Kmeans_2

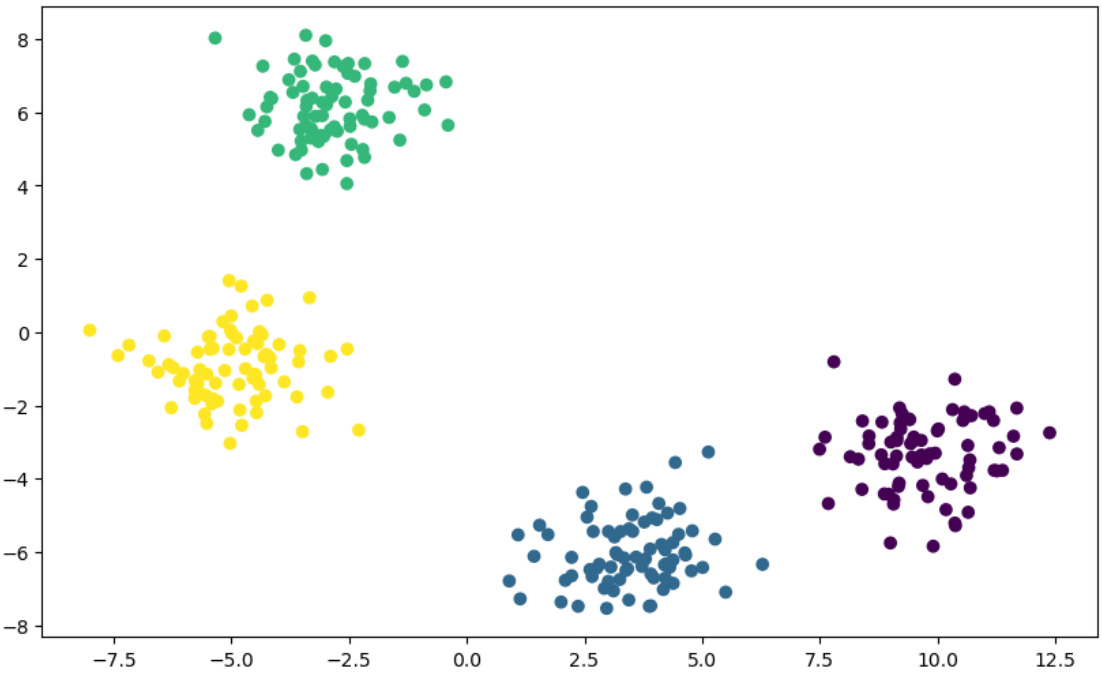
November 25, 2025

[1]:
```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

[2]:
```
Caracteristicas, Labels = make_blobs(
    n_samples = 300,
    centers= 4,
    cluster_std=1.0,
    random_state=56
)
```

[3]:
```
plt.figure(figsize=(10,6))
plt.scatter(Caracteristicas[:,0],Caracteristicas[:,1],c=Labels)
```

[3]:
```
<matplotlib.collections.PathCollection at 0x7fe118a25650>
```



[4]:
```
scalar = StandardScaler()
```

[5]:
```
scalar_caracteristicas=scalar.fit_transform(Caracteristicas)
```

[6]:
```
scalar_caracteristicas[:10]
```

[6]:
```
array([[ 1.41802166, -0.5018383 ],
       [ 1.57293969, -0.2467037 ],
       [ 1.41248855, -0.51139454],
       [ 1.22780771, -0.3893276 ],
       [ 1.39174457, -0.41603987],
       [-0.85124499,  1.27648513],
       [-1.11489448,  0.28730228],
       [-1.19826263, -0.14078788],
       [ 0.22043578, -0.80607113],
       [ 1.53899806, -0.90507224]])
```

[7]:
```
 Caracteristicas[:10]
```

[7]:
```
array([[ 9.66316603, -3.3535356 ],
       [10.57114476, -2.17710796],
       [ 9.63073631, -3.39759948],
       [ 8.54831693, -2.8347479 ],
       [ 9.50915526, -2.95791837],
       [-3.63706692,  4.84632763],
       [-5.18232366,  0.28519781],
       [-5.67094664, -1.68872931],
       [ 2.64408295, -4.75635554],
       [10.37221197, -5.21285044]])
```

[8]:
```
kmeans=KMeans(
    init="random",
    n_clusters=4,
    n_init=20,
    max_iter=800,
    random_state=56
)
```

[9]:
```
kmeans.fit(scalar_caracteristicas)
```

[9]:
```
KMeans(init='random', max_iter=800, n_clusters=4, n_init=20, random_state=56)
```

K-Means: Inércia A inércia mede o quão bem um conjunto de dados foi agrupado pelo K-Means. Ele é calculado medindo a distância entre cada ponto de dados e seu centroide, elevando essa distância ao quadrado e somando esses quadrados em um cluster.

Um bom modelo é aquele com baixa inércia E baixo número de clusters (K). No entanto, isso é uma compensação porque, à medida que Kaumenta, a inércia diminui.

Para encontrar o ideal K para um conjunto de dados, use o método Elbow ; encontre o ponto em que a diminuição da inércia começa a diminuir. K=5 é o "cotovelo" deste gráfico.

[10]:
```
kmeans.inertia_
```

[10]:
```
21.54100429325745
```

[11]:
```
kmeans.cluster_centers_
```

[11]:
```
array([[ 1.43382985, -0.49686105],
       [-0.71995304,  1.55603539],
       [ 0.36657176, -1.08085844],
       [-1.08044857,  0.0216841 ]])
```

[12]:
```
kmeans.n_iter_
```

[12]:
```
4
```

[13]:
```
kmeans.labels_[:10]
```

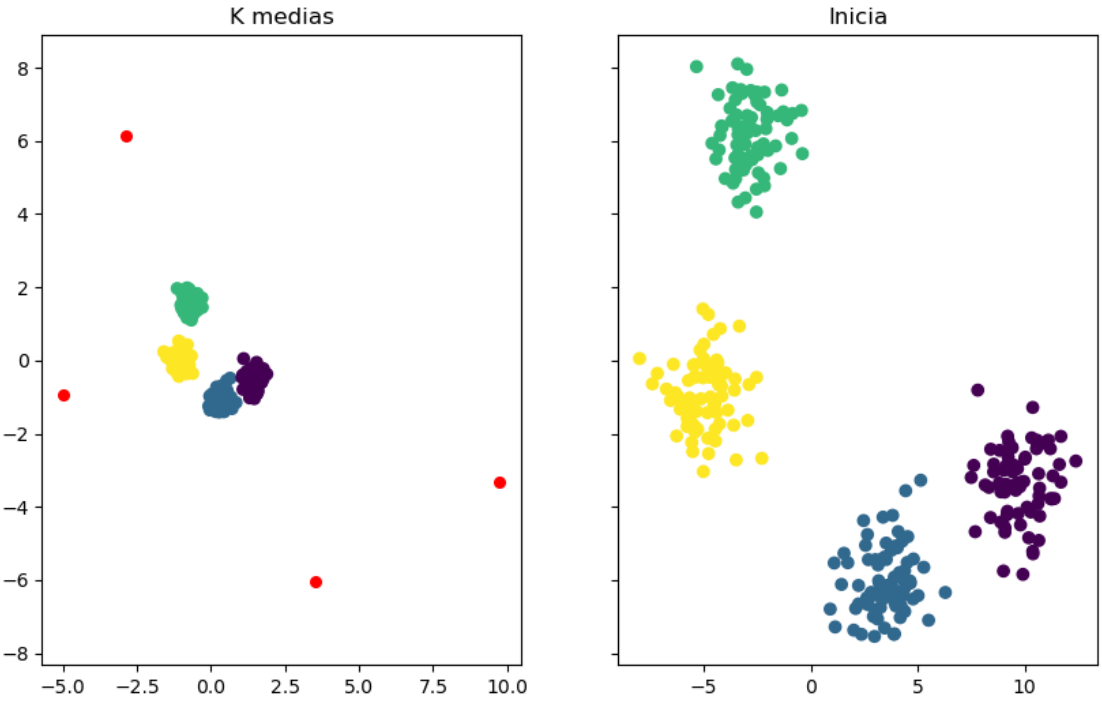
[13]:
```
array([0, 0, 0, 0, 0, 1, 3, 3, 2, 0], dtype=int32)
```

[14]:
```
Labels[:10]
```

[14]:
```
array([0, 0, 0, 0, 0, 2, 3, 3, 1, 0])
```

[17]:
```
f,(eixo1, eixo2)=plt.subplots(1, 2, sharey=True, figsize=(10,6))
eixo1.set_title('K medias')
eixo1.scatter(scalar_caracteristicas[:,0],scalar_caracteristicas[:,1],c=Labels)
eixo1.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],
              s=30, c='red',label='Centroides'
              )
eixo2.set_title('Inicia')
eixo2.scatter(Caracteristicas[:,0],Caracteristicas[:,1],c=Labels)
```

[17]:
```
<matplotlib.collections.PathCollection at 0x7f5ed61931d0>
```

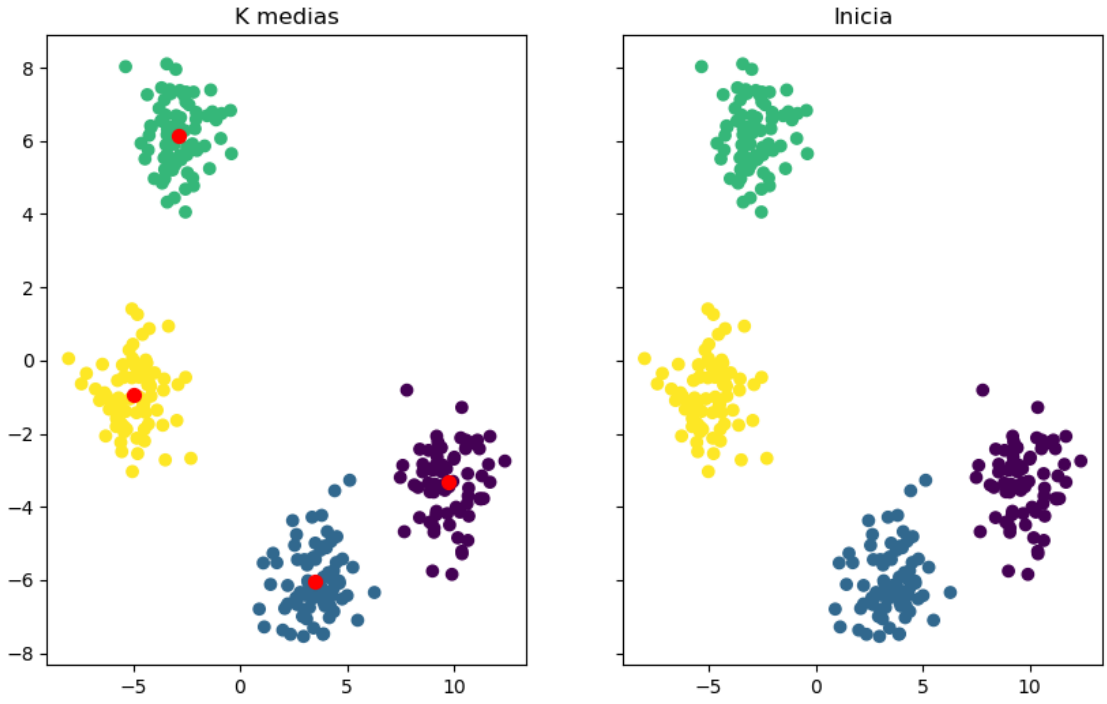


[18]:
```
kmeans.fit(Caracteristicas)
```

[18]:
```
KMeans(init='random', max_iter=800, n_clusters=4, n_init=20, random_state=56)
```

[19]:
```
f,(eixo1, eixo2)=plt.subplots(1, 2, sharey=True, figsize=(10,6))
eixo1.set_title('K medias')
eixo1.scatter(Caracteristicas[:,0],Caracteristicas[:,1],c=Labels)
eixo1.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],
              s=50, c='red',label='Centroides'
              )from sklearn.metrics import confusion_matrix,␣
 ↪classification_report
eixo2.set_title('Inicia')
eixo2.scatter(Caracteristicas[:,0],Caracteristicas[:,1],c=Labels)
```

[19]:
```
<matplotlib.collections.PathCollection at 0x7f5ed6671dd0>
```



[20]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

[21]:
```
print('Matriz de confusão')
print(confusion_matrix(Labels, kmeans.labels_))
```

```
Matriz de confusão
[[75  0  0  0]
 [ 0  0 75  0]
 [ 0 75  0  0]
 [ 0  0  0 75]]
```

[22]:
```
print('RELATÓRIO DE CLASSIFICAÇÃO')
print(classification_report(Labels, kmeans.labels_))
```

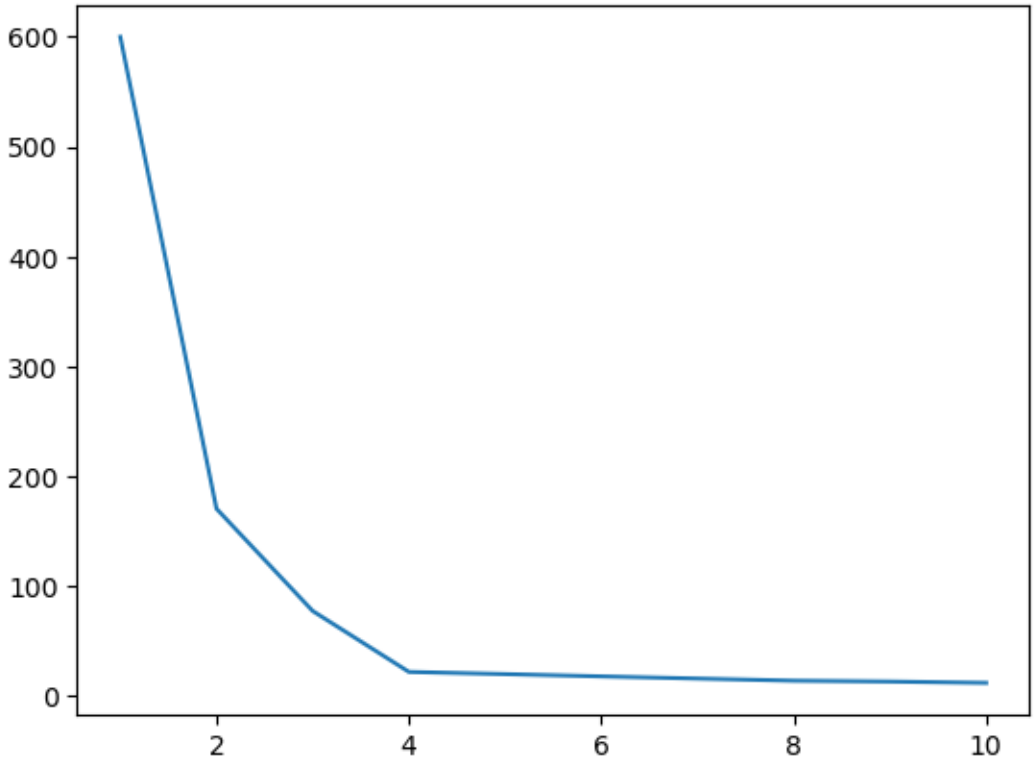
```
RELATÓRIO DE CLASSIFICAÇÃO
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        75
           1       0.00      0.00      0.00        75
           2       0.00      0.00      0.00        75
           3       1.00      1.00      1.00        75

    accuracy                           0.50       300
   macro avg       0.50      0.50      0.50       300
weighted avg       0.50      0.50      0.50       300
```

[23]:
```
kmeans_valores={
    'init':'random',
    'n_init':10,
    'max_iter':300,
    'random_state':56,
}
```

[24]:
```
SER=[]
for k in range(1,11):
    kmeansCT=KMeans(n_clusters=k, **kmeans_valores)
    kmeansCT.fit(scalar_caracteristicas)
    SER.append(kmeansCT.inertia_)
```

[25]:
```
plt.plot(range(1,11),SER)
```

[25]:
```
[<matplotlib.lines.Line2D at 0x7f5ed6060f50>]
```

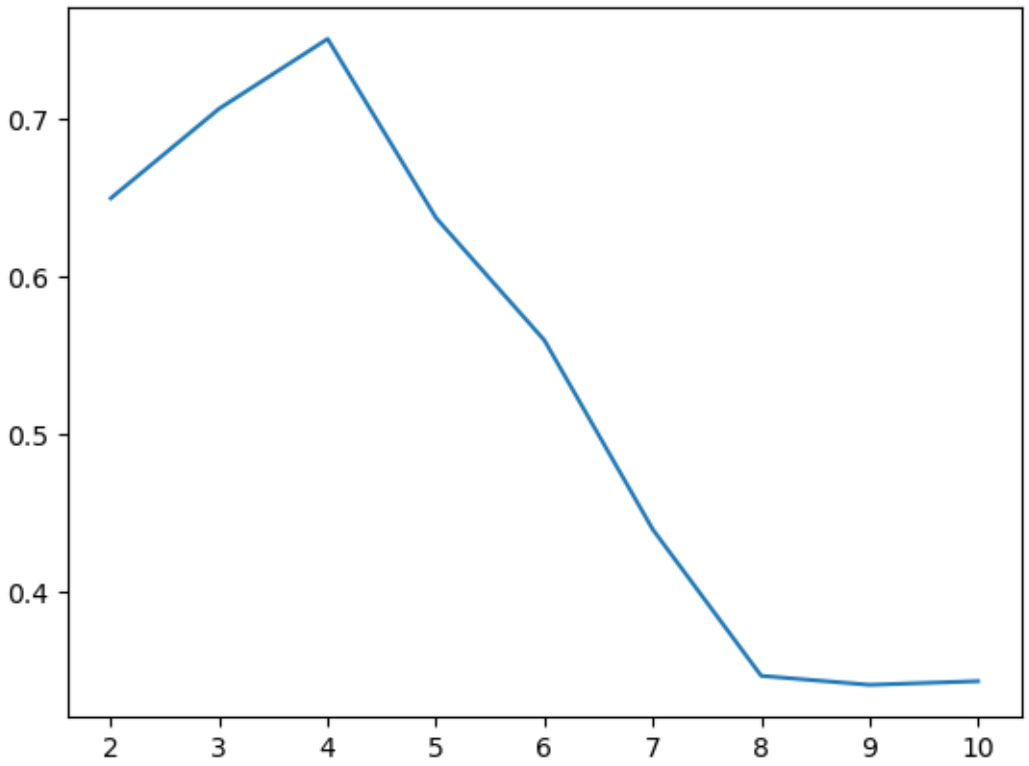


[26]:
```
coeficiente_silueta=[]
for k in range(2,11):
    kmeans_S=KMeans(n_clusters=k, **kmeans_valores)
```

```
    kmeans_S.fit(scalar_caracteristicas)
    score=silhouette_score(scalar_caracteristicas,kmeans_S.labels_)
    coeficiente_silueta.append(score)
```

[27]:
```
plt.plot(range(2,11), coeficiente_silueta)
```

[27]:
```
[<matplotlib.lines.Line2D at 0x7f5ecc3e2750>]
```

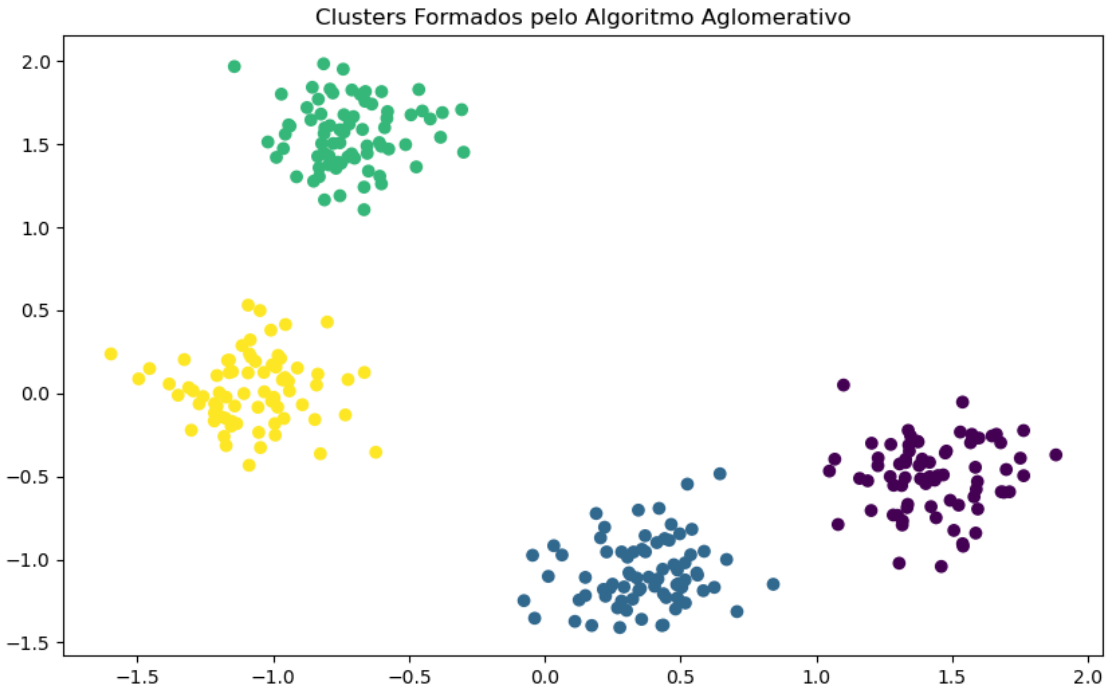


[28]:
```
from sklearn.cluster import AgglomerativeClustering

from sklearn.cluster import kmeans_plusplus
```

[29]:
```
clustering = AgglomerativeClustering().fit(Caracteristicas)
```

[30]:
```
# Aplicando o algoritmo de cluster aglomerativo
agg_clustering = AgglomerativeClustering(n_clusters=4)
agg_labels = agg_clustering.fit_predict(scalar_caracteristicas)

# Visualizando os clusters formados
plt.figure(figsize=(10, 6))
plt.scatter(scalar_caracteristicas[:, 0], scalar_caracteristicas[:, 1],␣
 ↪c=agg_labels)
```

```
plt.title('Clusters Formados pelo Algoritmo Aglomerativo')
plt.show()
```



[31]:
```
print('Matriz de confusão')
print(confusion_matrix(Labels, agg_labels))
```

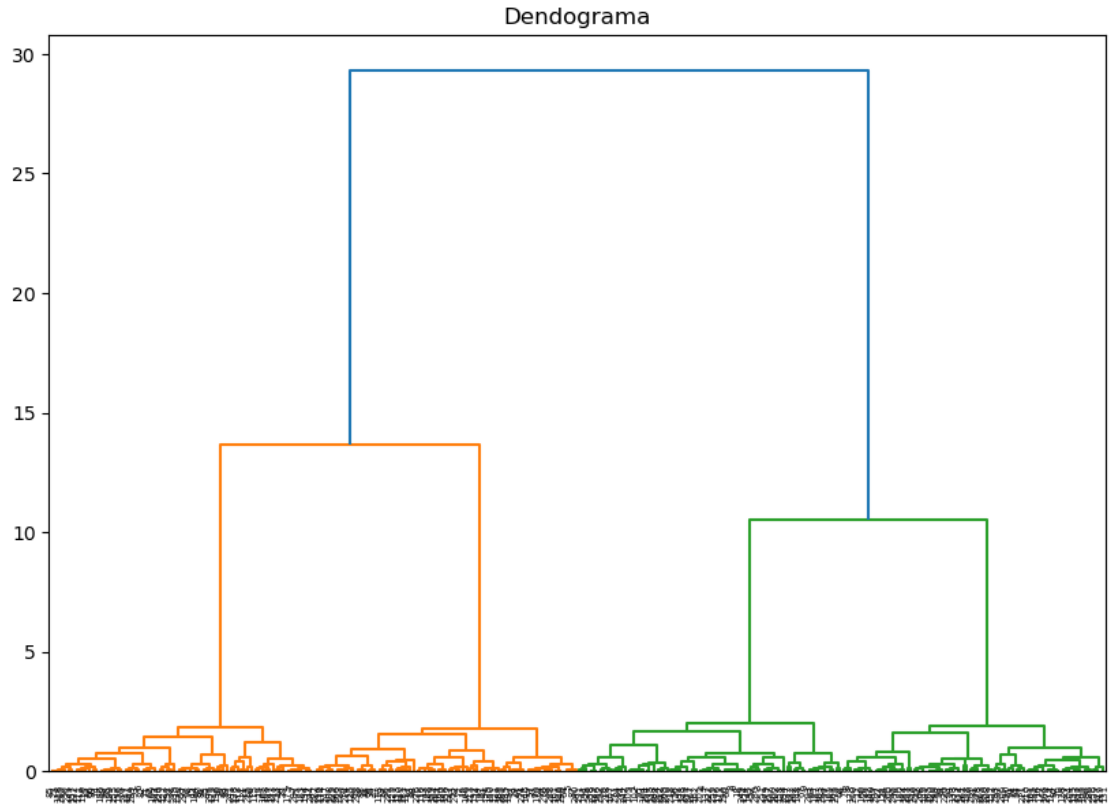
```
Matriz de confusão
[[75  0  0  0]
 [ 0 75  0  0]
 [ 0  0 75  0]
 [ 0  0  0 75]]
```

[32]:
```
print('RELATÓRIO DE CLASSIFICAÇÃO')
print(classification_report(Labels, agg_labels))
```

```
RELATÓRIO DE CLASSIFICAÇÃO
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        75
           1       1.00      1.00      1.00        75
           2       1.00      1.00      1.00        75
           3       1.00      1.00      1.00        75

    accuracy                           1.00       300
   macro avg       1.00      1.00      1.00       300
```

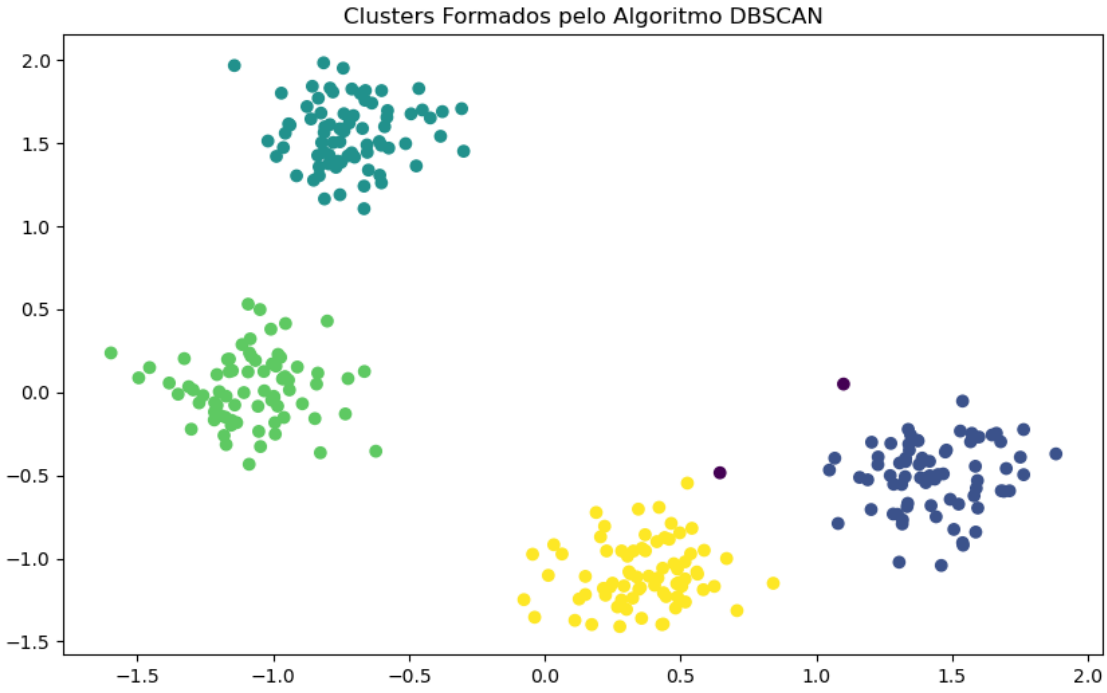
```
weighted avg       1.00      1.00      1.00       300
```

[33]:
```
from scipy.cluster.hierarchy import dendrogram, linkage
# Criando o dendograma
linked = linkage(scalar_caracteristicas, method='ward')

plt.figure(figsize=(10, 7))
dendrogram(linked,
            orientation='top',
            distance_sort='descending',
            show_leaf_counts=True)
plt.title('Dendograma')
plt.show()
```



[34]:
```
from sklearn.cluster import DBSCAN


# Aplicando o algoritmo DBSCAN
dbscan = DBSCAN(eps=0.3, min_samples=8)
dbscan_labels = dbscan.fit_predict(scalar_caracteristicas)
```

```
# Visualizando os clusters formados
plt.figure(figsize=(10, 6))
plt.scatter(scalar_caracteristicas[:, 0], scalar_caracteristicas[:, 1],␣
 ↪c=dbscan_labels)
plt.title('Clusters Formados pelo Algoritmo DBSCAN')
plt.show()
```



[35]:
```
print('Matriz de confusão')
print(confusion_matrix(Labels, dbscan_labels))
```

```
Matriz de confusão
[[ 0  0  0  0  0]
 [ 1 74  0  0  0]
 [ 1  0  0  0 74]
 [ 0  0 75  0  0]
 [ 0  0  0 75  0]]
```

[36]:
```
from sklearn.metrics.cluster import adjusted_rand_score
sc = silhouette_score(scalar_caracteristicas, dbscan_labels)
print("Silhouette Coefficient:%0.2f" % sc)
ari = adjusted_rand_score(Labels, dbscan_labels)
print("Adjusted Rand Index: %0.2f" % ari)
```

```
Silhouette Coefficient:0.69
```

```
Adjusted Rand Index: 0.99
```