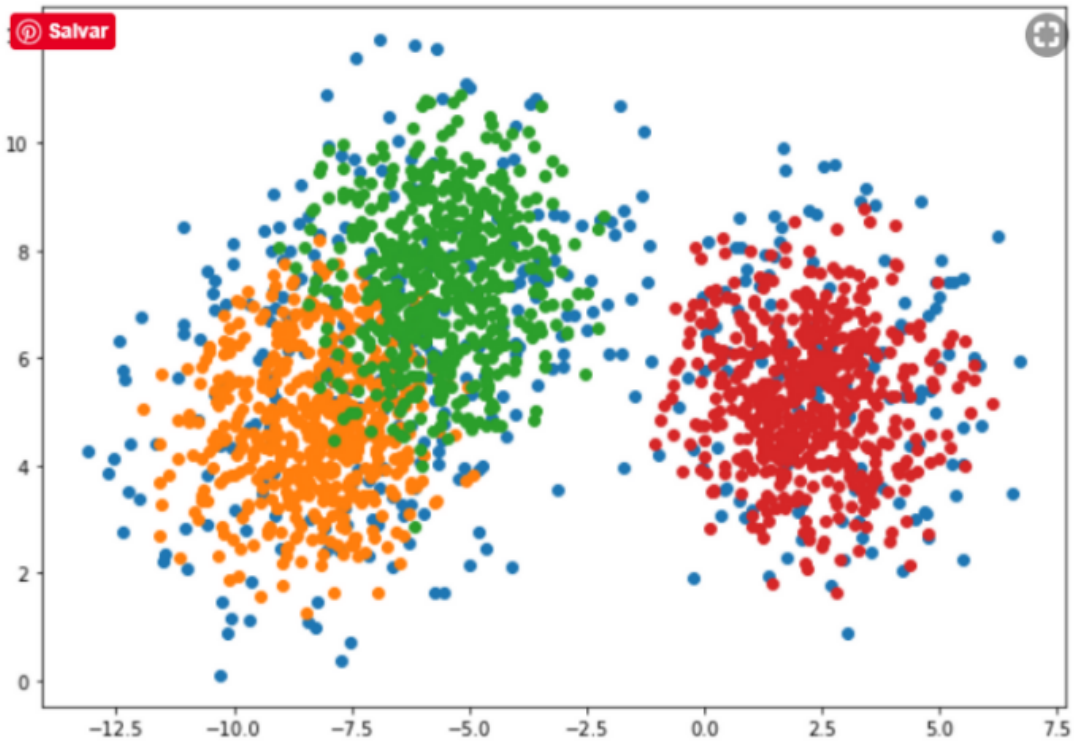
04 Continue com o processo até visitar e rotular cada ponto dentro da vizinhança Ɛ do cluster.

05 Após a conclusão do processo, comece novamente com um novo ponto não visitado, levando assim à descoberta de mais clusters ou de ruído. No final do processo, certifique-se de marcar cada ponto como cluster ou ruído.

IMPLEMENTAÇÃO BÁSICA

ANÁLISE COM ALGORITMOS DE CLUSTERING

```
model = DBSCAN(eps=1.8, min_samples=10)
# fit model and predict clusters
yhat = model.fit_predict(X)
# retrieve unique clusters
clusters = unique(yhat)
# create scatter plot for samples from each cluster
for cluster in clusters:
    # get row indexes for samples with this cluster
    row_ix = where(yhat == cluster)
    # create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```



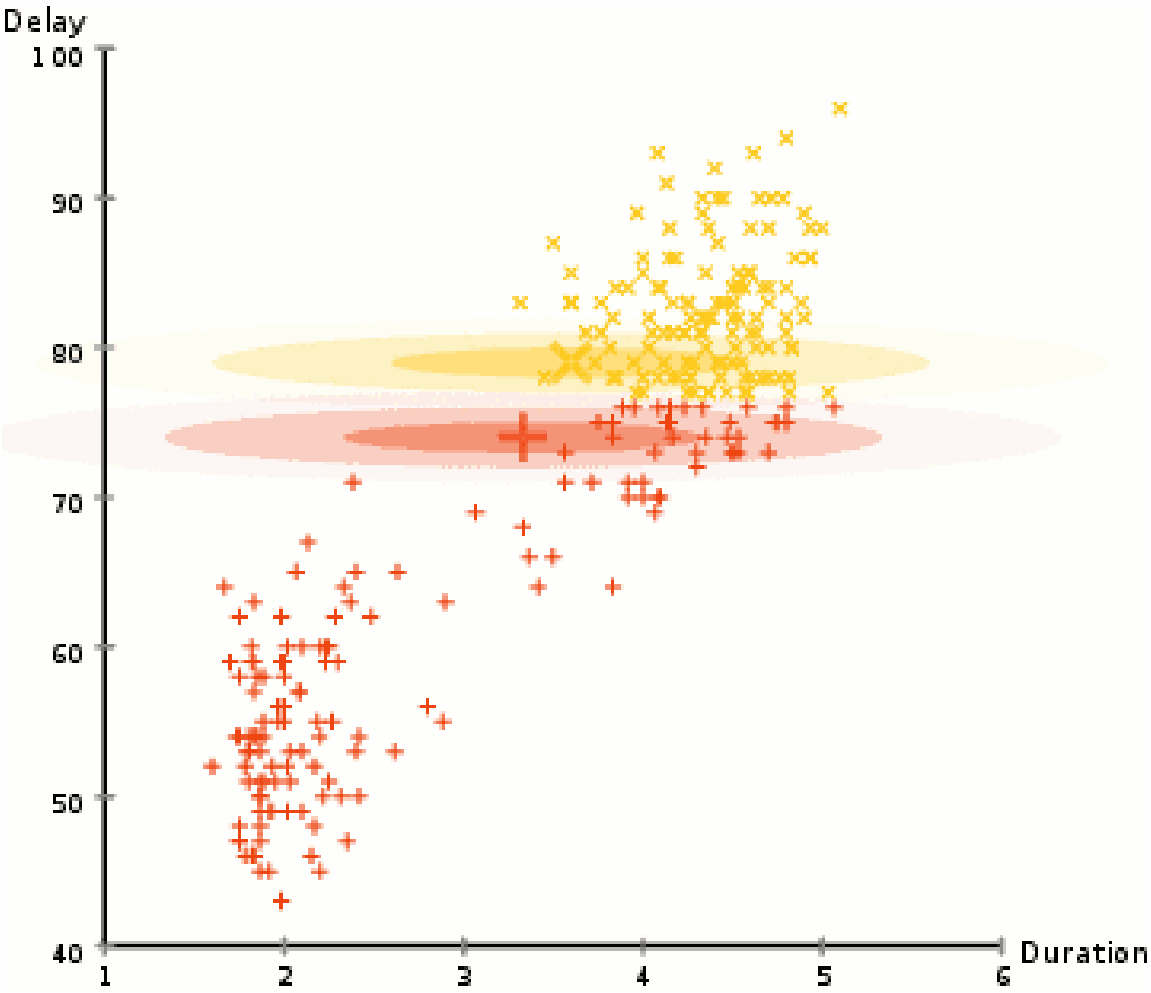
AVANÇAR

ANÁLISE COM ALGORITMOS DE CLUSTERING

CLUSTERIZAÇÃO BASEADA EM DISTRIBUIÇÃO

Expectation–Maximization (EM) Clustering using Gaussian Mixture Models (GMM)

Uma das principais desvantagens do K-Means é seu uso ingênuo ou simplista do K-Means para determinar o centro do cluster. Observando a imagem abaixo, podemos ver porque usar o K-Means não é a melhor maneira de fazer esta escolha. No lado esquerdo, parece bastante óbvio visualmente que existem dois clusters circulares com raios diferentes centrados na mesma média. O K-Means não pode lidar com isso porque os valores médios dos clusters estão muito próximos. Sem falar que o K-Means também falha nos casos em que os clusters não são circulares, novamente porque usa como resultado do a média como centro do cluster.



ANÁLISE COM ALGORITMOS DE CLUSTERING

Gaussiana, esta é uma suposição menos restritiva do que dizer que eles são circulares usando a média. Dessa forma, **temos dois parâmetros para descrever a forma dos clusters: a média e o desvio padrão!** Tomando um exemplo em duas dimensões, isso significa que os clusters podem assumir qualquer tipo de forma elíptica (uma vez que temos um desvio padrão nas direções x e y). Assim, cada distribuição gaussiana é atribuída a um único cluster.

Para encontrar os parâmetros do Gaussiano para cada cluster (por exemplo, a média e o desvio padrão), usaremos um algoritmo de otimização chamado Expectation — Maximization (EM). Dê uma olhada no gráfico abaixo como uma ilustração das gaussianas sendo ajustadas aos clusters. Em seguida, podemos prosseguir com o processo de agrupamento de Expectativa-Maximização usando GMMs.

01 Semelhante ao cluster K-means, selecionamos o número de clusters e inicializamos aleatoriamente os parâmetros de distribuição gaussiana para cada um deles.

02 Com esse background, calcule a probabilidade de cada ponto de dados pertencer a um determinado cluster. Quanto mais próximo o ponto estiver do centro da Gauss, maiores são as chances de ele pertencer ao cluster.

03 Com base nesses cálculos, determinamos um novo conjunto de parâmetros para as distribuições gaussianas para maximizar as probabilidades de pontos de dados dentro dos clusters. Usamos uma soma ponderada das posições dos pontos de dados para calcular essas probabilidades. A probabilidade de o ponto de dados pertencer a um determinado cluster é o fator de peso

04 Repita os passos 2 e 3 até a convergência onde não há muita variação.

PROS

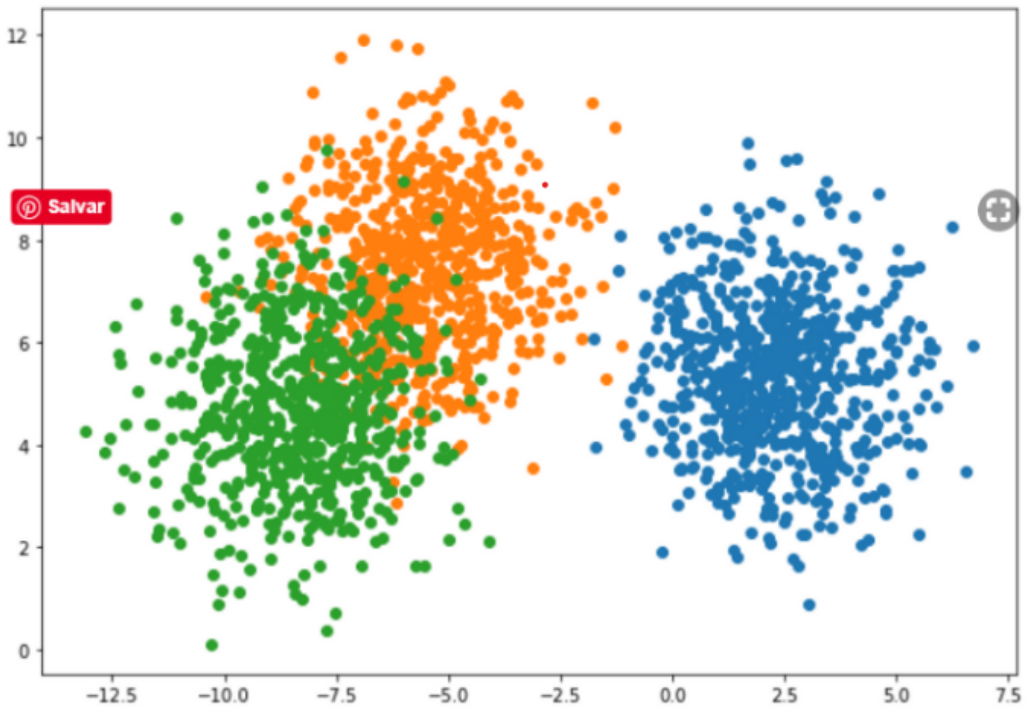
conceito de desvio padrão.

- Como esse conceito usa probabilidade, você tem vários clusters por data points. Portanto, se um determinado data point pertence a dois clusters sobrepostos, podemos defini-lo de forma ainda mais precisa dizendo que pertence A% à Classe 1 e B% à Classe 2.

IMPLEMENTAÇÃO BÁSICA

ANÁLISE COM ALGORITMOS DE CLUSTERING

```
# define the model
model = GaussianMixture(n_components=3)
# fit the model
model.fit(X)
# assign a cluster to each example
yhat = model.predict(X)
# retrieve unique clusters
clusters = unique(yhat)
# create scatter plot for samples from each cluster
for cluster in clusters:
    # get row indexes for samples with this cluster
    row_ix = where(yhat == cluster)
    # create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```

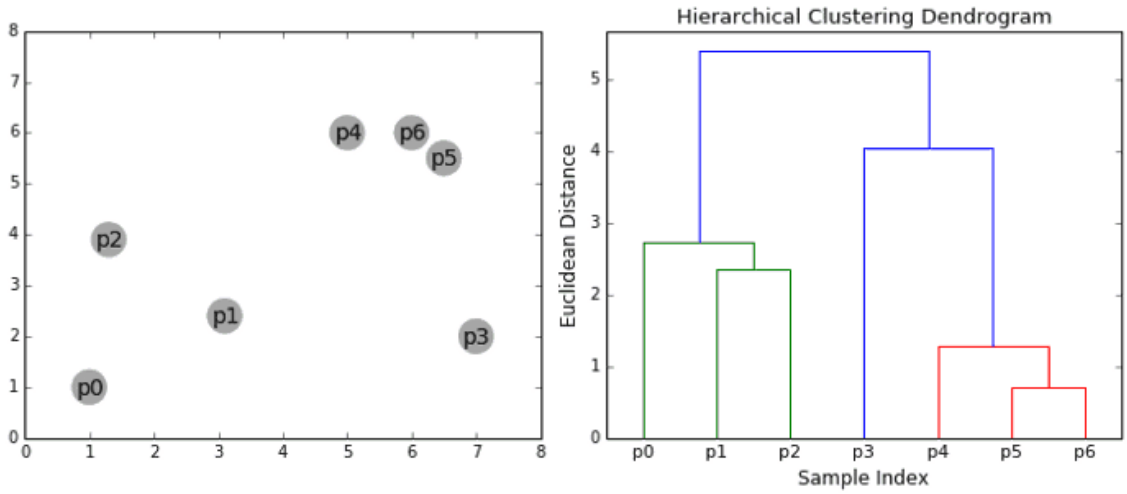


AVANÇAR

CLUSTERIZAÇÃO POR CONECTIVIDADE

Agglomerative Hierarchical Clustering

Os algoritmos de Hierarchical Clustering ou agrupamento hierárquico se enquadram em 2 categorias: *top-down* ou *bottom-up*. Algoritmos *bottom-up* ou ascendentes tratam cada ponto de dados como um único cluster no início e então mesclam (ou aglomeram) pares de clusters sucessivamente até que todos os clusters tenham sido mesclados em um único cluster que contém todos os pontos de dados. O agrupamento hierárquico *bottom-up* é, portanto, denominado agrupamento aglomerativo hierárquico ou HAC. Esta hierarquia de clusters é representada como uma árvore (ou dendrograma). A raiz da árvore é o único cluster que reúne todas as amostras, sendo as folhas os aglomerados com apenas uma amostra. Confira o gráfico abaixo para ver uma ilustração antes de passar para as etapas do algoritmo.



Funcionamento do Agglomerative Hierarchical Clustering

01 Começamos tratando cada ponto de dados como um único cluster, ou seja, se houver X pontos de dados em nosso conjunto de dados, teremos X clusters. Em seguida, selecionamos uma métrica de distância que mede a distância entre dois clusters. Como exemplo, usaremos o average linkage, que define a distância entre dois clusters como a distância média entre os

ANÁLISE COM ALGORITMOS DE CLUSTERING

02 Em cada iteração, combinamos dois clusters em um. Os dois clusters a serem combinados são selecionados como aqueles com a menor ligação média. Ou seja, de acordo com nossa métrica de distância selecionada, esses dois clusters têm a menor distância entre si e, portanto, são os mais semelhantes e devem ser combinados.

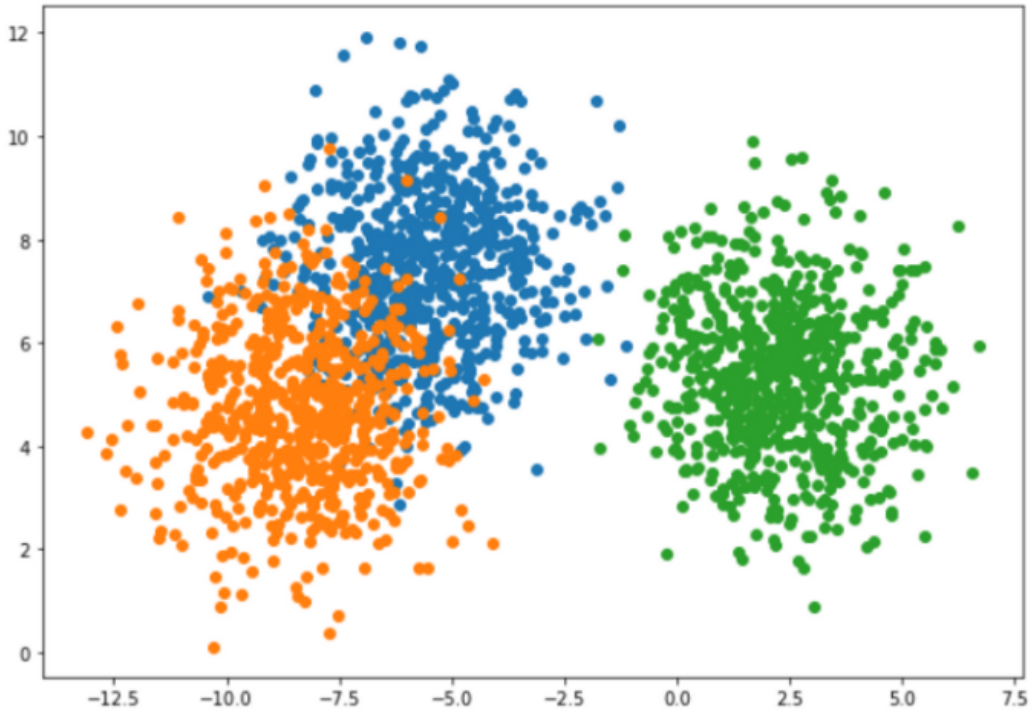
03 A etapa 2 é repetida até chegarmos à raiz da árvore, ou seja, temos apenas um cluster que contém todos os data points. Desta forma, podemos selecionar quantos clusters queremos no final, simplesmente escolhendo quando parar de combinar os clusters, ou seja, quando pararmos de construir a árvore!

O clustering hierárquico não exige que especifiquemos o número de clusters e podemos até selecionar qual número de clusters parece melhor, já que estamos construindo uma árvore. Além disso, o algoritmo não é sensível à escolha da métrica de distância; todos eles tendem a funcionar igualmente bem, enquanto com outros algoritmos de agrupamento, a escolha da métrica de distância é crítica. **Um caso de uso particularmente bom de métodos de agrupamento hierárquico é quando os dados subjacentes têm uma estrutura hierárquica e você deseja recuperar a hierarquia; outros algoritmos de agrupamento não podem fazer isso.** Essas vantagens do agrupamento hierárquico vêm ao custo de menor eficiência, pois tem uma complexidade de tempo de O (n^3), ao contrário da complexidade linear de K-Means e GMM.

IMPLEMENTAÇÃO BÁSICA

ANÁLISE COM ALGORITMOS DE CLUSTERING

```
model = AgglomerativeClustering(n_clusters=3)
# fit model and predict clusters
yhat = model.fit_predict(X)
# retrieve unique clusters
clusters = unique(yhat)
# create scatter plot for samples from each cluster
for cluster in clusters:
    # get row indexes for samples with this cluster
    row_ix = where(yhat == cluster)
    # create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```



AVANÇAR

EXEMPLOS DE USOS DE ALGORITMOS DE CLUSTERING

SISTEMAS DE DIAGNÓSTICO

A profissão médica usa o K-Means na criação de sistemas de apoio à decisão médica mais inteligentes, especialmente no tratamento de doenças do fígado.

MOTORES DE BUSCA

A clusterização é a espinha dorsal dos motores de busca. Quando uma pesquisa é realizada, os resultados da pesquisa precisam ser agrupados e os mecanismos de pesquisa frequentemente usam clustering para fazer isso.

REDES DE SENSORES SEM FIO

O algoritmo de clustering desempenha o papel de localizar os cluster heads, que coletam todos os dados em seu respectivo cluster.

MARKETING E VENDAS

Personalização e targeting em marketing são um grande negócio. Isso é conseguido examinando as características específicas de uma pessoa e compartilhando com ela as campanhas que tiveram sucesso com outras pessoas semelhantes.

Qual é o problema? Se sua empresa está tentando obter o melhor retorno sobre seu investimento em marketing, é crucial que você faça o targeting das pessoas de uma maneira correta. Se errar, você corre o risco de não realizar nenhuma venda ou, pior, prejudicar a confiança do cliente na sua empresa.

Como os clusters funcionam? Os algoritmos de clustering são capazes de agrupar pessoas com características semelhantes e probabilidade de compra. Depois de ter os grupos, você pode executar testes em cada grupo com uma cópia de marketing diferente que o ajudará a direcionar melhor suas mensagens para eles no futuro.

ANÁLISE COM ALGORITMOS DE CLUSTERING

ANÁLISE COM ALGORITMOS DE CLUSTERING

IMPLEMENTANDO 2 DOS ALGORITMOS MAIS COMUNS

K-Means Clustering

Vamos trabalhar em um problema de segmentação de clientes de varejo. Você pode baixar o conjunto de dados usando este link.

O objetivo desse dataset é segmentar os clientes de um varejista com base em seus gastos anuais em diversas categorias de produtos, como leite, mercearia, região, etc. Então, vamos começar!

Antes de tudo, vamos importar as bibliotecas:

```
# importando as bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.cluster import KMeans

# investigando a cara dos dados
df=pd.read_csv("Wholesale customers data.csv")
df.head()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

Temos os dados de gastos dos clientes em diferentes produtos como Leite, Comestíveis, Congelados, Detergentes, etc. Agora, temos que segmentar os clientes com base nos detalhes fornecidos. Antes de fazer isso, vamos retirar algumas estatísticas relacionadas aos dados:

ANÁLISE COM ALGORITMOS DE CLUSTERING

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000
mean	1.322727	2.543182	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1524.870455
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2820.105937
min	1.000000	1.000000	3.000000	55.000000	3.000000	25.000000	3.000000	3.000000
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000	742.250000	256.750000	408.250000
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	965.500000
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1820.250000
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	47943.000000

Aqui, vemos que há muita variação na magnitude dos dados. Variáveis como Channel e Region têm magnitude baixa, enquanto variáveis como Fresh, Milk, Grocery, etc. têm magnitude maior.

Sabemos que o K-Means é um algoritmo baseado em distância. Essas diferenças de magnitude podem criar um problema. Então, é imprescindível primeiro trazer todas as variáveis para a mesma magnitude:

```
# padronizando os dados
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)

# statisticas do dataset padronizado
pd.DataFrame(df_scaled).describe()
```

	0	1	2	3	4	5	6	7
count	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02
mean	-2.452584e-16	-5.737834e-16	-2.422305e-17	-1.589638e-17	-6.030530e-17	1.135455e-17	-1.917658e-17	-8.276208e-17
std	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00
min	-6.902971e-01	-1.995342e+00	-9.496831e-01	-7.787951e-01	-8.373344e-01	-6.283430e-01	-6.044165e-01	-5.402644e-01
25%	-6.902971e-01	-7.023369e-01	-7.023339e-01	-5.783063e-01	-6.108364e-01	-4.804306e-01	-5.511349e-01	-3.964005e-01
50%	-6.902971e-01	5.906683e-01	-2.767602e-01	-2.942580e-01	-3.366684e-01	-3.188045e-01	-4.336004e-01	-1.985766e-01
75%	1.448652e+00	5.906683e-01	3.905226e-01	1.890921e-01	2.849105e-01	9.946441e-02	2.184822e-01	1.048598e-01
max	1.448652e+00	5.906683e-01	7.927738e+00	9.183650e+00	8.936528e+00	1.191900e+01	7.967672e+00	1.647845e+01

Agora sim! Vamos inicializar o K-Means:

ANÁLISE COM ALGORITMOS DE CLUSTERING

```
# fittando o k means nos dados padronizados
kmeans.fit(df_scaled)
```

	0	1	2	3	4	5	6	7
count	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02
mean	-2.452584e-16	-5.737834e-16	-2.422305e-17	-1.589638e-17	-6.030530e-17	1.135455e-17	-1.917658e-17	-8.276208e-17
std	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00
min	-6.902971e-01	-1.995342e+00	-9.496831e-01	-7.787951e-01	-8.373344e-01	-6.283430e-01	-6.044165e-01	-5.402644e-01
25%	-6.902971e-01	-7.023369e-01	-7.023339e-01	-5.783063e-01	-6.108364e-01	-4.804306e-01	-5.511349e-01	-3.964005e-01
50%	-6.902971e-01	5.906683e-01	-2.767602e-01	-2.942580e-01	-3.366684e-01	-3.188045e-01	-4.336004e-01	-1.985766e-01
75%	1.448652e+00	5.906683e-01	3.905226e-01	1.890921e-01	2.849105e-01	9.946441e-02	2.184822e-01	1.048598e-01
max	1.448652e+00	5.906683e-01	7.927738e+00	9.183650e+00	8.936528e+00	1.191900e+01	7.967672e+00	1.647845e+01

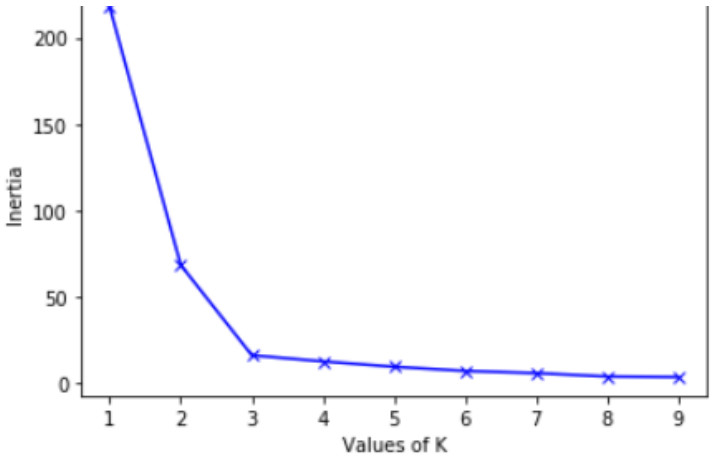
Vamos avaliar o quão bem estão os clusters formados. Para fazer isso, vamos calcular a inércia dos clusters. A inércia é um indicativo de quão estáveis os clusters estão. Ou seja, caso eu promova mais n-iterações, quão diferentes os clusters vão ser entre si a cada nova rodada :Vamos avaliar o quão bem estão os clusters formados. Para fazer isso, vamos calcular a inércia dos clusters. A inércia é um indicativo de quão estáveis os clusters estão. Ou seja, caso eu promova mais n-iterações, quão diferentes os clusters vão ser entre si a cada nova rodada:

 # INERTIA ON THE FITTED DATA
KMEANS.INERTIA_

Output: 2599.38555935614

Obtivemos um valor de inércia de quase 2600. Isso é bom? Não sabemos! Uma forma de avaliar isto é através do método elbow ou cotovelo. Em datasets com um número de clusters muito claros, o método registra uma queda significativa na inércia, formando um cotovelo no gráfico. Mais ou menos assim:

ANÁLISE COM ALGORITMOS DE CLUSTERING



Método do cotovelo. Fonte: https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/

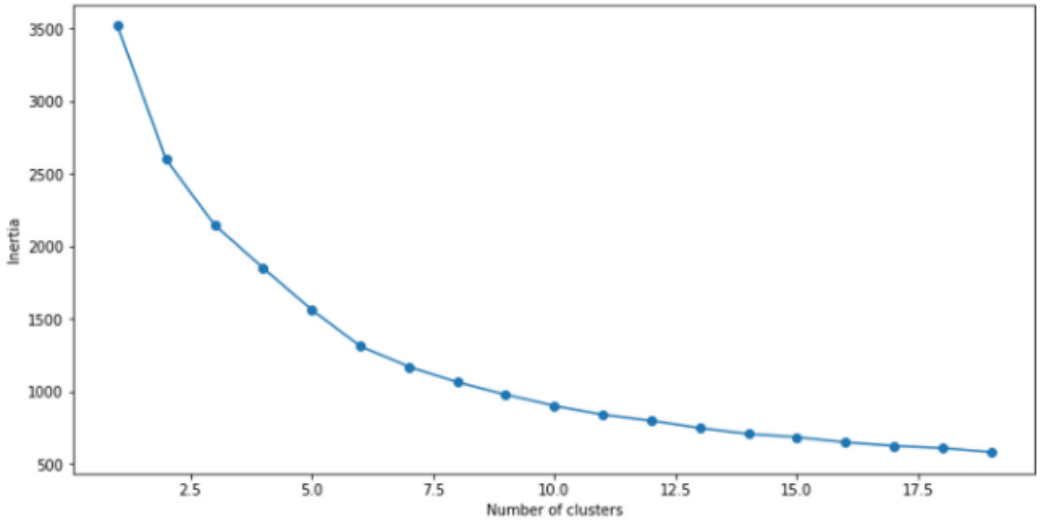
Agora, vamos ver como podemos usar a curva de cotovelo para determinar o número ideal de clusters no nosso dataset.

Vamos fazer um loop for para fittar diversos modelos KMeans, e, a cada modelo vamos aumentar o número de clusters. Vamos armazenar o valor de inércia de cada modelo e, em seguida, plotá-lo para visualizar o resultado:

ANÁLISE COM ALGORITMOS DE CLUSTERING

```
    kmeans = KMeans(n_clusters = cluster, init='k-means++')
    kmeans.fit(data_scaled)
    SSE.append(kmeans.inertia_)

# convertendo os resultados em um dataframe e plotando
frame = pd.DataFrame({'Cluster':range(1,20), 'SSE':SSE})
plt.figure(figsize=(12,6))
plt.plot(frame['Cluster'], frame['SSE'], marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
```



Como podemos perceber, nossos dados não formam um cotovelo tão nítido, mas vemos uma mudança de ângulo significativa entre 5 e 8 clusters

Então podemos escolher qualquer número de clusters entre 5 e 8. Vamos definir o número de clusters como 6 e ajustar o modelo:

ANÁLISE COM ALGORITMOS DE CLUSTERING

```
pred = kmeans.predict(df_scaled)

# criamos um novo dataframe com os dados do df original
# adicionamos a etiqueta dos clusters

df_final = pd.DataFrame(df)
df_final['cluster'] = pred
df_final['cluster'].value_counts()
```

```
3    172
2    124
0     86
1     46
4     11
5      1
Name: cluster, dtype: int64
```

Essa é a quantidade de observações para cada cluster. Vale ressaltar que ainda há uma inercia alta, então para cada vez que você rodar o modelo, vai haver uma leve mudança na quantidade e nas composições dos mode

Agora que temos os clusters, o que os diferencia?

DESAFIO: FAÇA UMA EDA E TENTE JUSTIFICAR AS DIFERENÇAS ENTRE OS DIVERSOS CLUSTERS.

```
df_final[df_final['cluster']==0]
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	cluster
198	1	1	11686	2154	6824	3527	592	697	0
199	1	1	9670	2280	2112	520	402	347	0
202	1	1	25203	11487	9490	5065	284	6854	0
203	1	1	583	685	2216	469	954	18	0
204	1	1	1956	891	5226	1383	5	1328	0
...	...	...	...	...	...	...	...	...	...
332	1	2	22321	3216	1447	2208	178	2602	0
336	1	2	13970	1511	1330	650	146	778	0
337	1	2	9351	1347	2611	8170	442	868	0
338	1	2	3	333	7021	15601	15	550	0
339	1	2	2617	1188	5332	9584	573	1942	0

Hierarchical Clustering

ANÁLISE COM ALGORITMOS DE CLUSTERING

covariância em uma relação hierárquica. Entre as diversas aplicações está a capacidade de clusterizar ações que oscilam juntas. Isto pode ser particularmente interessante para ajudar traders e investidores a formarem uma carteira variada.
Para poder ilustrar este exemplo, vamos usar dados de 60 ativos de empresas americanas referente ao período de 2010 a 2015. Primeiro vamos fazer as importações necessárias:

```
# Import the dendrogram function
from scipy.cluster.hierarchy import dendrogram
# Import the fcluster and linkage functions
from scipy.cluster.hierarchy import linkage, fcluster

df_stock = pd.read_csv('company-stock-movements-2010-2015-incl.csv')
df_stock.head()
```

	Unnamed: 0	2010-01-04	2010-01-05	2010-01-06	2010-01-07	2010-01-08	2010-01-11	2010-01-12	2010-01-13	2010-01-14	...	2013-10-16	2013-10-17	2013-10-18
0	Apple	0.580000	-0.220005	-3.409998	-1.170000	1.680011	-2.689994	-1.469994	2.779997	-0.680003	...	0.320008	4.519997	2.899987
1	AIG	-0.640002	-0.650000	-0.210001	-0.420000	0.710001	-0.200001	-1.130001	0.069999	-0.119999	...	0.919998	0.709999	0.119999
2	Amazon	-2.350006	1.260009	-2.350006	-2.009995	2.960006	-2.309997	-1.640007	1.209999	-1.790001	...	2.109985	3.699982	9.570008
3	American express	0.109997	0.000000	0.260002	0.720002	0.190003	-0.270001	0.750000	0.300004	0.639999	...	0.680001	2.290001	0.409996
4	Boeing	0.459999	1.770000	1.549999	2.690003	0.059997	-1.080002	0.360000	0.549999	0.530002	...	1.559997	2.480003	0.019997

5 rows × 964 columns

Para poder processar os nossos dados, vamos precisar extrair os valores do nosso dataframe das variáveis numéricas e transformá-los em uma array. Ao mesmo tempo, vamos precisar extrair o nome das empresas e armazenar em uma lista:

```
companies = df_stock.iloc[:,0].to_list()
movements = df_stock.drop(df_stock.columns[0], axis=1).values
```

Os algoritmos de clustering hierárquicos são sensíveis à distância, então para não deixar que eles sejam influenciados por diferentes escalas, mas apenas pelas variações, vamos precisar normalizar os nossos dados:

ANÁLISE COM ALGORITMOS DE CLUSTERING

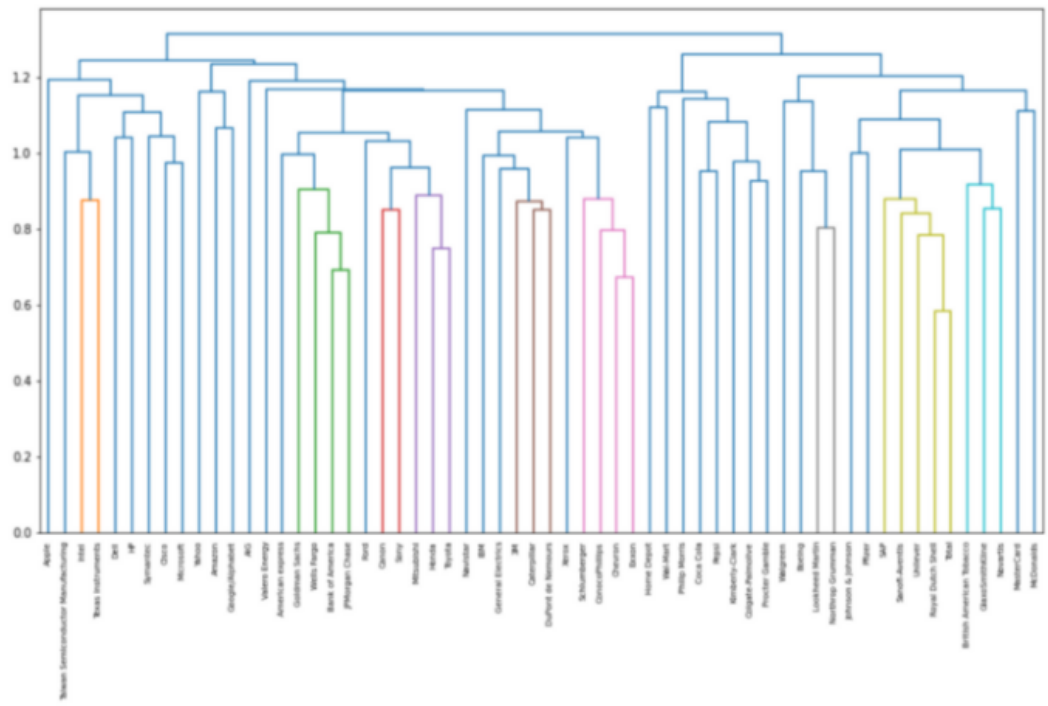
```
# Normalize the movements: normalized_movements
normalized_movements = normalize(movements)
```

Agora conseguimos gerar uma visualização que irá nos dar um entendimento da formação dos clusters hierárquicos:

```
# set the fig size
fig= plt.figure(figsize=(15,8))

# Calculate the linkage: mergings
mergings = linkage(normalized_movements, method='complete')

# Plot the dendrogram
dendrogram(mergings,
           labels=companies,
           leaf_rotation=90,
           leaf_font_size=7
            )
plt.show()
```



WOW! Olha o link entre a variação dos papéis de todas estas empresas!

Agora precisamos extrair os clusters para fazer as nossas recomendações.

ANÁLISE COM ALGORITMOS DE CLUSTERING

por exemplo, teremos 60 clusters. Um para cada empresa. Se fizermos acima de 1.4 teremos apenas um grande cluster com as 60 empresas. Precisamos estabelecer um corte que otimize e capture o que estamos buscando (a covariância dos ativos).

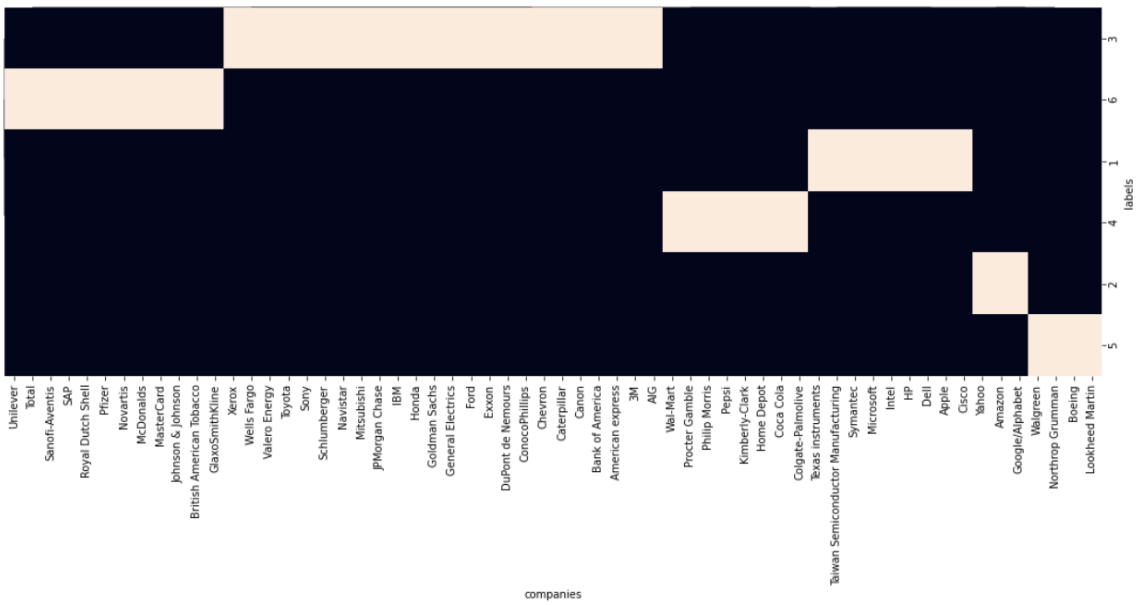
```
# fazendos as importações necessárias
from scipy.cluster.hierarchy import fcluster

# Escolhendo o ponto de corte da nossa árvore do dendrograma
# Faça uma alteração nesse valor mudam os clusters
labels = fcluster(mergings, 1.2, criterion='distance' )

# criando um dataframe com as labes e as empresas= df
df = pd.DataFrame({'labels': labels, 'companies': companies})

# Criando uma crosstab: ct
ct = pd.crosstab(df['labels'], df['companies'])

# transformando a crosstab em um cluster map
sns.clustermap(ct, figsize=(15, 8), cbar_pos=None,
dendrogram_ratio=0.001)
```



Agora conseguimos enxergar facilmente os clusters e as empresas que os compõem!

E o melhor de tudo, já temos um dataframe com a label de cada cluster e o nome de todas as empresas…

ANÁLISE COM ALGORITMOS DE CLUSTERING

11	1	Cisco
14	1	Dell
22	1	HP
24	1	Intel
33	1	Microsoft
47	1	Symantec
50	1	Taiwan Semiconductor Manufacturing
51	1	Texas instruments

DESAFIO: CRIE UM NOME PARA CADA LABEL QUE AJUDE AOS INVESTIDORES A IDENTIFICAR QUAIS EMPRESAS ESTÃO EM UM DETERMINADO CLUSTER.

AVANÇAR

ANÁLISE COM ALGORITMOS DE CLUSTERING

CONCLUSÃO

Que bom que você chegou até aqui!

Chegamos ao fim dos estudos básicos de algoritmos não-supervisionados. Durante essa jornada, passamos pelos principais tipos de algoritmos de clusterização passando por exemplos dos mais conhecidos e no final, podemos ver o KMeans e o Cluster Hierárquico em uma aplicação de datasets reais. Estas novas habilidades são fundamentais para a caixa de ferramenta de profissionais de ciência de dados.

Com este texto e o de Redução de dimensionalidade, finalizamos o bloco de aprendizado não supervisionado.

Como sempre, esperamos que tenha ficado claro cada parte dos nossos textos, mas se a qualquer momento surgir uma dúvida, lembre-se, tem sempre alguém do time Tera pronto para te ajudar!

O QUE ACHOU DESTA AULA?

Deixe seu feedback para continuarmos melhorando sua experiência.

1 MIN

AVALIAR

VOLTAR PARA O CURSO