```
import pandas as pd
```

```
dt = pd.read_csv('/content/data.csv')
```

```
dt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2017 entries, 0 to 2016
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        2017 non-null   int64
 1   acousticness      2017 non-null   float64
 2   danceability      2017 non-null   float64
 3   duration_ms       2017 non-null   int64
 4   energy            2017 non-null   float64
 5   instrumentalness  2017 non-null   float64
 6   key               2017 non-null   int64
 7   liveness          2017 non-null   float64
 8   loudness          2017 non-null   float64
 9   mode              2017 non-null   int64
 10  speechiness       2017 non-null   float64
 11  tempo             2017 non-null   float64
 12  time_signature    2017 non-null   float64
 13  valence           2017 non-null   float64
 14  target            2017 non-null   int64
 15  song_title        2017 non-null   object
 16  artist            2017 non-null   object
dtypes: float64(10), int64(5), object(2)
memory usage: 268.0+ KB
```

```
x = dt.iloc[:,3:]
x.head()
```

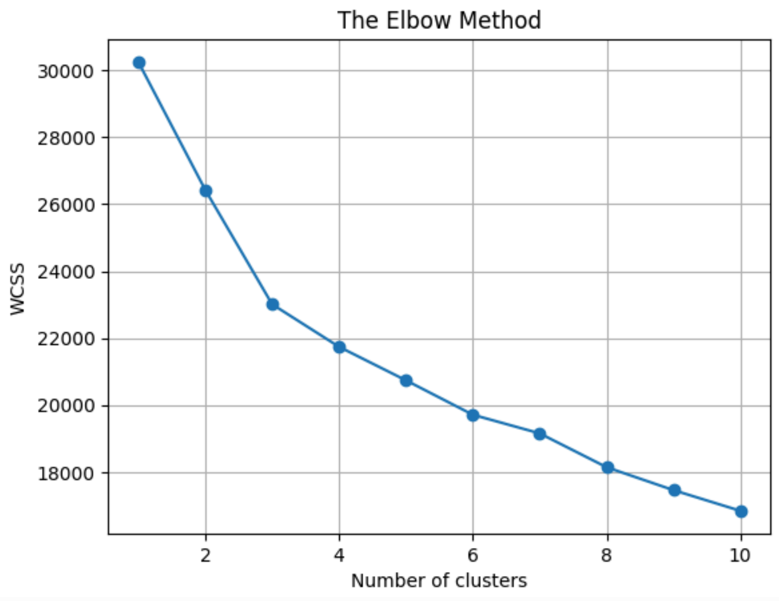
	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence	target	song_
0	204600	0.434	0.021900	2	0.1650	-8.795	1	0.4310	150.062	4.0	0.286	1	Ma
1	326933	0.359	0.006110	1	0.1370	-10.401	1	0.0794	160.083	4.0	0.588	1	Re
2	185707	0.412	0.000234	2	0.1590	-7.148	1	0.2890	75.044	4.0	0.173	1	
3	199413	0.338	0.510000	5	0.0922	-15.236	1	0.0261	86.468	4.0	0.230	1	Mas

```
from sklearn.cluster import KMeans
```

```
wcss = []
x_numeric = dt.drop(['song_title', 'artist'], axis=1).select_dtypes(include=['float64', 'int64'])
from sklearn.preprocessing import StandardScaler
x_scaled = StandardScaler().fit_transform(x_numeric)
```

```
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init="k-means++", random_state=42)
    kmeans.fit(x_scaled)
    wcss.append(kmeans.inertia_)
```

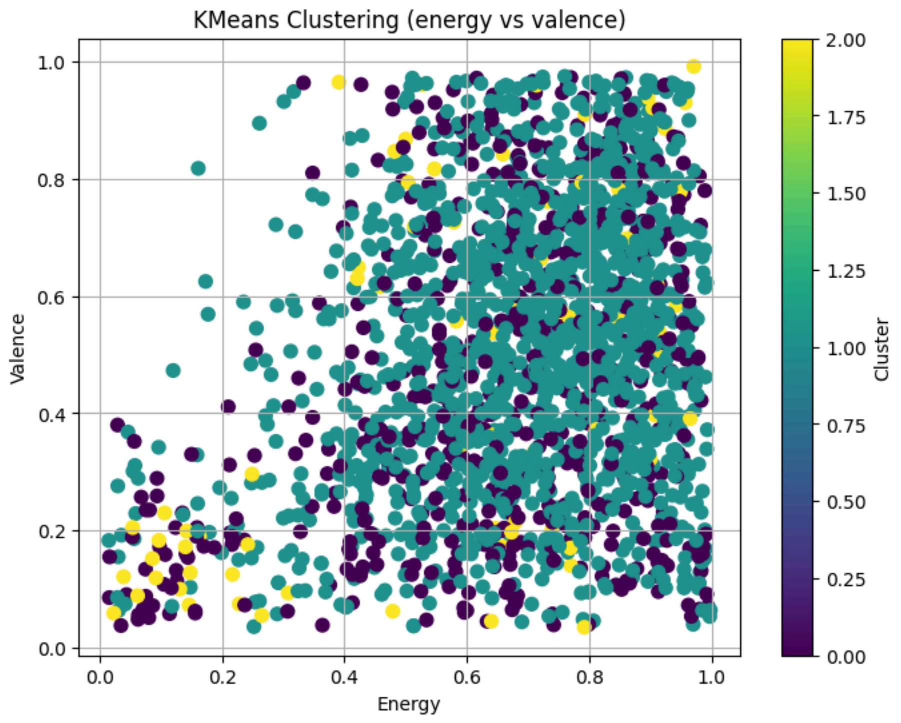
```
import matplotlib.pyplot as plt
plt.plot(range(1, 11), wcss, marker='o')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
```



```
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)
x_kmeans = kmeans.fit_predict(x)
```

```
labels = kmeans.labels_
dt['clusters']=labels
```

```
plt.figure(figsize=(8, 6))
plt.scatter(x['energy'], x['valence'], c=y_kmeans, cmap='viridis', s=50)
plt.title("KMeans Clustering (energy vs valence)")
plt.xlabel("Energy")
plt.ylabel("Valence")
plt.colorbar(label='Cluster')
plt.grid(True)
plt.show()
```



Start coding or generate with AI.