Universidade do Estado de Santa Catarina (UDESC)
Joinville, Santa Catarina, Brasil

Aluno: Leandro Ribeiro Rittes

1-) No conjunto de dados "Dados-Tarefa-02.csv" aplique os métodos (a) k-Means, (b) k-Medoids, (c) DBSCAN e (d) BIRCH.

1) Importação de bibliotecas e pré processamento dos dados

```
import pandas as pd
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
import numpy as np
from matplotlib import pyplot as plt
```

```
from yellowbrick.cluster import KElbowVisualizer
```

```
from yellowbrick.cluster import SilhouetteVisualizer
```

```
from sklearn_extra.cluster import KMedoids
import numpy as np
```

```
from sklearn.cluster import DBSCAN
```

```
from sklearn.cluster import Birch
```

Construindo o DataFrame com o arquivo "Dados-Tarefa-02.csv"

```
df = pd.read_csv('./Dados-Tarefa-02.csv')
```

	Index	d1	d2
0	0	1.225160	-0.951731
1	1	1.016304	-1.725175
2	2	0.335340	-1.724896
3	3	1.786348	-1.782653
4	4	1.016751	1.062569
...	...	...	...
995	995	0.929594	-0.743331
996	996	-0.338431	-0.343315
997	997	1.542708	-0.055665
998	998	0.816646	-1.250919
999	999	1.137823	-1.261520

1000 rows × 3 columns

1) a) K-Means

Dando um valor aleatório para ser a quantidade de cluster final e criando um array com os valores de ‘d1’ e ‘d2’ , no formato [row1[d1,d2],row2[d1,d2], ...]

```
k = 2
X = np.array(df[['d1','d2']].values.tolist())
```

```
array([[ 1.22515974, -0.95173116],
       [ 1.01630365, -1.72517506],
       [ 0.33534004, -1.7248955 ],
       ...,
       [ 1.54270795, -0.05566478],
       [ 0.81664566, -1.2509187 ],
       [ 1.13782254, -1.26151957]])
```

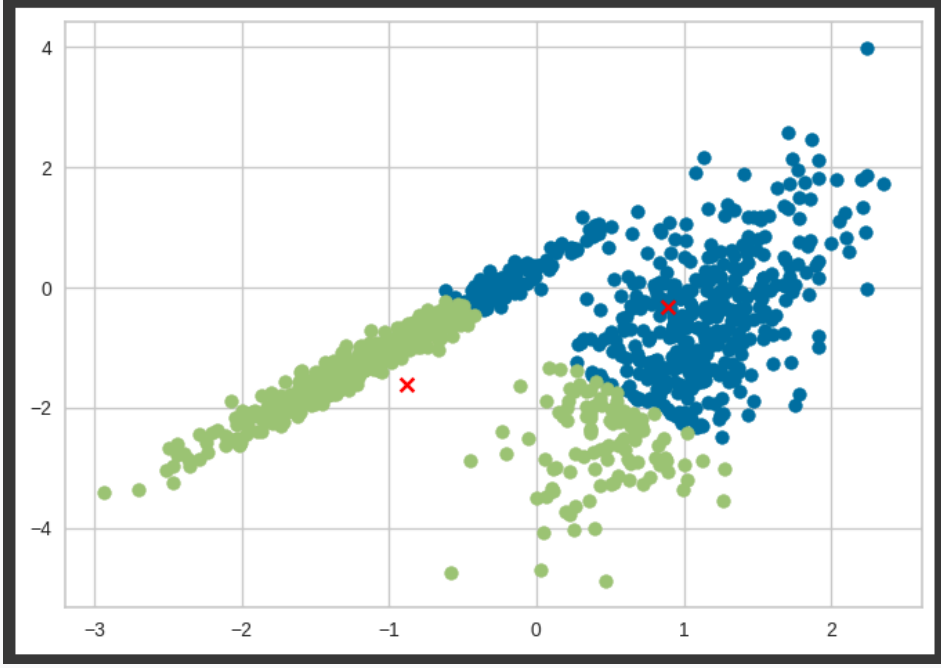
Criando a instância de Kmeans e passando os valores de ‘X’ para ser resolvido

```
kmeans = KMeans(n_clusters=k,random_state=0,n_init='auto')
kmeans.fit(X)
```

```
KMeans
KMeans(n_clusters=2, n_init='auto', random_state=0)
```

Separando cada ponto em seu respectivo cluster, encontrando o centro de cada cluster e plotando o gráfico

```
for idx, class_value in enumerate(range(k)):
  row_ids = np.where(kmeans.labels_ == class_value)
  sc = plt.scatter(X[row_ids, 0], X[row_ids, 1])
  plt.scatter(kmeans.cluster_centers_[idx][0],kmeans.cluster_centers_[idx][1], color='red', marker='x')

plt.show()
```



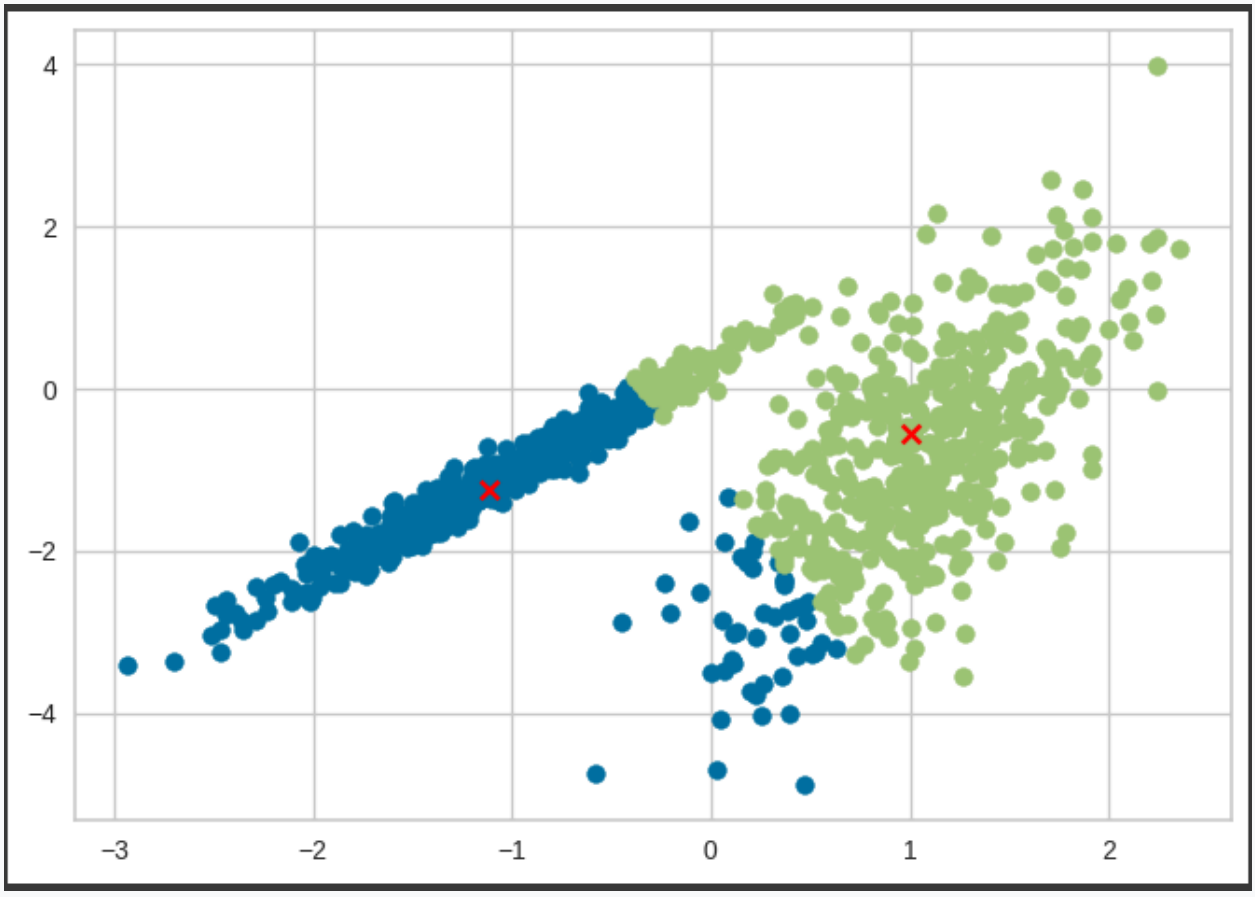
1) b) K-Medoids

Dando um valor aleatório para ser a quantidade de cluster final, criando a instância de KMedoids e passando os valores de ‘X’ para ser resolvido

```
k = 2
kmetroids = KMedoids(n_clusters=k,random_state=0)
kmetroids.fit(X)
```

```
KMedoids
KMedoids(n_clusters=2, random_state=0)
```

Separando cada ponto em seu respectivo cluster, encontrando o centro de cada cluster e plotando o gráfico

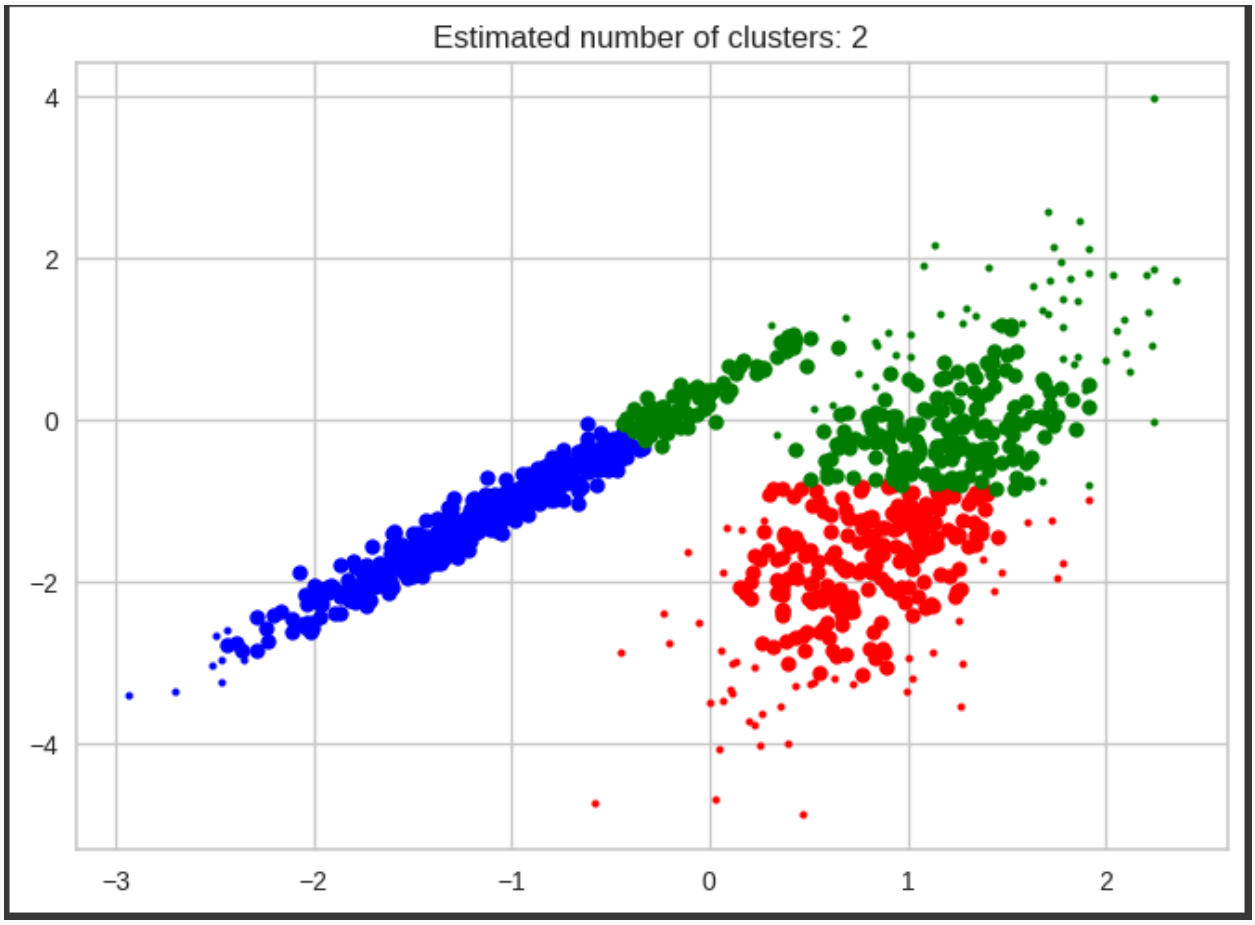


1) c) DBSCAN

criando a instância de DBSCAN e passando os valores de ‘X’ para ser resolvido, encontrando as quantidades de clusters e os pontos q são ‘ruidos’ e as labels

```
dbscan = DBSCAN(eps=0.3, min_samples=10).fit(X)
labels = dbscan.labels_

n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)
labels = dbscan.labels_
```

visualiza os resultados do algoritmo DBSCAN, destacando os pontos principais e o ruído em um gráfico de dispersão com cores diferentes para cada cluster

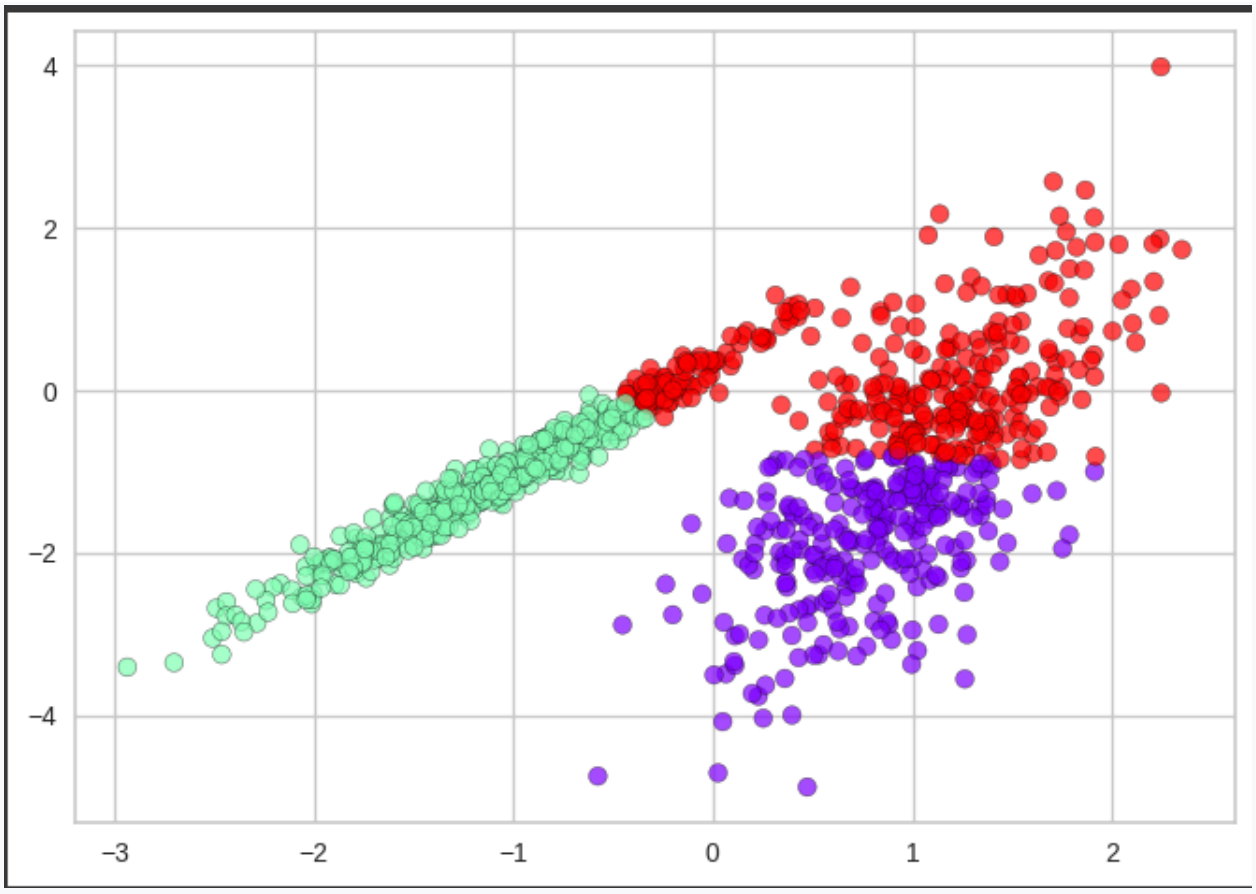


1) d) BIRCH

criando a instância de BIRCH e passando os valores de ‘X’ para ser resolvido,Usa o modelo ajustado para predizer os rótulos dos clusters para os dados X

```
plt.scatter(X[:,0], X[:,1], c=labels, cmap='rainbow', alpha=0.7, edgecolors='k')
plt.show()
```

Usa `plt.scatter` para criar um gráfico de dispersão dos dados X.As cores dos pontos são determinadas pelos rótulos dos clusters (`labels`), com a colormap `'rainbow'`.

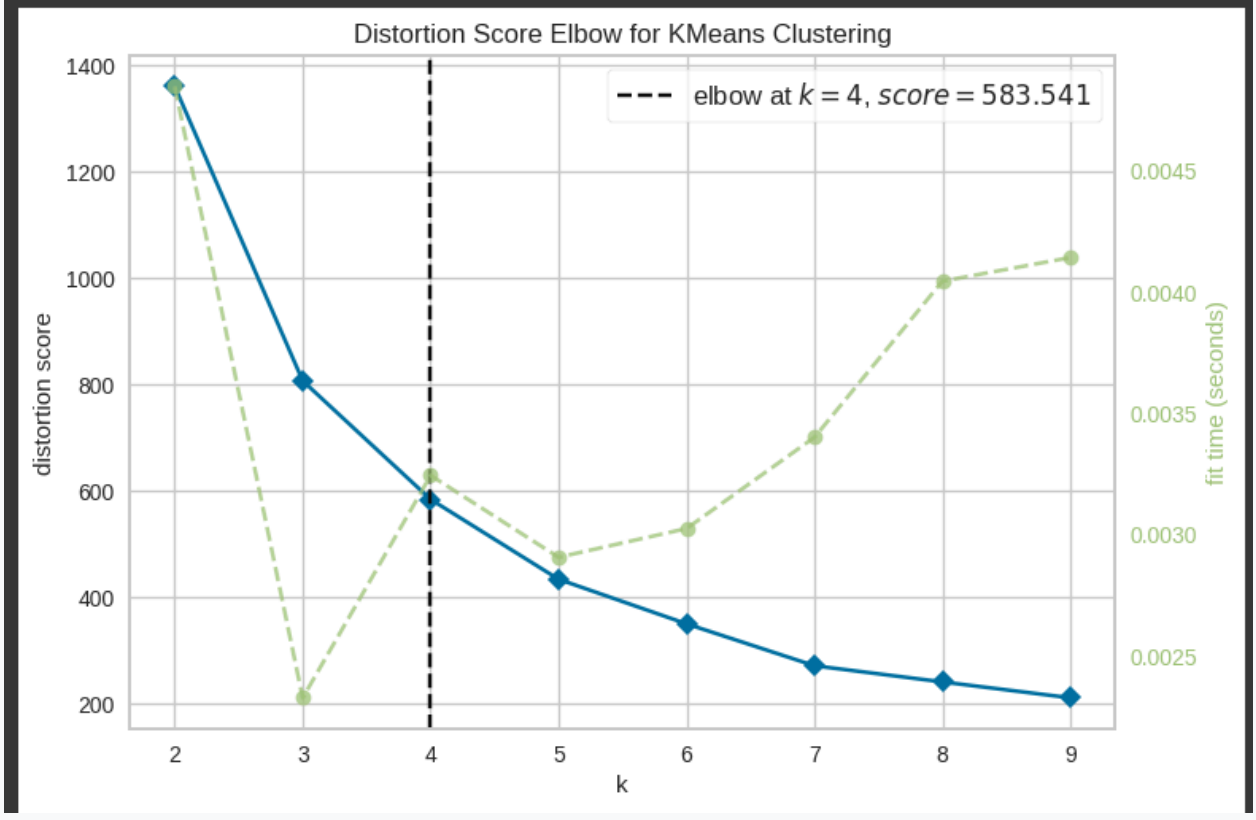


2-) Aplicar Elbow e Silhouette para encontrar o valor de k para os algoritmos k-Means (a) e k-Medoids (b). Qual o melhor valor de k (i.e., número de clusters) para estes casos?

2) a) Elbow para K-Means

Cria um objeto `KElbowVisualizer` para o algoritmo K-means, especificando que o número de clusters a ser avaliado está no intervalo de 2 a 10. Ajusta o visualizador aos dados `X` para calcular a pontuação de distorção para cada número de clusters. Exibe o gráfico do método do cotovelo, que ajuda a identificar o número ideal de clusters.

```
visualizer = KElbowVisualizer(kmeans, k=(2,10))
visualizer.fit(X)
visualizer.show()
```

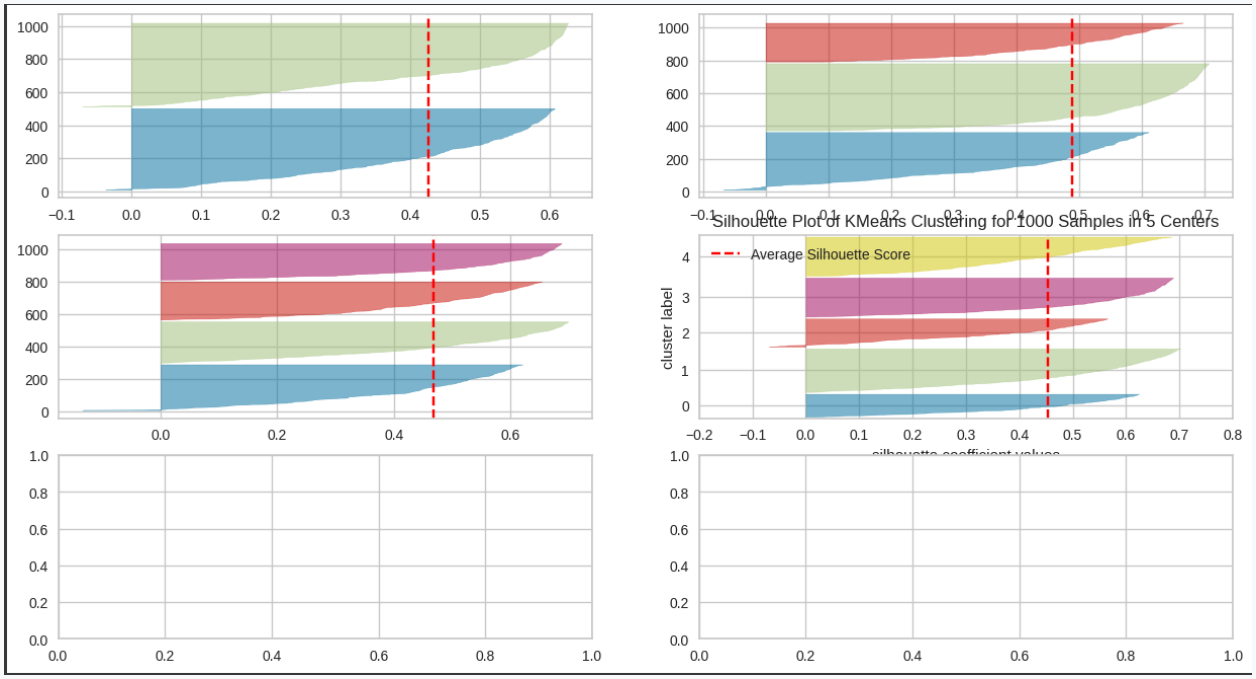


Com base no gráfico, o melhor valor para o "k" (número de clusters) é 4

Silhouette Score para K-Means

O código cria uma figura com subplots e para cada valor de 2 a 5 clusters, ajusta um modelo KMeans aos dados X, visualizando o diagrama de silhueta correspondente em cada subplot. No final, exibe todos os gráficos de silhueta, ajudando a avaliar a qualidade dos clusters para diferentes valores de k.

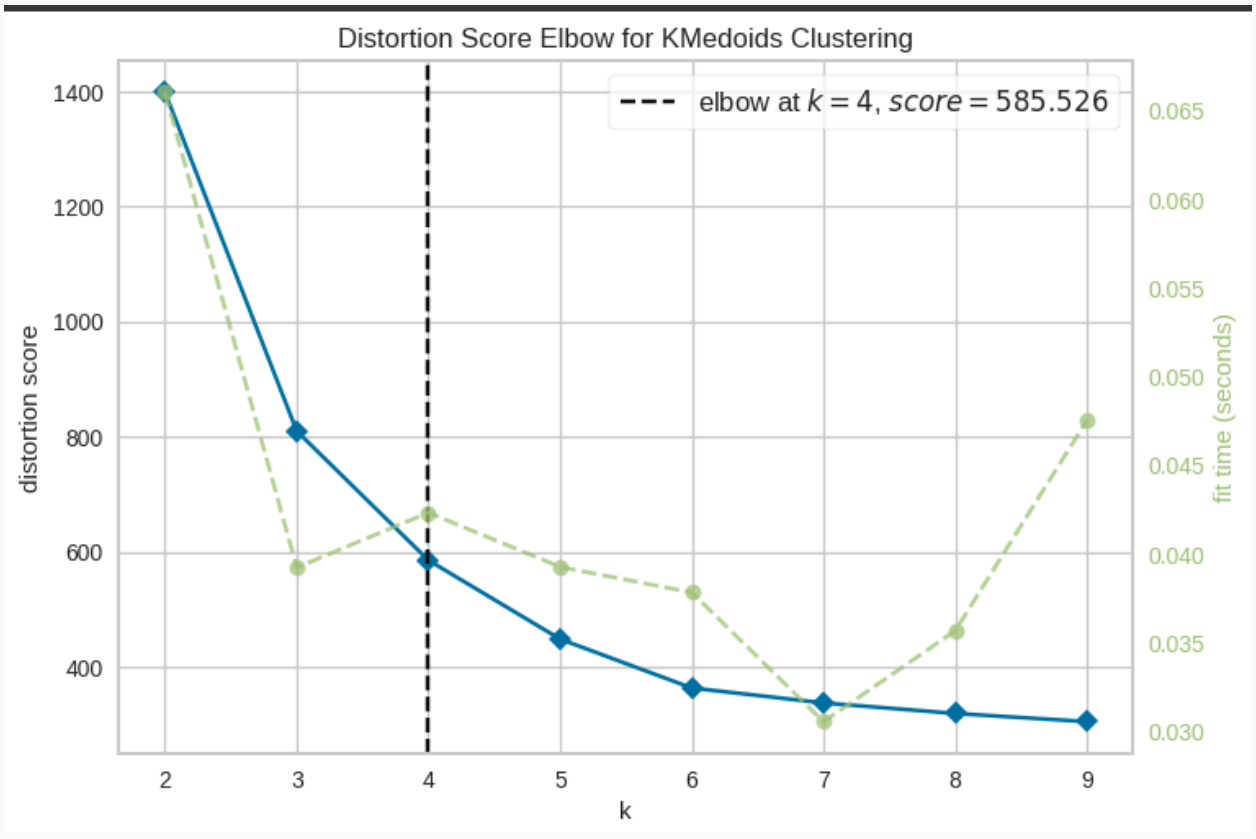
```
fig, ax = plt.subplots(3, 2, figsize=(15,8))
for i in [2, 3, 4, 5]:
  kmeans = KMeans(n_clusters=i, init='k-means++', n_init=10,max_iter=100, random_state=42)
  q, mod = divmod(i, 2)
  visualizer = SilhouetteVisualizer(kmeans, colors='yellowbrick',ax=ax[q-1][mod])
  visualizer.fit(X)

visualizer.show()
```



Com base no gráfico, o melhor valor para o "k" (número de clusters) pode tanto ser 2, 3 e 4, e isso se dá pelo fato de que esses valores tiveram os valores de cada cluster maior ou igual a média do dataset e teve tamanhos semelhantes de um cluster para outro, já o k = 5 não teve essas condições

2) b) Elbow para K-Medoids

Cria um objeto `KElbowVisualizer` para o algoritmo K-means, especificando que o número de clusters a ser avaliado está no intervalo de 2 a 10. Ajusta o visualizador aos dados X para calcular a pontuação de distorção para cada número de clusters. Exibe o gráfico do método do cotovelo, que ajuda a identificar o número ideal de clusters.

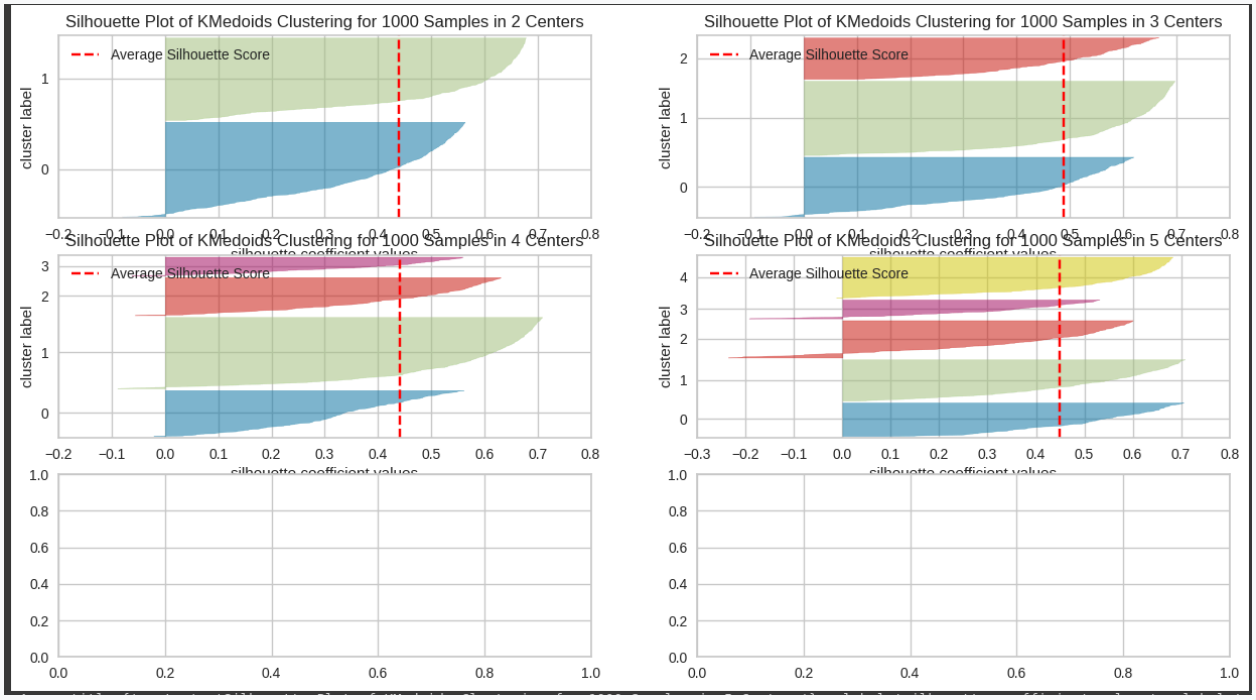


Com base no gráfico, o melhor valor para o "k" (número de clusters) é 4

Silhouette Score para K-Medoids

O código cria uma figura com subplots e para cada valor de 2 a 5 clusters, ajusta um modelo KMeans aos dados X, visualizando o diagrama de silhueta correspondente em cada subplot. No final, exibe todos os gráficos de silhueta, ajudando a avaliar a qualidade dos clusters para diferentes valores de k.

```
fig, ax = plt.subplots(3, 2, figsize=(15,8))
for i in [2, 3, 4, 5]:
  kmetroids = KMedoids(n_clusters=i, init='k-medoids++',random_state=42)
  q, mod = divmod(i, 2)
  visualizer = SilhouetteVisualizer(kmetroids, colors='yellowbrick',ax=ax[q-1][mod])
  visualizer.fit(X)
  visualizer.finalize()

visualizer.show()
```



Com base no gráfico, o melhor valor para o "k" (número de clusters) é 3, e isso se dá pelo fato de q k = 4 e k = 5 não tiveram os valores de cada cluster maior ou igual a média do dataset (linha vermelha pontilhada), e k = 2 teve tamanhos muito diferente de um cluster para outro