Aluno:João Pedro Felicio Prudencio

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

Importando os Dados:

In [1]:
```
import pandas as pd
df = pd.read_csv('Dados-Tarefa-02.csv')
df = df.set_index("Index")
df
```

Out[1]:

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

1000 rows × 2 columns

Normalizando os dados:

In [4]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
data_normalized = scaler.fit_transform(df)
data_normalized
```

In [6]:
```
df_normalized = pd.DataFrame(data_normalized, columns=df.columns)
df_normalized
```

Out[6]:

	d1	d2
0	0.787066	0.443775
1	0.747541	0.356564
2	0.618671	0.356595
3	0.893269	0.350083
4	0.747625	0.670901
...	...	...
995	0.731131	0.467274
996	0.491162	0.512378
997	0.847161	0.544813
998	0.709756	0.410040
999	0.770538	0.408844

1000 rows × 2 columns

K-Means

In [9]:
```
from sklearn.cluster import KMeans
```

In [10]:
```
kmeans_dados = KMeans(n_clusters=6, random_state=0)
rotulos_kmeans = kmeans_dados.fit_predict(df_normalized)

# rotulos_kmeans
```

In [11]:
```
kmeans_dados.cluster_centers_
```

Out[11]:
```
array([[0.5249308 , 0.55861771],
       [0.22503368, 0.32535566],
       [0.65860503, 0.27659636],
       [0.36822755, 0.43941231],
       [0.75676336, 0.46032539],
       [0.84466639, 0.64038163]])
```

In [12]:
```
df_normalized['Cluster K-Means'] = kmeans_dados.labels_
df_normalized
```

Out[12]:

	d1	d2	Cluster K-Means
0	0.787066	0.443775	4
1	0.747541	0.356564	4
2	0.618671	0.356595	2
3	0.893269	0.350083	4
4	0.747625	0.670901	5
...	...	...	...
995	0.731131	0.467274	4
996	0.491162	0.512378	0
997	0.847161	0.544813	5
998	0.709756	0.410040	4
999	0.770538	0.408844	4

1000 rows × 3 columns

K-Medoids

In [13]:
```
from sklearn_extra.cluster import KMedoids
```

In [14]:
```
kmedoids_dados = KMedoids(n_clusters=6, random_state=0)
rotulos_kmedoids = kmedoids_dados.fit_predict(df_normalized)
df_normalized['Cluster K-Medoids'] = kmedoids_dados.labels_

df_normalized
```

Out[14]:

	d1	d2	Cluster K-Means	Cluster K-Medoids
0	0.787066	0.443775	4	1
1	0.747541	0.356564	4	1
2	0.618671	0.356595	2	4
3	0.893269	0.350083	4	1
4	0.747625	0.670901	5	1
...	...	...	...	...
995	0.731131	0.467274	4	1
996	0.491162	0.512378	0	3
997	0.847161	0.544813	5	1
998	0.709756	0.410040	4	1
999	0.770538	0.408844	4	1

1000 rows × 4 columns

DBSCAN

In [15]:
```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.8, min_samples=3)

dbscan_dados = dbscan.fit(df_normalized)

rotulos_dbscan = dbscan_dados.labels_
rotulos_dbscan

df_normalized['Cluster DBSCAN'] = rotulos_dbscan
```

In [16]:
```
df_normalized
```

Out[16]:

	d1	d2	Cluster K-Means	Cluster K-Medoids	Cluster DBSCAN
0	0.787066	0.443775	4	1	0
1	0.747541	0.356564	4	1	0
2	0.618671	0.356595	2	4	1
3	0.893269	0.350083	4	1	0
4	0.747625	0.670901	5	1	2
...	...	...	...	...	...
995	0.731131	0.467274	4	1	0
996	0.491162	0.512378	0	3	5
997	0.847161	0.544813	5	1	2
998	0.709756	0.410040	4	1	0
999	0.770538	0.408844	4	1	0

1000 rows × 5 columns

BIRCH

In [18]:
```
from sklearn.cluster import Birch

birch = Birch(n_clusters=5, threshold=0.2)

birch.fit(df_normalized)

rotulos_birch = birch.labels_

df_normalized['Cluster BIRCH'] = rotulos_birch
```

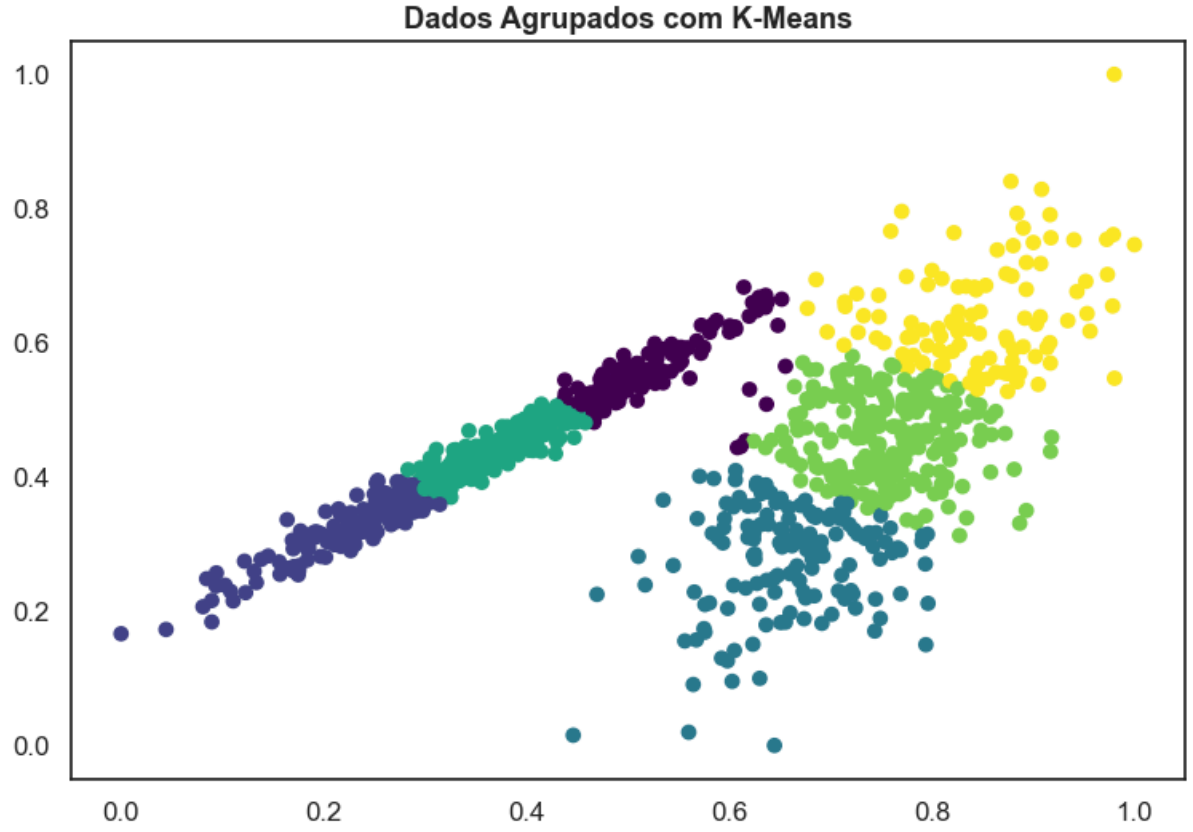
In [19]:
```
df_normalized
```

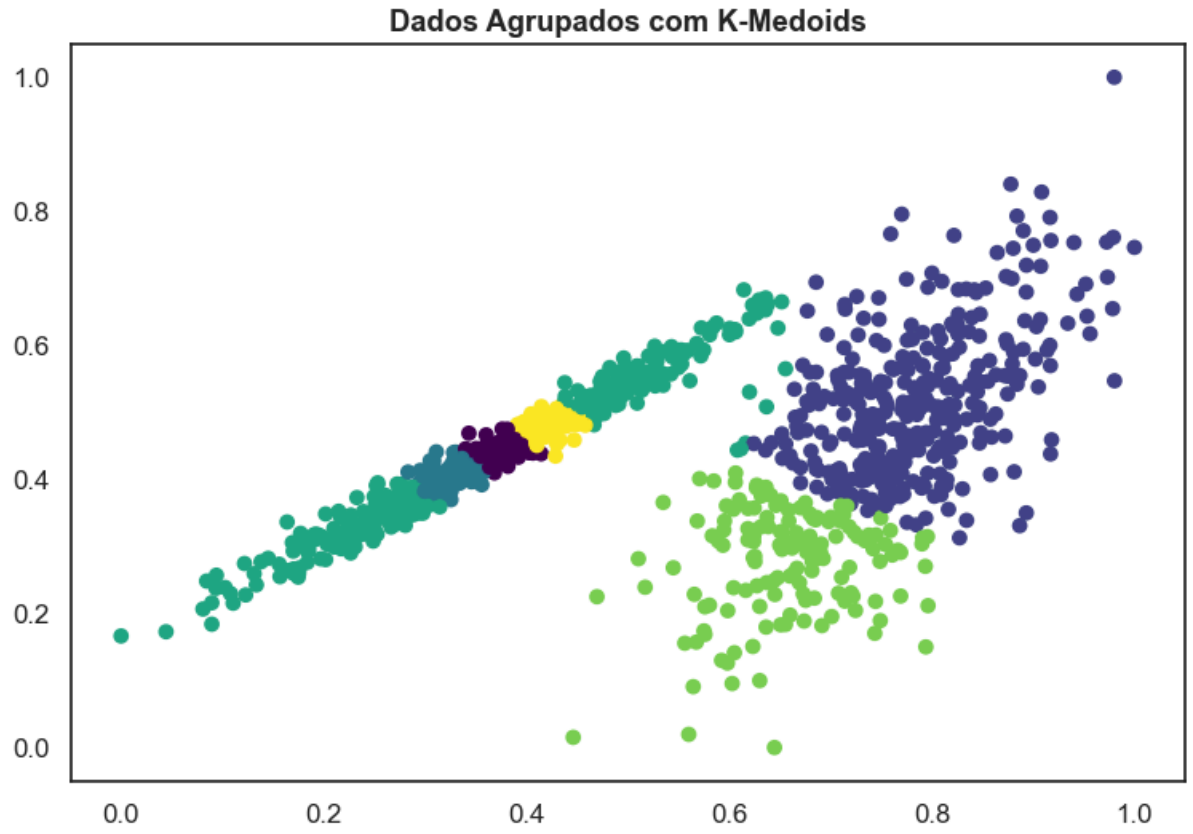
Out[19]:

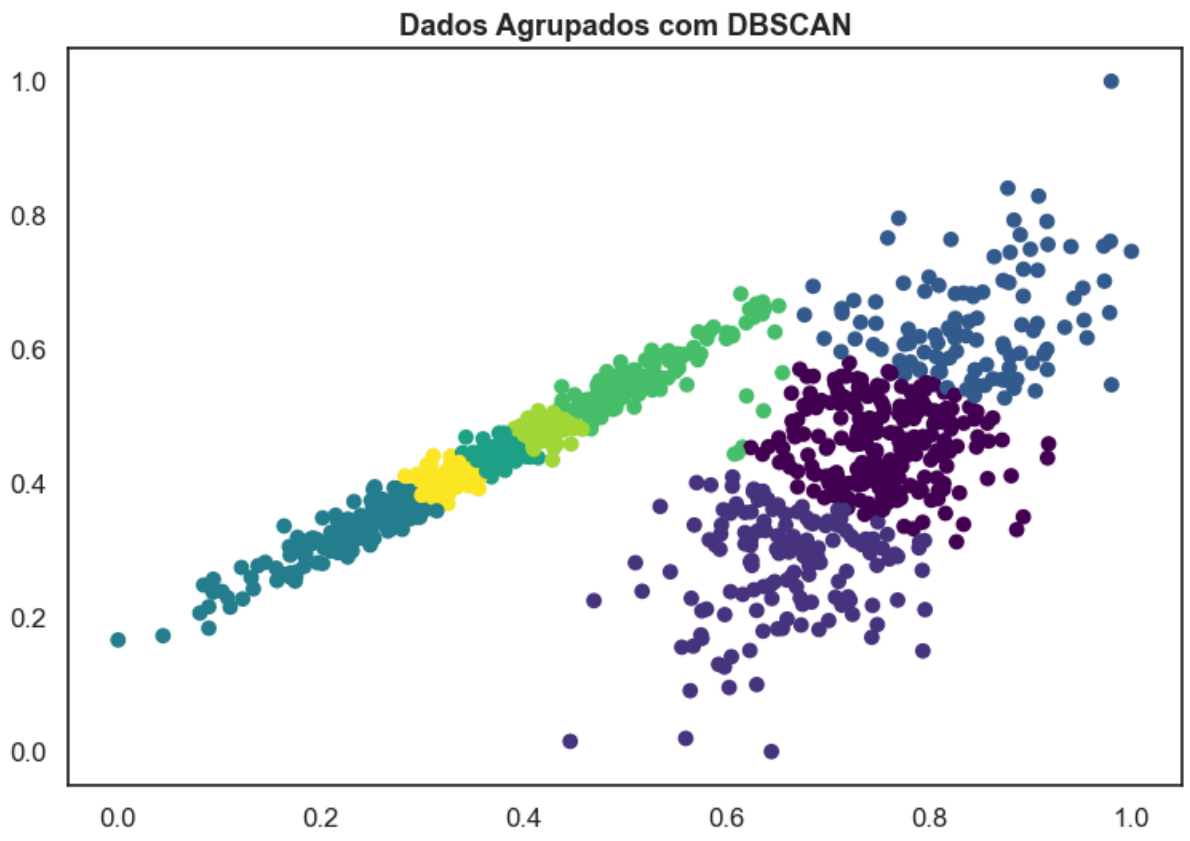
	d1	d2	Cluster K-Means	Cluster K-Medoids	Cluster DBSCAN	Cluster BIRCH
0	0.787066	0.443775	4	1	0	1
1	0.747541	0.356564	4	1	0	1
2	0.618671	0.356595	2	4	1	4
3	0.893269	0.350083	4	1	0	1
4	0.747625	0.670901	5	1	2	1
...	...	...	...	...	...	...
995	0.731131	0.467274	4	1	0	1
996	0.491162	0.512378	0	3	5	2
997	0.847161	0.544813	5	1	2	1
998	0.709756	0.410040	4	1	0	1
999	0.770538	0.408844	4	1	0	1

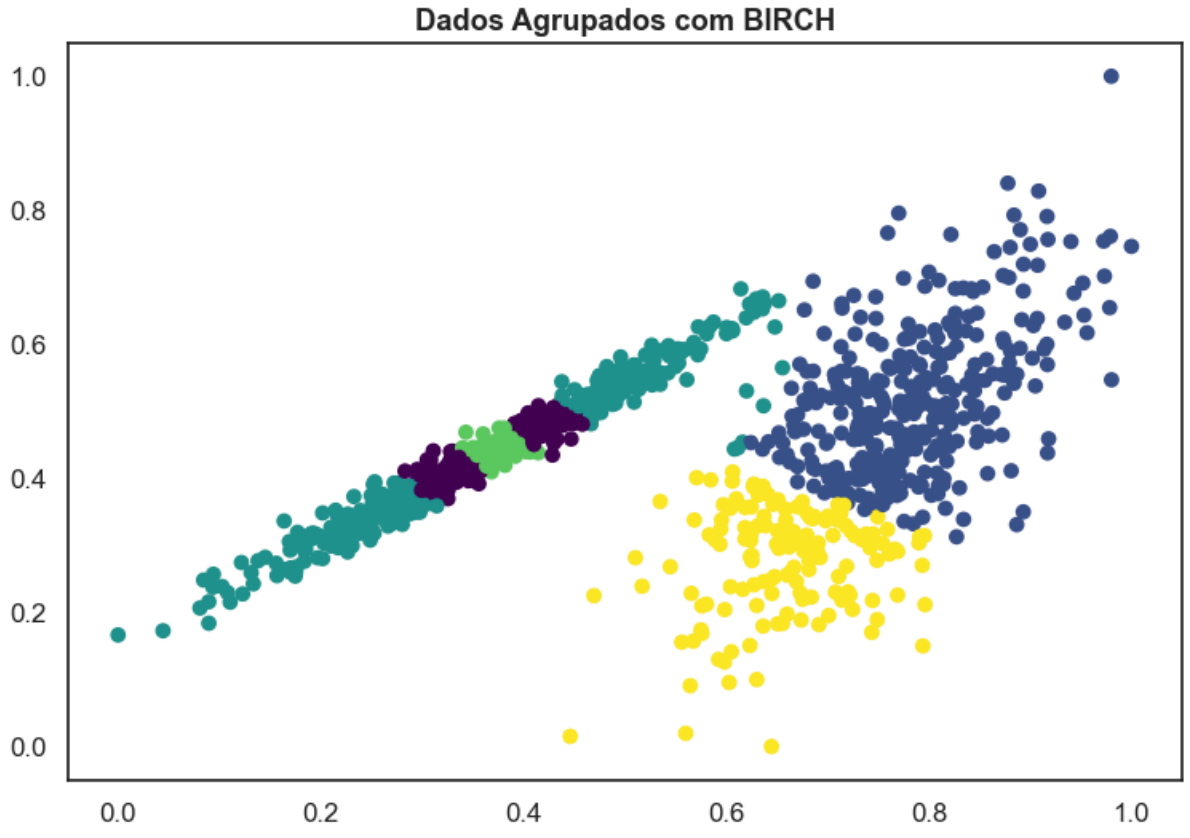
1000 rows × 6 columns

Plots:


Dados Agrupados com K-Means
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Dados Agrupados com K-Medoids
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Dados Agrupados com DBSCAN
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

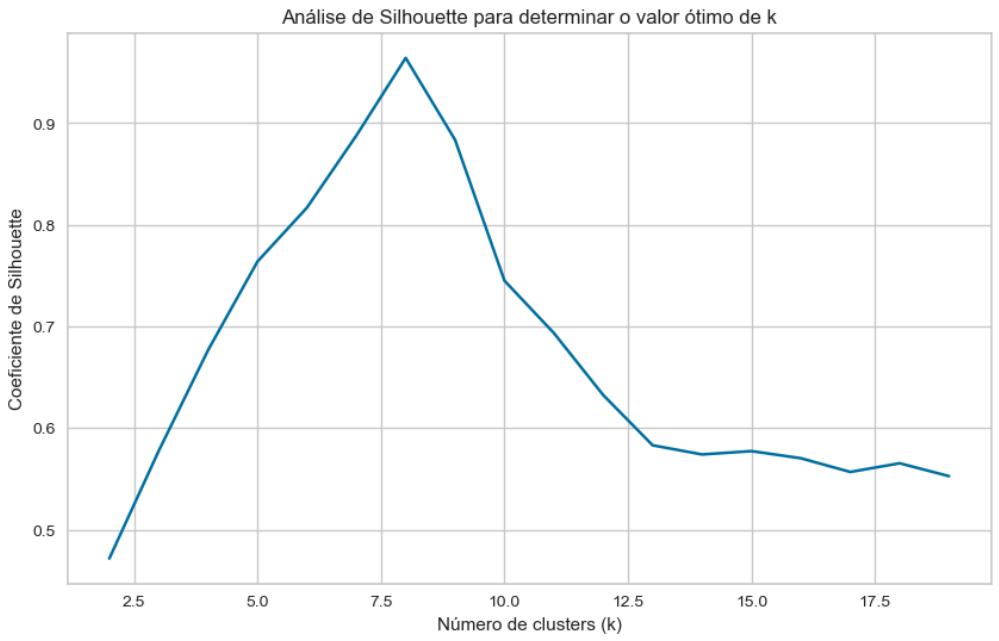

Dados Agrupados com BIRCH
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

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?

Silhouette:

K ótimo para K-Means = 8

In [31]:
```
k_values = range(2, 20)
silhouette_scores = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(df_normalized)
    silhouette_avg = silhouette_score(df_normalized, labels)
    silhouette_scores.append(silhouette_avg)

plt.figure(figsize=(10, 6))
plt.plot(k_values, silhouette_scores, 'bx-')
plt.xlabel('Número de clusters (k)')
plt.ylabel('Coeficiente de Silhouette')
plt.title('Análise de Silhouette para determinar o valor ótimo de k')
plt.show()

optimal_k = k_values[np.argmax(silhouette_scores)]
print(f'Melhor valor de k: {optimal_k}')
```



K ótimo para K-Medoids = 6

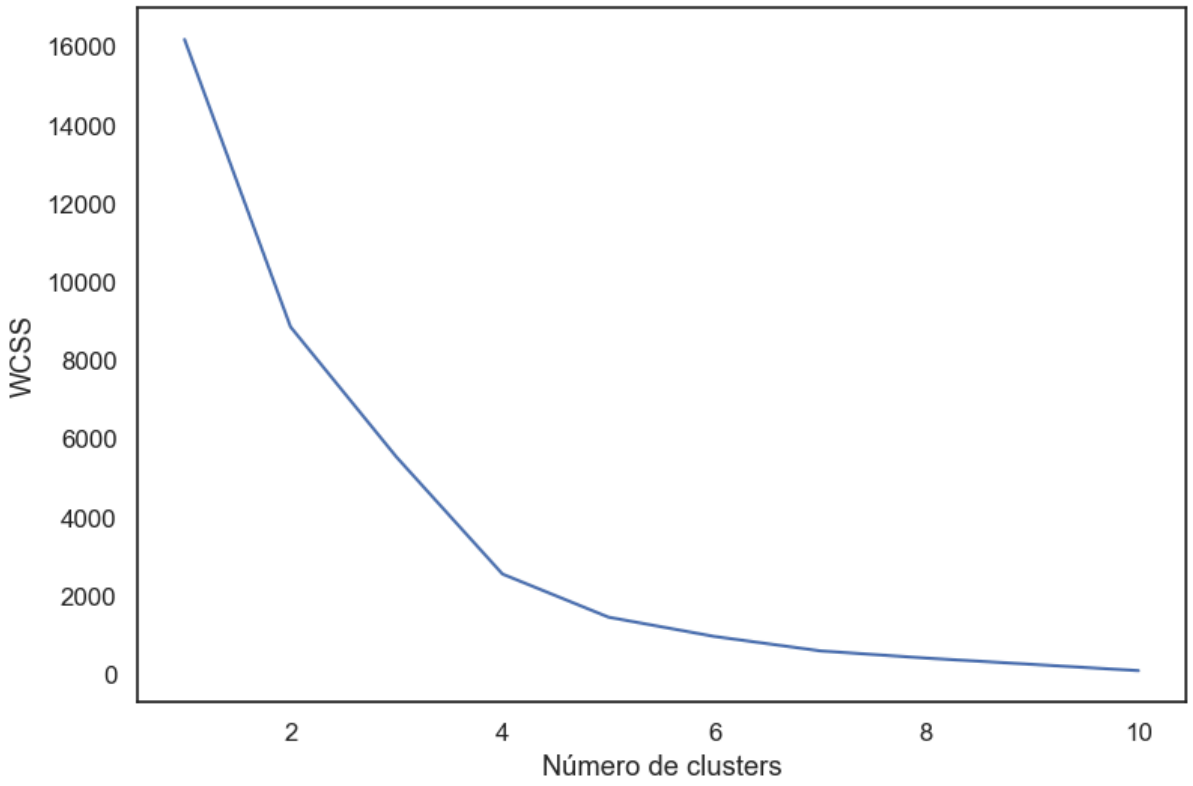
In [32]:
```
k_values = range(2, 11)
silhouette_scores = []

for k in k_values:
    kmedoids = KMedoids(n_clusters=k, random_state=42)
    labels = kmedoids.fit_predict(df_normalized)
    silhouette_avg = silhouette_score(df_normalized, labels)
    silhouette_scores.append(silhouette_avg)

plt.figure(figsize=(10, 6))
plt.plot(k_values, silhouette_scores, 'bx-')
plt.xlabel('Número de clusters (k)')
plt.ylabel('Coeficiente de Silhouette')
plt.title('Análise de Silhouette para determinar o valor ótimo de k')
plt.show()

optimal_k = k_values[np.argmax(silhouette_scores)]
print(f'Melhor valor de k:: {optimal_k}')
```

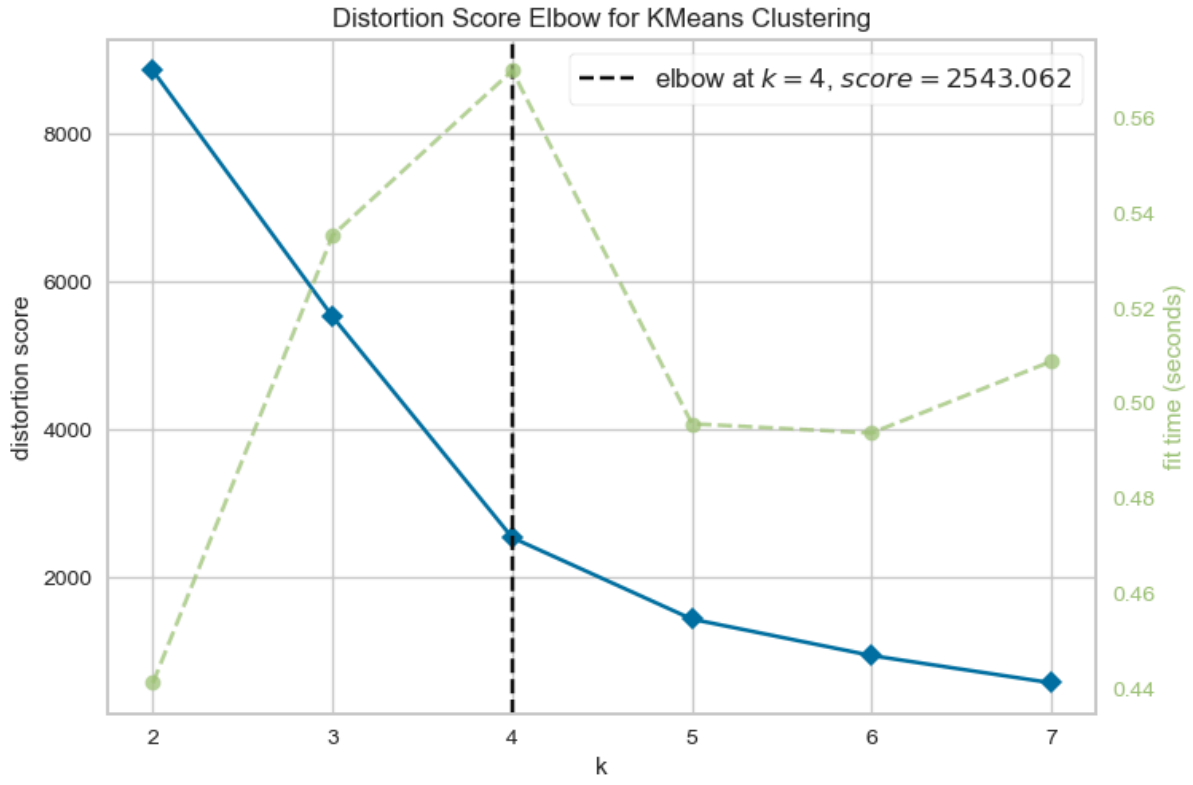
Elbow:

K ótimo para K-Means = 4

In [24]:
```
#Elbow method
#Inertia: soma das distâncias quadradas das amostras até o centro do cluster mais próximo

wcss = []
for i in range(1,11):
  kmeans_dados_e = KMeans(n_clusters=i, random_state=42)
  kmeans_dados_e.fit(df_normalized)
  wcss.append(kmeans_dados_e.inertia_)
```

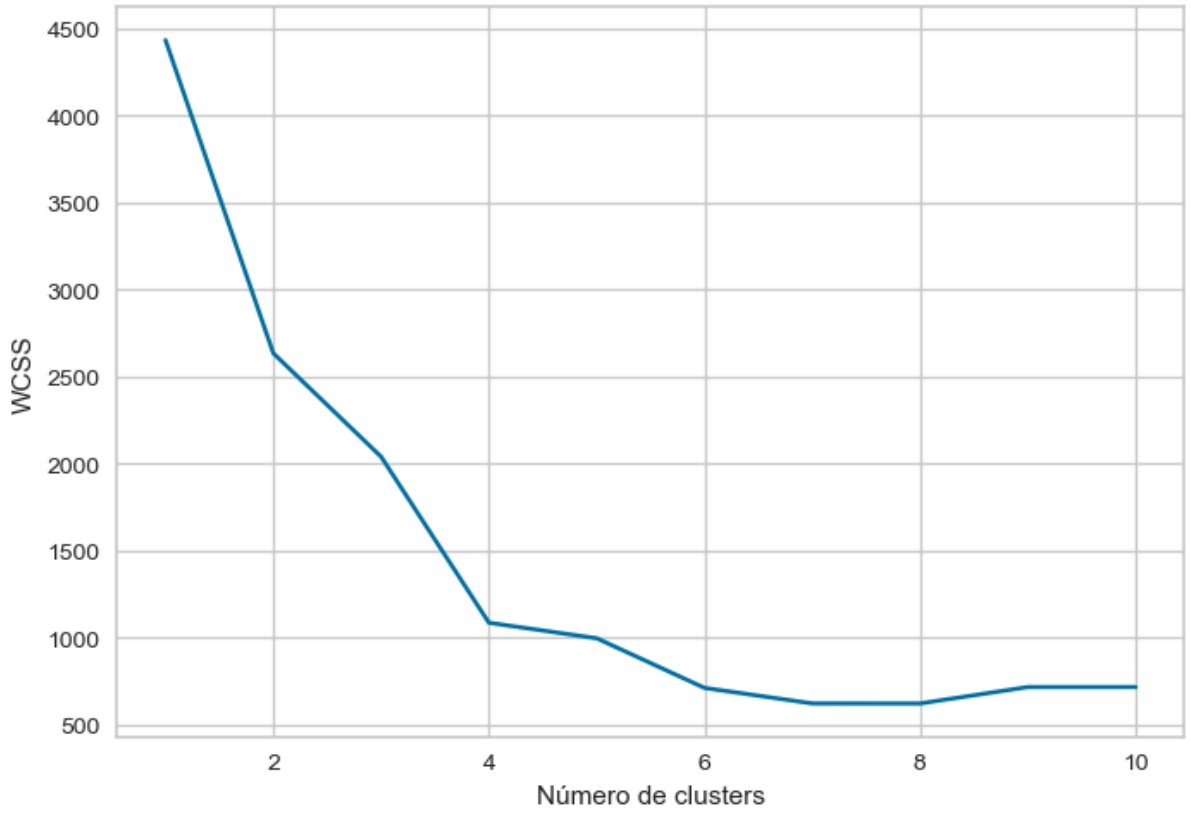
In [26]:
```
plt.plot(range(1, 11), wcss)
plt.xlabel('Número de clusters')
plt.ylabel('WCSS')
plt.show()
```

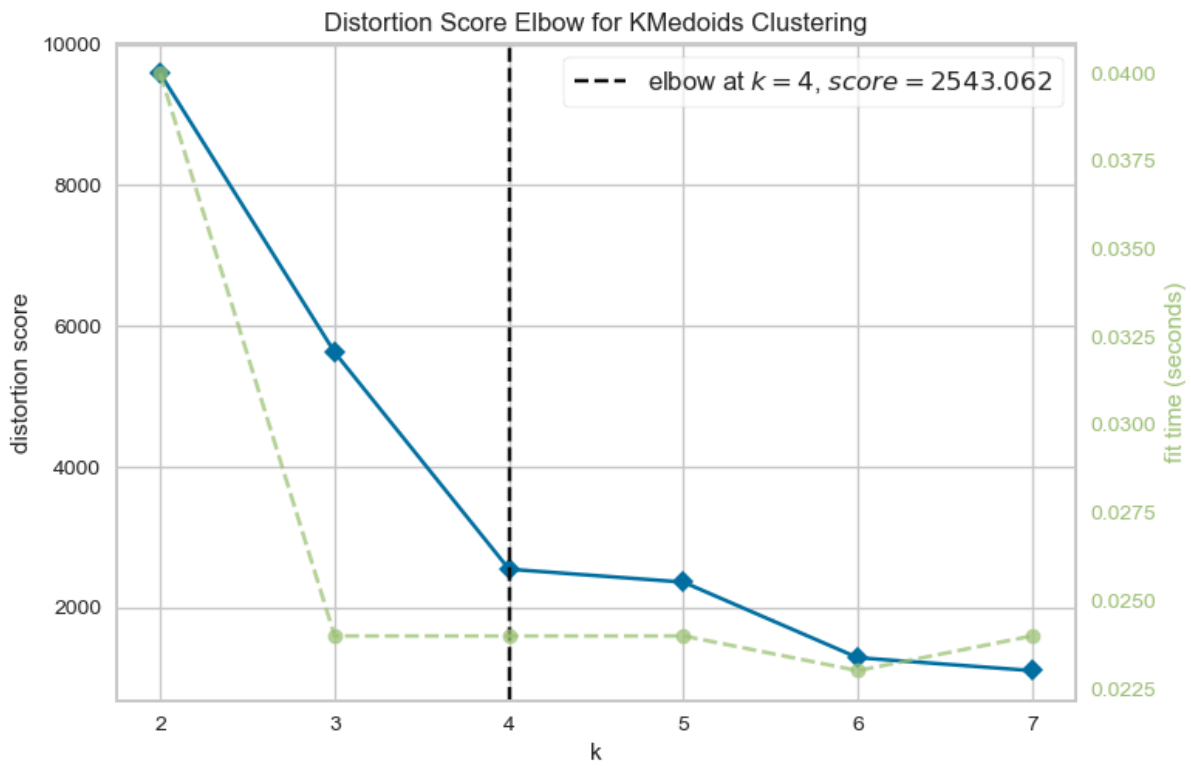

16000
14000
12000
10000
8000
6000
4000
2000
0
WCSS
2
4
6
8
10
Número de clusters


Distortion Score Elbow for KMeans Clustering
elbow at $k = 4$, $score = 2543.062$
8000
6000
4000
2000
distortion score
0.56
0.54
0.52
0.50
0.48
0.46
0.44
fit time (seconds)
2
3
4
5
6
7
k

K ótimo para K-Medoids = 4

In [30]:
```
#Elbow method
#Inertia: soma das distâncias quadradas das amostras até o centro do cluster mais próximo

wcss = []
for i in range(1,11):
  kmedoids_dados_e = KMedoids(n_clusters=i, random_state=42)
  kmedoids_dados_e.fit(df_normalized)
  wcss.append(kmedoids_dados_e.inertia_)

wcss

plt.plot(range(1, 11), wcss)
plt.xlabel('Número de clusters')
plt.ylabel('WCSS')
plt.show()

from yellowbrick.cluster import KElbowVisualizer

kmedoids_dados_elbow = KMedoids(random_state=42)

#distorção: média das distâncias quadradas dos centros dos clusters dos respectivos clusters
grafico = KElbowVisualizer(kmedoids_dados_elbow, k=(2,8))

grafico.fit(df_normalized)
grafico.show()
```

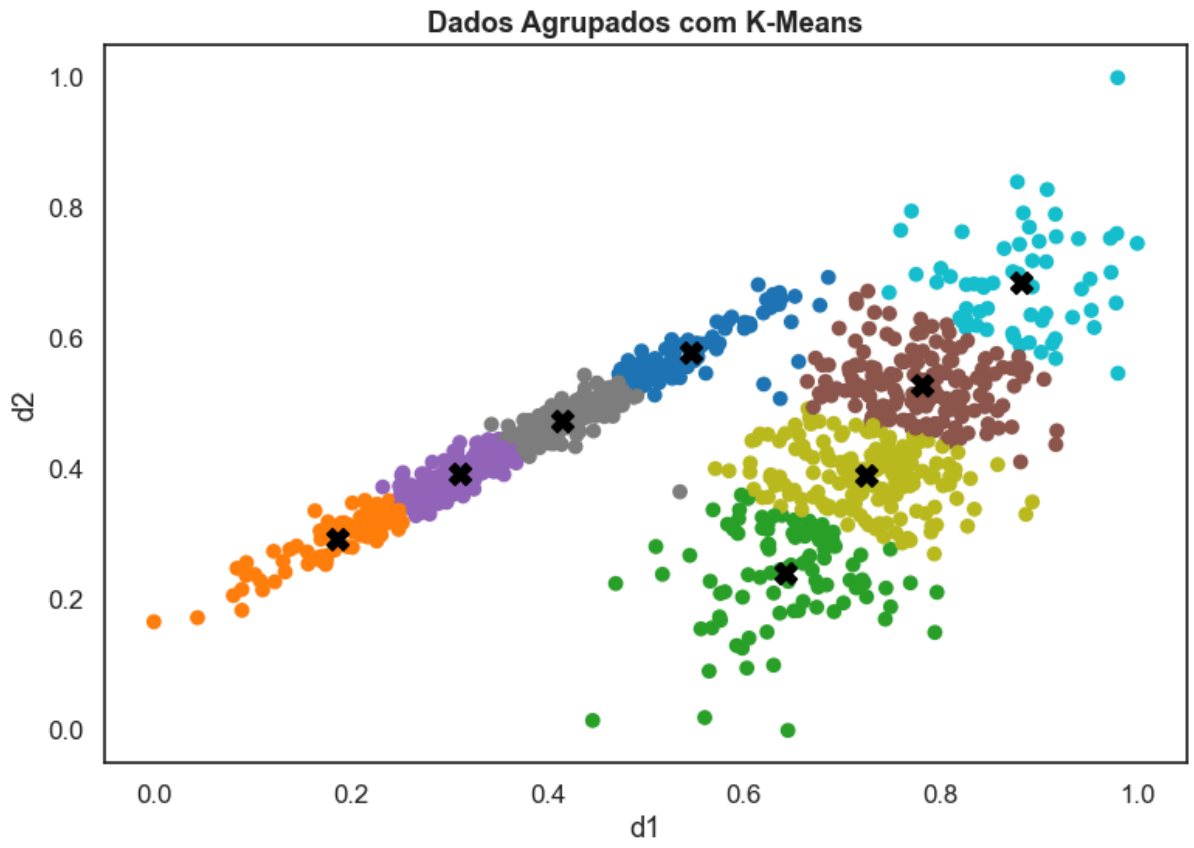



Distortion Score Elbow for KMedoids Clustering
elbow at $k = 4$, $score = 2543.062$
distortion score
fit time (seconds)
k
10000
8000
6000
4000
2000
0.0400
0.0375
0.0350
0.0325
0.0300
0.0275
0.0250
0.0225
2
3
4
5
6
7

3-) Faça gráficos com os dados, mostrando em cores diferentes cada cluster e seu centroide (no k-Means)/Objeto central (K-Metoids)

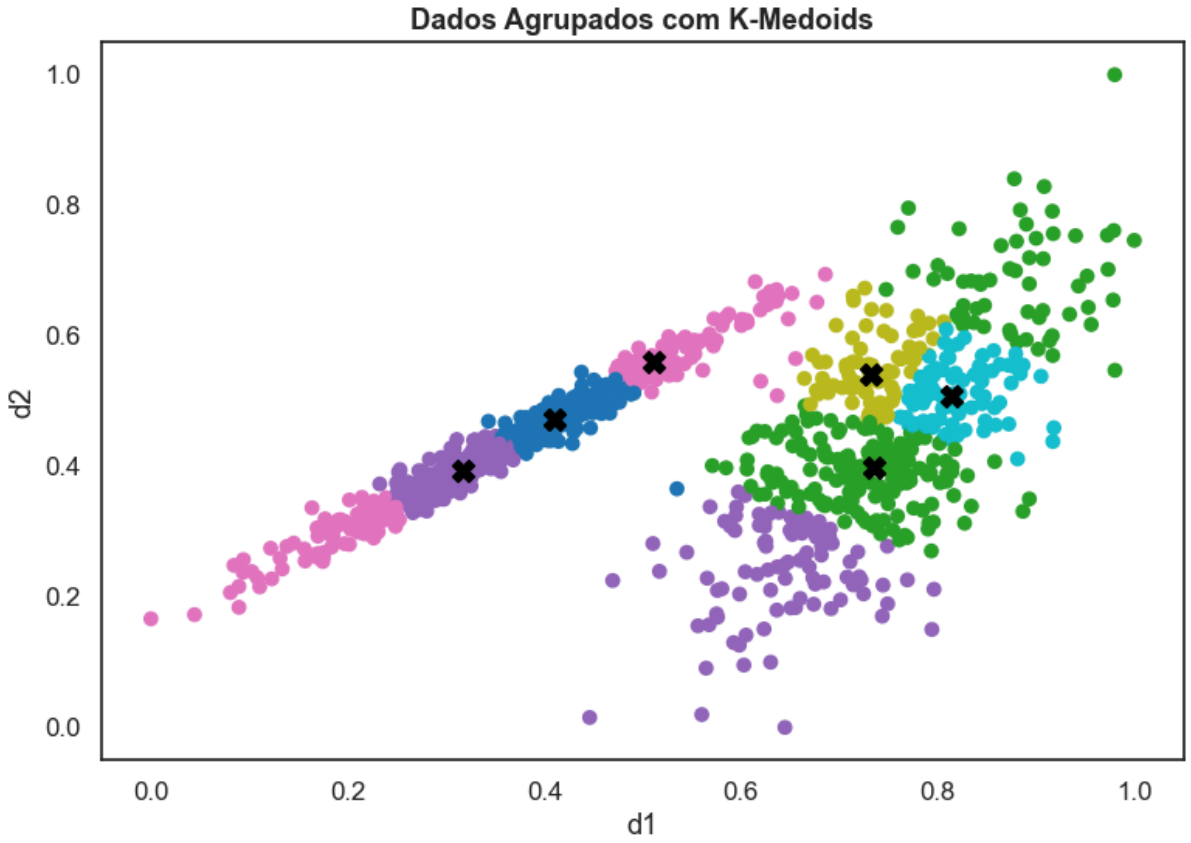
K-Means com K = 8

```
In [8]: sns.set(style='white', rc={'figure.figsize': (9, 6)}, font_scale=1.1)

        scatter = plt.scatter(x=df_normalized.d1, y=df_normalized.d2, c=rotulos_kmeans, cmap='tab10')

        centroids = kmeans_dados.cluster_centers_
        plt.scatter(centroids[:, 0], centroids[:, 1], s=100, c='black', edgecolors='black', linewidth=1, marker='X')

        plt.title('Dados Agrupados com K-Means', fontweight='bold')
        plt.xlabel('d1')
        plt.ylabel('d2')
        plt.show()
```



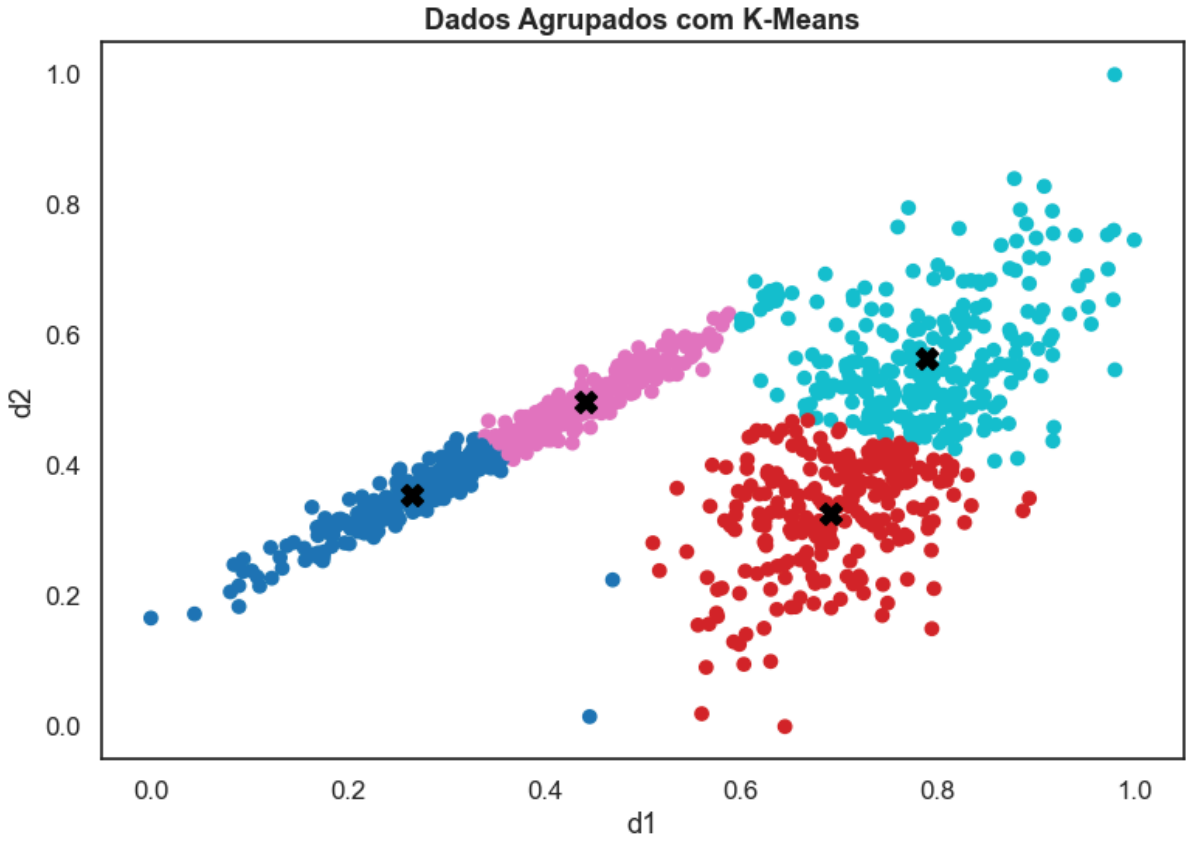
K-Medoids com K = 6

In [11]:
```
sns.set(style='white', rc={'figure.figsize': (9, 6)}, font_scale=1.1)

scatter = plt.scatter(x=df_normalized.d1, y=df_normalized.d2, c=rotulos_kmedoids, cmap='tab10')

medoids = kmedoids_dados.cluster_centers_
plt.scatter(medoids[:, 0], medoids[:, 1], s=100, c='black', edgecolors='black', linewidth=1, marker='X')

plt.title('Dados Agrupados com K-Medoids', fontweight='bold')
plt.xlabel('d1')
plt.ylabel('d2')
plt.show()
```



K-Means com K = 4

In [7]:
```
sns.set(style='white', rc={'figure.figsize': (9, 6)}, font_scale=1.1)

scatter = plt.scatter(x=df_normalized.d1, y=df_normalized.d2, c=rotulos_kmeans, cmap='tab10')

centroids = kmeans_dados.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], s=100, c='black', edgecolors='black', linewidth=1, marker='X')

plt.title('Dados Agrupados com K-Means', fontweight='bold')
plt.xlabel('d1')
plt.ylabel('d2')
plt.show()
```



K-Medoids com K = 4

In [9]:
```
sns.set(style='white', rc={'figure.figsize': (9, 6)}, font_scale=1.1)

scatter = plt.scatter(x=df_normalized.d1, y=df_normalized.d2, c=rotulos_kmedoids, cmap='tab10')

medoids = kmedoids_dados.cluster_centers_
plt.scatter(medoids[:, 0], medoids[:, 1], s=100, c='black', edgecolors='black', linewidth=1, marker='X')

plt.title('Dados Agrupados com K-Medoids', fontweight='bold')
plt.xlabel('d1')
plt.ylabel('d2')
plt.show()
```