

**CENTRO UNIVERSITÁRIO FARIAS BRITO**  
**CIÊNCIA DA COMPUTAÇÃO**

INTELIGÊNCIA ARTIFICIAL  
PROFESSOR: JOSÉ BELO ARAGÃO JÚNIOR

**RELATÓRIO:**  
**AGRUPAMENTO K-MEANS**

EQUIPE:  
Pedro Victor da Silva Ávila - Matrícula:1620237  
Renê Victor Lucas - Matrícula:1611181  
Samuel Benevides Linhares - Matrícula:1721168

## INTRODUÇÃO

Em Ciência de Dados, o **agrupamento k-means** (do inglês *k-means clustering*) é um método de quantização de vetores muito usado na análise de agrupamentos (*clusters*). A técnica consiste em particionar  $n$  observações em  $k$  agrupamentos onde cada observação pertence ao grupo com a mais próxima média. Isso resulta em uma divisão de espaço de dados conhecido como células de Voronoi.

### Como funciona o Kmeans na prática:

1. Selecioná-se os centróides iniciais, normalmente, a seleção de centróides é feita de forma aleatória, mas para essa atividade, foi proposto utilizar centróides de um arquivo csv;
2. Verifica para cada instância qual o centróide mais próximo;

### 2.1-Como saber qual o centróide mais próximo?

Para isso, devemos calcular a distância euclidiana de cada instância para cada um dos centróides:

#### Fórmula:

Distância entre duas instâncias  $\mathbf{p}_i$  e  $\mathbf{p}_j$  definida como:

$$d = \sqrt{\sum_{k=1}^n (p_{ik} - p_{jk})^2}$$

$p_{ik}$  e  $p_{jk}$  para  $k = 1, \dots, n$  são os  $n$  atributos que descrevem as instâncias  $\mathbf{p}_i$  e  $\mathbf{p}_j$ , respectivamente

Outra forma de se interpretar essa fórmula:

$$d_{(p1,p2)} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

nos dois casos acima,  $p1$  e  $p2$  são pontos nos quais iremos calcular a distância.

No desse trabalho, cada instância(também podendo ser chamada de ponto) possui 4 valores( $x_1, x_2, x_3, x_4$ ) e cada centróide também possui 4 valores( $x_1, x_2, x_3, x_4$ ).

Dessa forma, uma forma de se ver o calcula da distância euclidiana é:

$$\sqrt{((p2x1-p1x1)^2+(p2x2-p1x2)^2+(p2x3-p1x3)^2+(p2x4-p1x4)^2)}$$

Também podendo ser representado dessa forma:

$\text{math.sqrt}(\text{pow}(p2x1-p1x1,2)+\text{pow}(p2x2-p1x2,2)+\text{pow}(p2x3-p1x3,2)+\text{pow}(p2x4-p1x4,2))$

3. Depois de identificar a qual classe(cluster) cada instância pertence, deve-se pegar a posição dos novos clusters(dos novos centróides). Para isso deve-se fazer a somatória de  $x_1, x_2, x_3$  e  $x_4$  de todas as instâncias de cada cluster e em seguida dividir pela quantidade de instâncias do cluster. Falando de outra forma, é necessário tirar a média de  $x_1, x_2, x_3$  e  $x_4$  para cada cluster, assim a média será a nova posição dos clusters;

4. Depois de feito isso é necessário repetir os processos 2 para frente até que se estabilize. Para saber se os clusters estabilizaram basta verificar se os clusters de cada instância pararam de mudar. Quando clusters de todas as instâncias pararam de mudar, é hora de parar o loop.

Esses processos acima devem ser feitos para valores de  $k$  de 2 a 12;

Depois de feito isso deve-se pegar a média de distância para os clusters para cada valor de  $k$  e guardar esses valores para se utilizar no método de elbow para saber o valor do  $k$  Ideal.

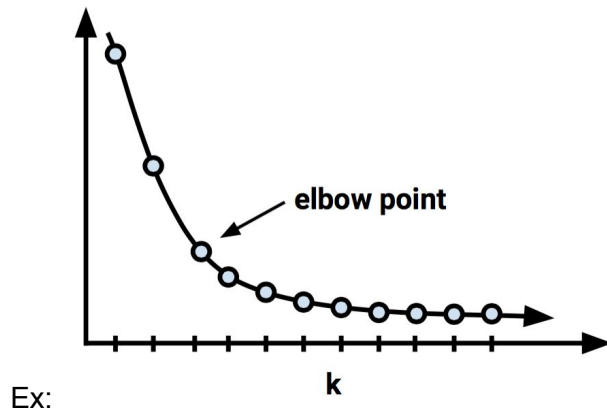
### **Método de Elbow**

O método Elbow se trata de uma técnica interessante para encontrar o valor ideal do parâmetro  $k$ .

Basicamente o que o método faz é testar a variância dos dados em relação ao número de clusters.

É considerado um valor ideal de  $k$  quando o aumento no número de clusters não representa um valor significativo de ganho.

Para esse método deve-se plotar as distâncias médias para cada valor de  $K$ . Os valores formarão algo parecido com um braço e o observando temos de ir atrás do cotovelo, esse seria o indicador do “ $K$  ideal”.



### DESCRIÇÃO DA ATIVIDADE

Neste relatório, visamos estudar o comportamento e a implementação prática da técnica de agrupamento K-means.

O trabalho considera como posição inicial dos centroides de cada cluster as K primeiras linhas do arquivo `agrup_centroides_Q1.csv` usado como referência para o estudo. O relatório visa responder a duas questões:

- 1) Determinar a quantidade de clusters ideal de acordo com o método de Elbow;
- 2) Determinar o posicionamento final dos centroides, no caso ideal.

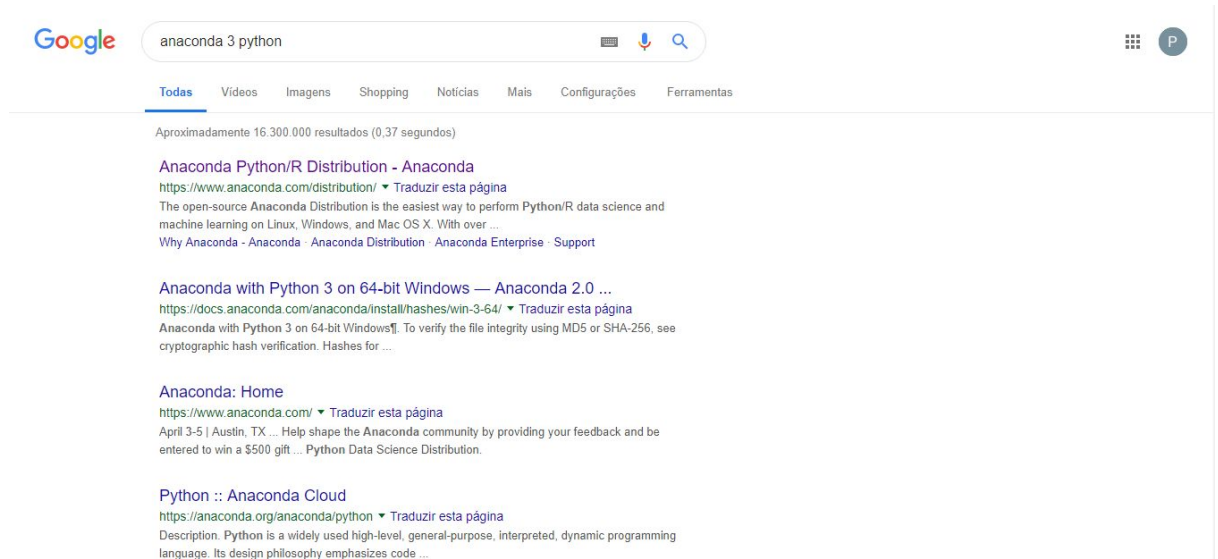
## INSTALAÇÃO:

Para essa etapa de instalação e execução será mostrada a linguagem para desenvolvimento do trabalho, nesse caso, o Python.





***Python*** é uma linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por **Guido van Rossum** em 1991. Atualmente possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos **Python Software Foundation**.

Antes de irmos para a etapa de execução do trabalho e começar testar a aplicação desenvolvida, precisamos primeiro instalar o interpretador python. Para isso, abra seu navegador e digite anaconda python na barra de busca dele.



Em seguida clique no link e você será direcionado para a página.

ProductsWhy Anaconda?SolutionsResourcesCompany[Download](#)






# Anaconda Distribution






The World's Most Popular Python/R Data Science Platform






[Download](#)

The open-source [Anaconda Distribution](#) is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 11 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas,







Clique na opção de “download”.

Desça e baixe a versão do anaconda python referente a seu sistema operacional.

## Anaconda 2018.12 for macOS Installer

### Python 3.7 version

Download

64-Bit Graphical Installer (652.7 MB)  
64-Bit Command Line Installer (557 MB)

### Python 2.7 version

Download

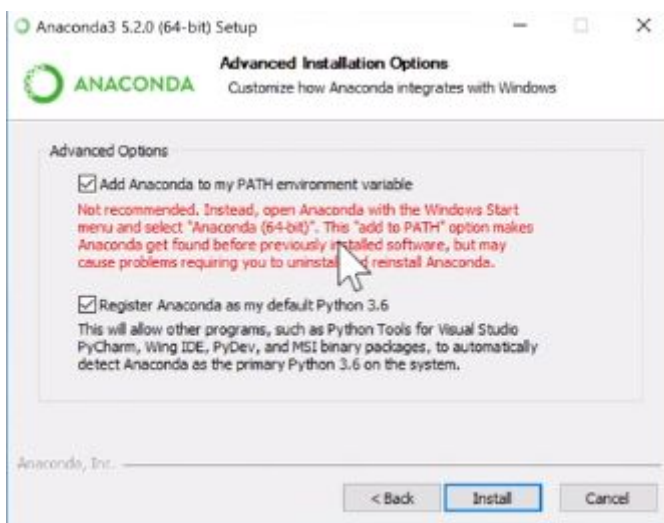
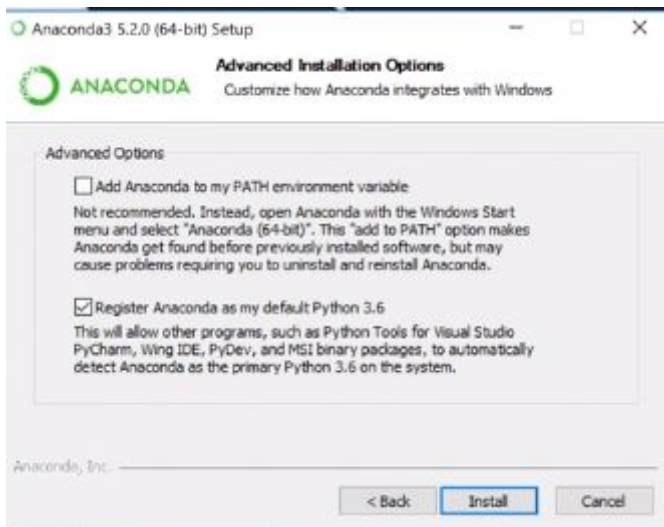
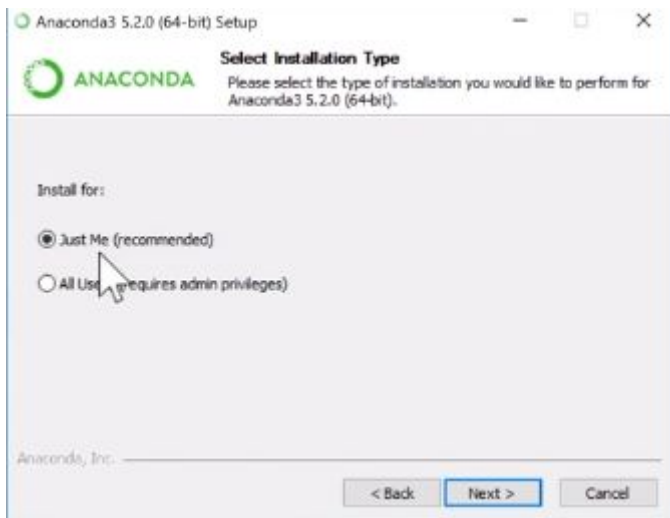
64-Bit Graphical Installer (640.7 MB)  
64-Bit Command Line Installer (547 MB)

## Get Started with Anaconda Distribution

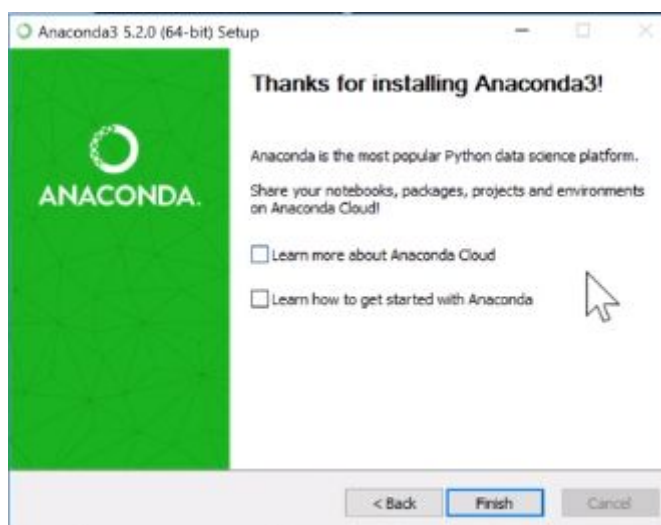
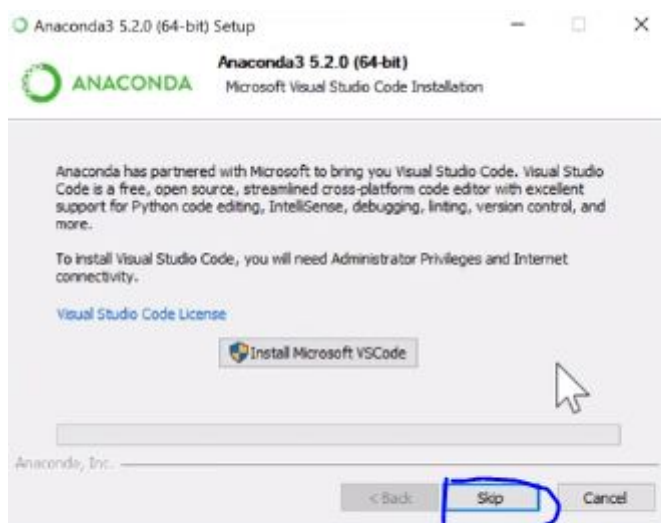
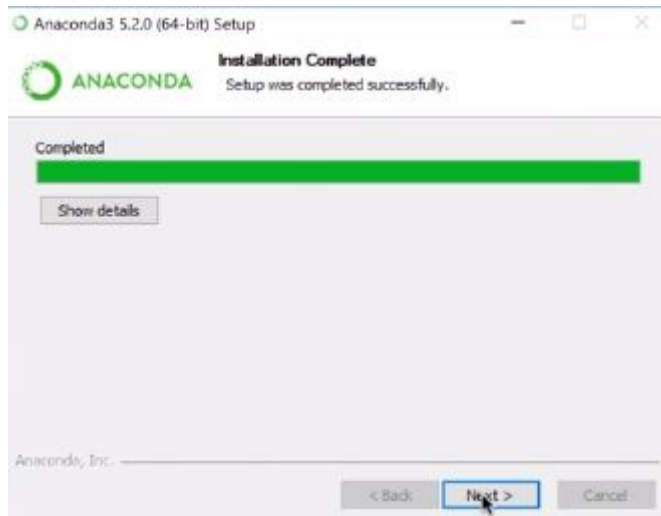
Name	Type	Size
○ Anaconda3-5.2.0-Windows-x86_64.exe	Application	646,472 KB

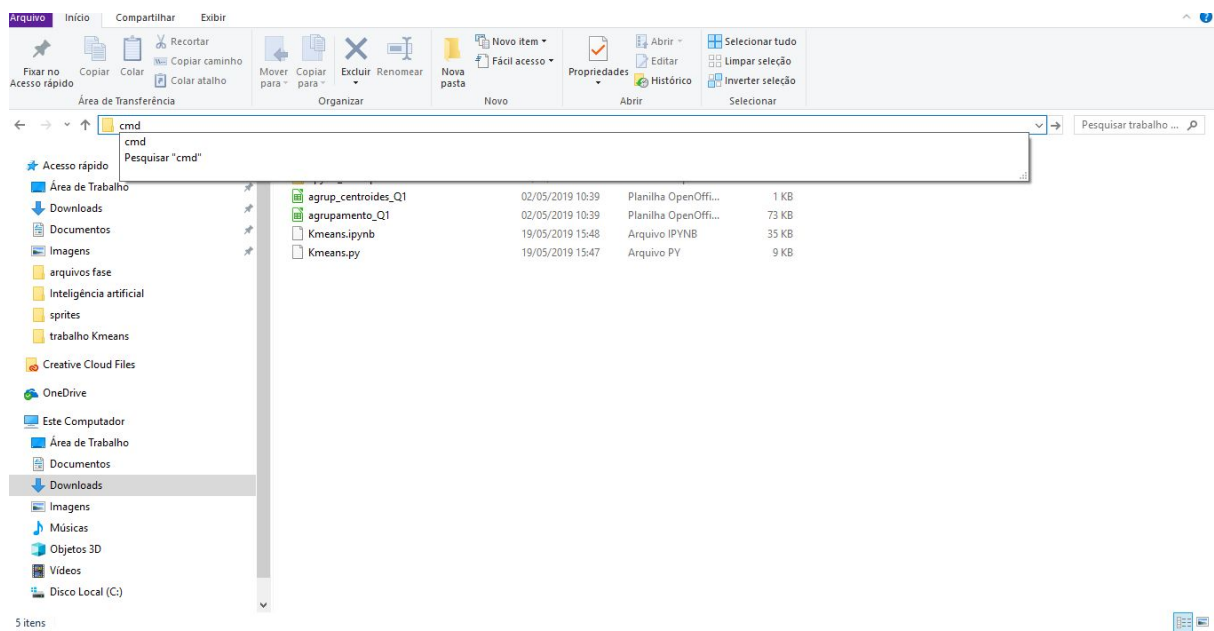
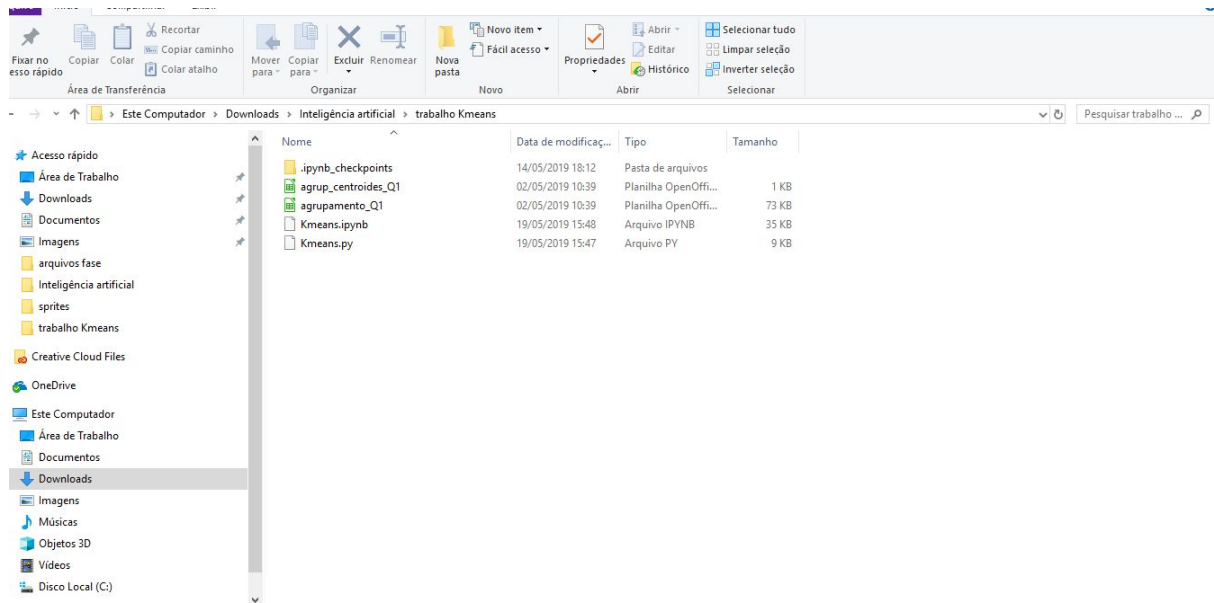
Type: Application  
Size: 631 MB  
Date modified: 2019-03-05 2:05 PM











```
C:\Windows\System32\cmd.exe
Microsoft Windows [versão 10.0.17763.503]
(c) 2018 Microsoft Corporation. Todos os direitos reservados.
C:\Users\pedru\Downloads\Inteligência artificial\trabalho Kmeans>python kmeans.py
```

```
C:\Users\pedru\Downloads\Inteligência artificial\trabalho Kmeans>jupyter notebook
[I 19:03:55.712 NotebookApp] JupyterLab extension loaded from C:\Users\pedru\Anaconda3\lib\site-packages\jupyterlab
[I 19:03:55.713 NotebookApp] JupyterLab application directory is C:\Users\pedru\Anaconda3\share\jupyter\lab
[I 19:03:55.733 NotebookApp] Serving notebooks from local directory: C:\Users\pedru\Downloads\Inteligência artificial\trabalho Kmeans
[I 19:03:55.734 NotebookApp] The Jupyter Notebook is running at:
[I 19:03:55.737 NotebookApp] http://localhost:8888/?token=6a7653aad1af751c328f7951de9a965f5c35a970809eed99
[I 19:03:55.740 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 19:03:56.168 NotebookApp]

To access the notebook, open this file in a browser:
    file:///C:/Users/pedru/AppData/Roaming/jupyter/runtime/nbserver-9724-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=6a7653aad1af751c328f7951de9a965f5c35a970809eed99
[I 19:04:52.939 NotebookApp] Kernel started: d1677a00-a993-45a5-9b15-b84c27742037
[I 19:05:08.123 NotebookApp] Adapting to protocol v5.1 for kernel d1677a00-a993-45a5-9b15-b84c27742037
[I 19:06:51.039 NotebookApp] Saving file at /Kmeans.ipynb
```

jupyter Kmeans Last Checkpoint: 3 horas atrás (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run

```

tamagr=len(agrupamento.values)

#função para calcular a distância euclidiana entre dois elementos passados
#nesse caso, ele calcula a distância entre um centroide e uma instância(elementos) passado como parâmetro

def CalcDistancia(p1,p2):
    #retorna a distância euclidiana
    #para esse calculo foi utilizado a função zip para juntar o p1 e p2 em uma única lista
    #desse forma, por exemplo se p1=[1,2,3] e p2=[4,5,6]
    #eles ficariam assim depois de usar o "zip" [[1,4],[2,5],[3,6]]
    #os elementos ficariam juntos e mais fácil trabalhar enquanto se faz a iteração

    #depois de criar uma lista com os elementos do centroide e da instancia, criamos uma lista
    #que contém as diferenças entre os pontos do centroide e instancia ao quadrado
    #isso é feito para que depois seja possível somar cada elemento da lista utilizando a função sum()
    #para que em seguida possamos utilizar a raiz quadrada e assim poderemos retornar a distância euclidiana
    #esse calculo feito abaixo representa isso:
    #math.sqrt(pow(p2x1-p1x1,2)+pow(p2x2-p1x2,2)+pow(p2x3-p1x3,2)+pow(p2x4-p1x4,2))
    return math.sqrt(sum([pow(a[1]-a[0],2) for a in list(zip(p1,p2))]))

#classe para armazenar os resultados dos Kmeans para cada K centroides
class KmeansStoreData():
    #metodo construtor dessa classe
    def __init__(self):
        self.centroids=[];#lista para armazenar os centroides finais(posições finais dos centroides)
        self.DistMedias=[];#lista para armazenar as distâncias médias
        self.Kideal=-1;

#metodo para adicionar as informações para cada K
#entre centenas de centroides finais para determinado K

```

```

k ideal : 5
Posição final dos centroides do k Ideal:
[1.0080972679625824, 0.0056562568000691, -0.006058822326704767, 0.029329271634600723]

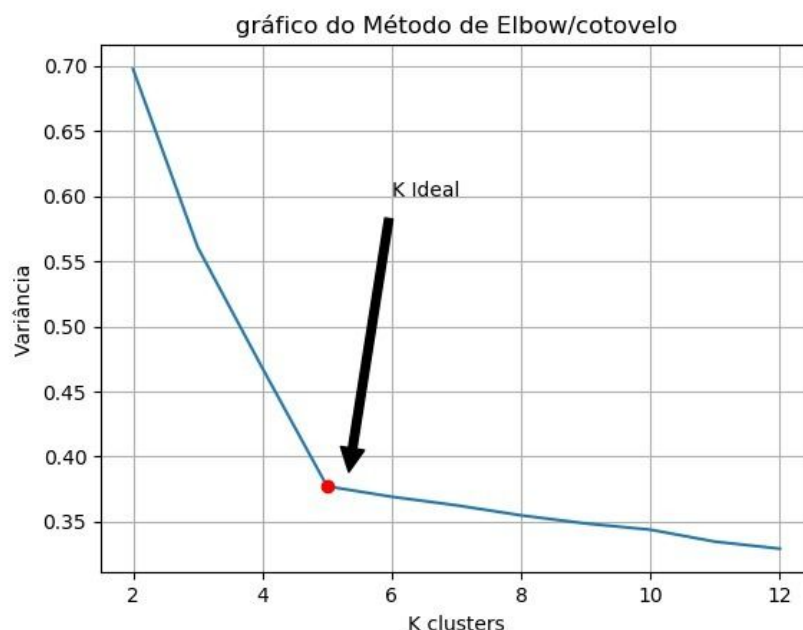
[0.9955617371432655, 0.0006261531508573007, -0.007804097531106432, 1.001521200218542]

[0.9959526972100605, 0.006462732222850456, 0.9997180082065248, 0.0031472125820982575]

[-0.034714727471357086, 0.016316290917238692, 0.010426475095138051, -0.016240750318706504]

[0.9907400881531284, 0.9930477734661887, 0.969936614894304, -0.004064822586828463]

```



O ponto vermelho representa o valor ideal de K que foi encontrado ao observar o gráfico. Como é possível ver, a partir daquele ponto, os valores, a variância pouco muda, a inclinação começa a ficar menor e por isso achamos que aquele era o K ideal.

### **ABORDAGEM UTILIZADA**

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# In[8]:
```

```
import pandas as pd
```

```
import math;
```

```
import matplotlib.pyplot as plt;
```

```
#variavel "centroides" guardarah o arquivo dos centroides.
```

```
#abrindo o arquivo csv utilizando o pandas
```

```
centroides=pd.read_csv("agrup_centroides_Q1.csv")
```

```
#aqui eh feita uma pequena "limpeza dos dados" para remover alguns elementos que nao  
serao necessarios;
```

```
#neste caso, serah removida a coluna "Unnamed: 0" que eh uma coluna dos indices do  
centroides
```

```
centroides=centroides.drop("Unnamed: 0",axis=1)
```

```
#o comando "drop", do pandas, serve para remover algum tipo de elemento, e utilizando o  
drop(nome da coluna, axis=1)
```

```
#digo para remover somente os elementos da coluna passada como parametro
```

```
#abrindo o arquivo csv que contem as instancias que serao classificadas pelo Kmeans
```

```
agrupamento=pd.read_csv("agrupamento_Q1.csv")
```

```
#criando uma variavel para armazenar o tamanho, a quantidade de linhas do arquivo que  
contem as instancias
```

```
#isso foi feito para nao precisar ficar chamando a funcao de contagem de linhas direto,  
assim, diminuindo a quantidade de processos
```

```
#desnecessarios
```

```
tamAgr=len(agrupamento.values)
```

```
#funcao para calcular a distancia euclidiana entre dois elementos passados
```

```
#nesse caso, ele calcula a distancia entre um centroide e uma instancia(elementos)  
passado como parametro
```

```
def CalcDistancia(p1,p2):
```

```
    #retorna a distancia euclidiana
```

```
    #para esse calculo foi utilizado a funcao zip para juntar centroide e instancia numa unica  
lista
```

```
#dessa forma, por exemplo se usarmos instancia=[1,2,3]e centroide[4,5,6]
#eles ficariam assim depois de usarmos o "zip":[[1,4],[2,5],[3,6]]
#os elementos ficariam juntos e mais facil de trabalhar enquanto se faz a iteracao
```

```
#depois de criar uma lista com os elementos do centroide e da instancia, criamos uma
lista
```

```
#que contem as diferencas entre os pontos do centroide e instancia ao quadrado
```

```
#isso eh feito para que depois seja possivel somar cada elemento da lista utilizando a
funcao sum()
```

```
#para que em seguida possamos utilizar a raiz quadrada e a assim podermos retornar a
distancia euclidiana
```

```
#esse calculo feito abaixo representa isso:
```

```
#math.sqrt(pow(p2x1-p1x1,2)+pow(p2x2-p1x2,2)+pow(p2x3-p1x3,2)+pow(p2x4-p1x4,2))
```

```
return math.sqrt(sum([pow(a[1]-a[0],2) for a in list(zip(p1,p2))]))
```

```
#classe para armazenar os resultados dos Kmeans para cada K centroides
```

```
class KmeansStoreData():
```

```
#metodo construtor dessa classe
```

```
def __init__(self):
```

```
    self.centroids=[];#lista para armazenar os centroides finais(posicoes finais dos
centroides)
```

```
    self.DistMedias=[];#lista para armazenar as distancias medias
```

```
    self.Kideal=-1;
```

```
#metodo para adicionar as informacoes para cada K
```

```
#centrs seriam os centroides finais para determinado K
```

```
#DistM seria a distancia media das instancias para o centroide num determinado k
```

```
def AddK(self,centrs,DistM):
```

```
    self.centroids.append(centrs);#adicionando centroides de determinado K a lista
```

```
    self.DistMedias.append(DistM);#adicionando distancia media de determinado k A lista
```

```
#funcao que calcula a nova posicao do centroide a partir das posicoes de suas instancias
```

```
def CalcCentroides(agrup):
```

```
    tamGrup=len(agrup);#pegando tamanho, quantidade de elementos proximos, que fazem
parte do centroide
```

```
    if(tamGrup>0):
```

```
        novoCentr=[0]*4;#criando uma lista de quatro elementos para servir de acumulador
```

```
        #os elementos dessa lista representaro cada elemento do centroide novo(x1,x2,x3,x4)
```

```
        #cada elemento comeca como sendo 0 para guardar a somatoria dos elementos das
instancias
```

```
        #para que depois possa usar esse somatoria para calcular a media e assim conseguir
o novo centroide
```

```
        #loop que vai iterar entre as instancias do centroide e o segundo loop para para
percorrer(x1,x2,x3,x4) e somar
```

```
        #esse valores das instancias ao do novo centroide
```

```

for a in agrup:
    for ind in range(0,4):
        novoCentr[ind]+=a[ind];

return [a/tamGrup for a in novoCentr];
    #retornando uma lista com a divisao de cada x acumulado pela quantidade de
instancias do centroide
    #pega a media de x1,x2,x3,x4 acumulado (pega a media dos valores) e retornar numa
lista contendo x1,x2,x3,x4 do novo centroide

return []:#precaucao para caso nao tiver elementos no centroide, ele retornar uma lista
vazia

#funcao do metodo de elbow
def Elbow():
    DataKmeans=KmeansStoreData();#criando um objeto para guardar as informacoes dos
centroides para cada valor de k
    #for que vai de 2 a 12. esse for vai atribuir o valor de k na chamada de funcao do kmeans
que por sua vez vai retornar
    #as informacoes referentes a aquele valor de k
    for k in range(2,13):
        ValoresParaK= KMeans(k);#guardando o valor para k centroides numa variavel aux
        DataKmeans.AddK(ValoresParaK[0],ValoresParaK[1]);
        #armazenando as informacoes de k no objeto que guarda todas as informacoes do
metodo de elbows
        #printando as medias
        for indice in range(0,len(DataKmeans.DistMedias)):
            print("Media/variancia para %r" %r
clusters:%r\n\n"%((indice+2),DataKmeans.DistMedias[indice]));

        #printando centroides do k Ideal
        print("k ideal : 5");
        print("Posicao final dos centroides do k Ideal:");
        for x in DataKmeans.centroids[3]:
            print("%r\n\n"%(x));

        print(DataKmeans.centroids[3]);

#plotando grafico de elbows
plt.figure();
plt.title("gráfico do Metodo de Elbow/cotovelo");
plt.grid();
plt.plot(list(range(2,13)),DataKmeans.DistMedias);

```

```

plt.plot(5.3,DataKmeans.DistMedias[3],'ro');
plt.xlabel("K clusters");
plt.ylabel("Variância");

plt.annotate("K Ideal",
xy=(5,DataKmeans.DistMedias[3]),xytext=(6,0.6),arrowprops=dict(facecolor='black',shrink=0.05));
plt.show();

```

```

def KMeans(k):

```

```

    centros=[list(a) for a in centroides.values[:k]];#k clusters. se inicia com os k clusters do
arquivo centroide
    somaDistAux=0;
    #print(centros)
    listCentTrein=[-1]*tamAgr;#lista que guardará o id do cluster de cada instancia

```

```

    print("--Executando Kmeans para %r clusters--"%(k));

```

```

    cont=0;#contador de etapas para estabilizar. Vai contar quantas loops foram necessários
ateh estabilizar

```

```

    while True:

```

```

        cont+=1;#incrementando contador

```

```

        #lista para guardar as classificações atuais

```

```

        listAuxC=[];

```

```

        somaDistAux=0;#variável que guarda as somas das distâncias

```

```

        agrupamC=[] for x in range(0,k);

```

```

        #lista que guardará as informações de cada centroid para depois poder calcular
novos centroides

```

```

        #loop para percorrer todas as instancias

```

```

        for a in range(0,tamAgr):

```

```

            distancias=[];#lista de distancias de tal instancias para cada cluster/centroide

```

```

            #loop que percorre todos os centroides para que assim possa ser feito o calculo da
distancia para todos os centroides

```

```

            for cent in centros:

```

```

                distancias.append(CalcDistancia(agrupamento.values[a],cent));

```

```

            #fim do loop dos centroides

```

```

            idCentro=distancias.index(min(distancias));#pegando o id do centroide com menor
distancia

```



```

        listAuxC.append(idCentro);#adicionando a lista, na posicao da instancia, o numero
do cluster
        #listCentTrein[a]=idCentro
        somaDistAux+=(distancias[idCentro]);#incrementando a soma de distancias com a
dist para o cluster
        agrupamC[idCentro].append(agrupamento.values[a]);

nCentroids=[CalcCentroides(agrupamC[x]) for x in range(0,k)]
#print([listCentTrein.count(y) for y in range(0,k)])

#print("\n\n")
if(listAuxC==listCentTrein):
    #condicao para parar o loop. quando a classe(cluster) de cada instancia nao muda
mais
    break;

for a in range(0,k):
    if(len(nCentroids[a])>0):
        centros[a]=nCentroids[a];#adicionando os novos centroides na lista de centroides

listCentTrein=listAuxC;
print("Execucao com %r clusters estabilizada na rodada %r\n\n"%(k,cont));

return [centros,somaDistAux/tamAgr]

Elbow();

```

# In[ ]:

# In[ ]:

## **CONCLUSÃO**

Através do exposto neste relatório, foi possível elucidar o funcionamento do método de agrupamento K-means, estudando os motivos que o fazem ser um dos principais métodos utilizado em classificação não supervisionada;

Uma boa prática para o K-means é que para agilizar certos processos e reduzir um pouco de tempo que levaria para terminar de executar o K-means, é interessante fazer a somatória das distâncias que serão usadas no método de elbows logo enquanto faz o cálculo das distâncias para os centróides. Dessa forma, evita-se ter de recalcular valores.

Uma coisa que também é importante no K-means é o critério de parada. Caso o critério de parada não for escolhido/construído corretamente, pode ser que o programa fique dando loops infinitos.