PRAKTIKUM DATA WAREHOUSING DAN DATA MINING

MODUL 11

CLUSTERING: ALGORITMA K-MEANS

Disusun oleh:

Adinda Aulia Hapsari

L200220037

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS KOMUNIKASI DAN INFORMATIKA

UNIVERSITAS MUHAMMADIYAH SURAKARTA

TAHUN 2024

Setelah kegiatan selesai, lembar kerja ini dicetak (di-print) dan dikumpulkan ke asisten. NIM : L200220037 Nama : Adinda Aulia Hapsari Nama Asisten : Diva Halimah Tanggal Praktikum : 13 Desember 2024	(Diisi oleh Asisten) Nilai Praktek : Tanda Tangan :

KEGIATAN PRAKTIKUM

Contoh Kasus:

Sekarang kita akan melihat contoh nyata dari pengelompokan k-means di mana kita akan membuat segmen pelanggan berdasarkan pendapatan tahunan mereka dan metrik yang akan kita sebut skor pengeluaran (dari 1 hingga 100). Dalam kumpulan data, pelanggan yang menghabiskan lebih banyak diberi skor lebih tinggi dibandingkan dengan pelanggan yang berbelanja lebih sedikit. Tujuan dari pengelompokan ini adalah mengidentifikasi pelanggan dengan pendapatan tinggi dan skor pengeluaran tinggi. Ini adalah pelanggan yang dapat ditargetkan lebih banyak dalam promosi dan kampanye pemasaran sehngga dapat lebih memaksimalkan keuntungan. Pada eksperimen ini kita menggunakan dataset Mall_Customers.csv yang terdapat di tautan sesuai tertulis di bagian alat dan bahan.

1. Buka Jupyter Notebook yang tersedia di masing-masing komputer kalian.
2. Pada langkah pertama, kita perlu memanggil beberapa library yang akan kita gunakan untuk melakukan eksperimen ini seperti pada potongan kode berikut ini:

```
[23]: from sklearn.cluster import KMeans

      import numpy as np
      import pandas as pd

      import seaborn as sns
      import matplotlib.pyplot as plt
      sns.set_style("darkgrid")
```

   Penjelasan :
   a. Kita perlu memanggil method KMeans karena kita akan menggunakan metode ini untuk melakukan clustering.
   b. Kemudian pandas akan kita gunakan untuk membaca data dari dataset.
   c. Terakhir, seaborn dan pyplot akan digunakan untuk memvisualisasikan data.

3. Langkah selanjutnya kita dapat melakukan pembacaan terhadap dataset yang akan kita gunakan. Untuk membaca/mengimpor dataset gunakan potongan kode berikut ini:

```
[24]: X = pd.read_csv('Mall_Customers.csv')
```

   Untuk membaca file CSV kita dapat menggunakan method yang ada pada pandas.

4. Selanjutnya kita bisa lihat 5 data teratas untuk mendapat gambaran dari data yang akan kita cluster. Untuk melihat 5 data teratas bisa menggunakan potongan kode berikut:

```
[25]: X.head()
```

[25]:

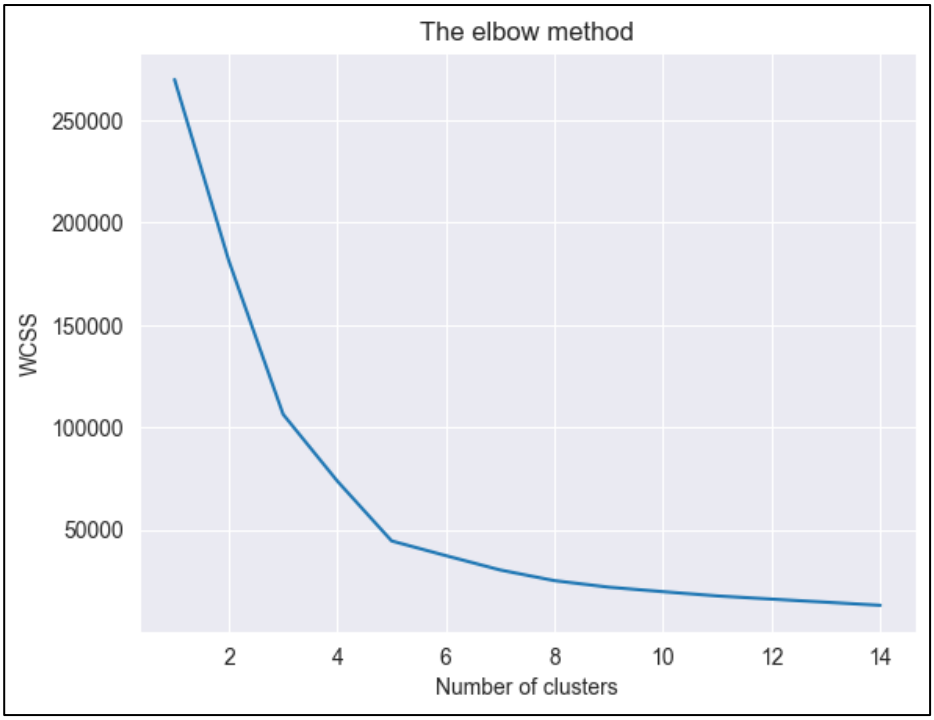
	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

5. Selanjutnya kita perlu untuk melakukan filter terhadap data yang ingin digunakan sebagai dasar pengelompokan. Sesuai yang sudah disampaikan sebelumnya, bahawa pengelompokan untuk data pemasaran ini berdasarkan annual income (pendapatan tahunan) dan spending score (skor pengeluaran). Sehingga untuk memilih kolom data yang digunakan dapat dengan menjalankan potongan kode berikut.

```
[26]: X = X.filter(["Annual Income (k$)",
                    "Spending Score (1-100)"], axis = 1)
```

6. Langkah awal untuk melakukan pengelompokan dengan algoritma K-means adalah menentukan jumlah cluster. Sekarang kita akan menerapkan Elbow Method pada dataset di atas. Elbow method ini memungkinkan kita untuk memilih jumlah cluster yang optimal untuk pengelompokan data. Untuk menerapkan Elbow method, silahkan gunakan potongan kode berikut ini.

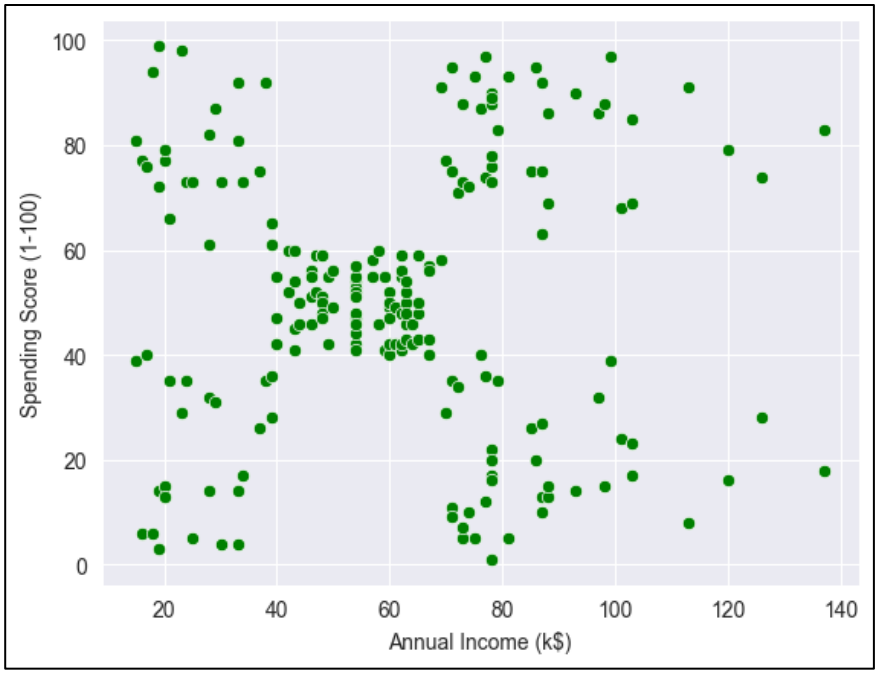
```
[34]: wcss = []

for i in range(1, 15):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
                    max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

#melakukan plot untuk hasil sehingga bisa
#melakukan observasi terhadap elbow
plt.plot(range(1, 15), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #WCSS = within cluster sum of squares
plt.show()
```



Kalian dapat melihat dengan jelas mengapa disebut elbow method dari grafik di atas. Jumlah cluster optimal adalah tempat dimana siku terjadi. Hal ini terjadi ketika WCSS (within cluster sum of squares) tidak berkurang secara signifikan pada iterasi berikutnya. Pada grafik di atas berarti jumlah cluster yang paling optimum adalah 5. Sekarang kita telah memiliki jumlah cluster yang optimal, selanjutnya kita bisa lakukan pengelompokan dengan algoritma K-means pada dataset penjualan di atas.

7. Selain dengan elbow method, kita juga bisa menganalisa kemungkinan jumlah cluster dengan melakukan visualisasi data dengan melakukan plot untuk masing-masing data point. Untuk melakukan plot kita bisa memanfaatkan library seaborn seperti potongan kode berikut ini.

```
[35]: sns.scatterplot(data = X, x="Annual Income (k$)",y="Spending Score (1-100)",
                     c=["green"])
```



Dari gambar visualisasi data di atas, kalian dapat melihat bahwa data pelanggan secara kasar dibagi menjadi 5 cluster, satu cluster di masing-masing dari empat sudut dan satu cluster di tengah. Sehingga hasil ini memverifikasi hasil dari elbow method sebelumnya.

8. Setelah kita mendapatkan jumlah cluster yang paling optimal, selanjutnya kita dapat melakukan pengelompokan data pelanggan dengan menggunakan algoritma K-means dengan menjalankan kode di bawah ini.

```
[36]: model = KMeans(n_clusters= 5)
      model.fit(X)

[36]: KMeans
      KMeans(n_clusters=5)
```

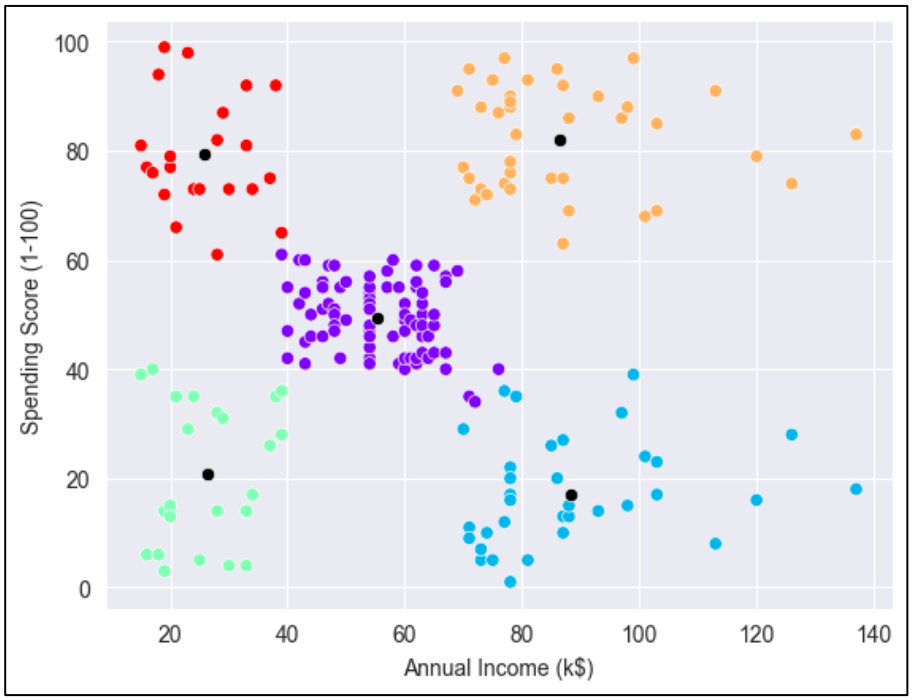
9. Setelah mengeksekusi kode di atas, data secara otomatis akan dikelompokan menjadi lima kelompok. Kita bisa mengecek centroid untuk masing-masing kelompok data dengan kode di bawah ini.

```
[37]: print(model.cluster_centers_)

      [[55.2962963  49.51851852]
       [88.2        17.11428571]
       [26.30434783 20.91304348]
       [86.53846154 82.12820513]
       [25.72727273 79.36363636]]
```

Dari eksekusi kode di atas akan ditampilkan titik data yang merupakan centroid untuk masing-masing kelompok yang dihasilkan dari algoritma K-means.

10. Output di atas menunjukkan bahwa nilai centroid untuk kedua koordinat (skor pendapatan dan pengeluaran tahunan) paling tinggi untuk cluster ke-4 (pada indeks 3). Mungkin kelompok ini adalah segmen pelanggan yang akan ditargetkan dengan kampanye pemasaran karena mereka memiliki pendapatan tahunan tertinggi dan kemungkinan pengeluaran uang tertinggi. Selanjutnya kita juga bisa memvisualisasikan hasil pengelompokan data dengan warna yang berbeda untuk masing-masing cluster dengan menggunakan kode di bawah ini.

```
[38]: sns.scatterplot(data = X, x="Annual Income (k$)", y="Spending Score (1-100)",
                      c= model.labels_, cmap='rainbow')

      sns.scatterplot(x=model.cluster_centers_[:, 0],
                      y=model.cluster_centers_[:, 1],
                      c=['black'])
```



TUGAS

Dikerjakan saat ini, jika tidak selesai bisa dilanjutkan di rumah.

1. Spotify adalah layanan streaming musik, podcast, dan video digital yang memberi Anda akses ke jutaan lagu dan konten lain dari artis di seluruh dunia. Discover Weekly, Daily Mix, dan Spotify Wrapped tahunan adalah beberapa fitur yang juga memanfaatkan teknologi sistem rekomendasi. Pernahkah anda mencari lagu dan akhirnya menemukan banyak lagu serupa yang Anda sukai dan simpan secara instan? Pada tugas kali ini kita akan melakukan eksperimen dengan Spotify dataset. Dataset yang digunakan adalah file spotify_data.csv yang dapat diperoleh di tautan tertera pada bagian alat dan bahan.
2. Setelah mengunduh dataset, lakukan pengelompokan data lagu dari spotify untuk mendapatkan playlist berdasarkan mood dari musik. Kalian bisa melakukannya dengan mengikuti langkah-langkah seperti yang dilakukan pada sesi praktikum sebelumnya. Dataset ini terdiri dari 17 kolom, namun yang akan digunakan digunakan sebagai fitur dalam pengelompokan hanya 14 kolom dari kolom 3 sampai kolom 16. Beberapa hal yang perlu diperhatikan:
   a. Pada sesi praktikum, sebagai dasar penentuan cluster hanya digunakan dua fitur. Sedangkan pada tugas akan digunakan 14 fitur yang berbeda.
   b. Karena fitur yang digunakan lebih dari dua, untuk memvisualisasikan hasil pengelompokan, kalian perlu melakukan plot dalam ruang 3 dimensi (opsional untuk nilai yang lebih maksimal)

```
[43]: #tugas 2a
      A = pd.read_csv('spotify_data.csv')
```

```
[44]: A.head()
```

[44]:

	Unnamed: 0	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence	target
0	0	0.0102	0.833	204600	0.434	0.021900	2	0.1650	-8.795	1	0.4310	150.062	4.0	0.286	1
1	1	0.1990	0.743	326933	0.359	0.006110	1	0.1370	-10.401	1	0.0794	160.083	4.0	0.588	1
2	2	0.0344	0.838	185707	0.412	0.000234	2	0.1590	-7.148	1	0.2890	75.044	4.0	0.173	1
3	3	0.6040	0.494	199413	0.338	0.510000	5	0.0922	-15.236	1	0.0261	86.468	4.0	0.230	1
4	4	0.1800	0.678	392893	0.561	0.512000	5	0.4390	-11.648	0	0.0694	174.004	4.0	0.904	1

[45]:
```
A.info()
```

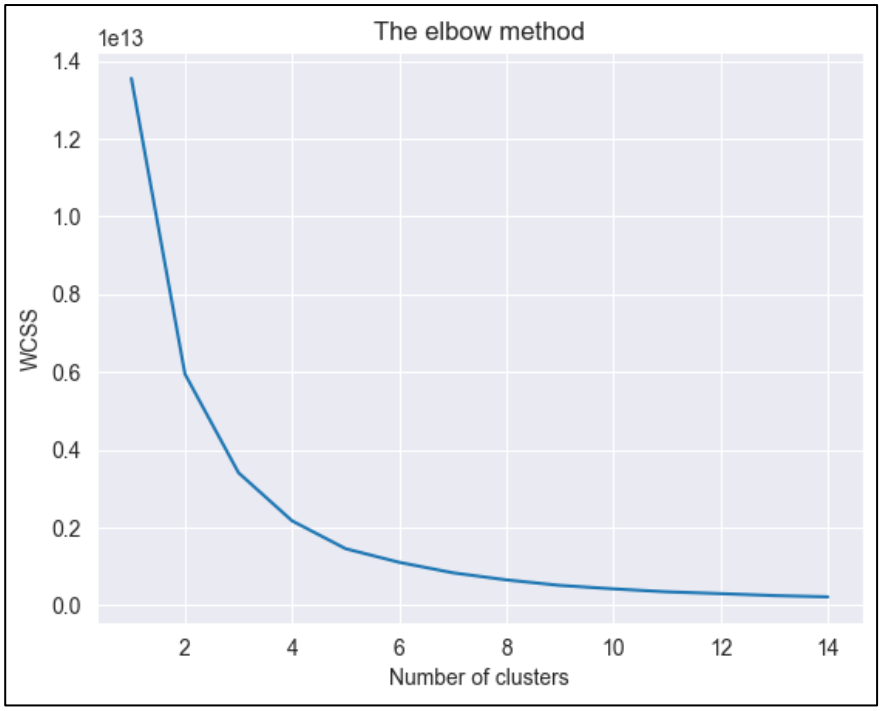
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2017 entries, 0 to 2016
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        2017 non-null   int64
 1   acousticness      2017 non-null   float64
 2   danceability      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
```

[46]:
```
A.drop(columns="Unnamed: 0",inplace=True)
A.drop(columns="artist",inplace=True)
A.drop(columns="song_title",inplace=True)
A.isnull().sum()
```

[46]:
```
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
key                 0
liveness            0
loudness            0
mode                0
speechiness         0
tempo               0
time_signature      0
valence             0
target              0
dtype: int64
```

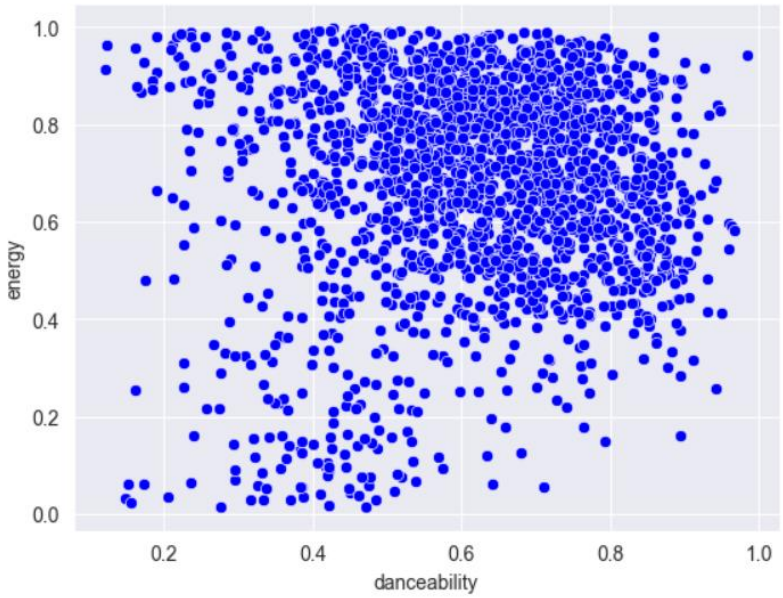
[47]:
```
xi = A.filter(["danceability", "energy"], axis = 1)
```

[48]:
```
wcss = []

for i in range (1, 15):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(A)
    wcss.append(kmeans.inertia_)

#Melakukan plot untuk hasil sehingga bisa melakukan observasi terhadap elbow
plt.plot(range(1,15), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #WCSS = within cluster sum of squares
plt.show()
```




```
[49]: sns.scatterplot(data = xi, x="danceability", y="energy", c = ["blue"])
```

```
[49]: <Axes: xlabel='danceability', ylabel='energy'>
```



```
[50]: model = KMeans(n_clusters=5)
      model.fit(xi)
```

[50]: KMeans

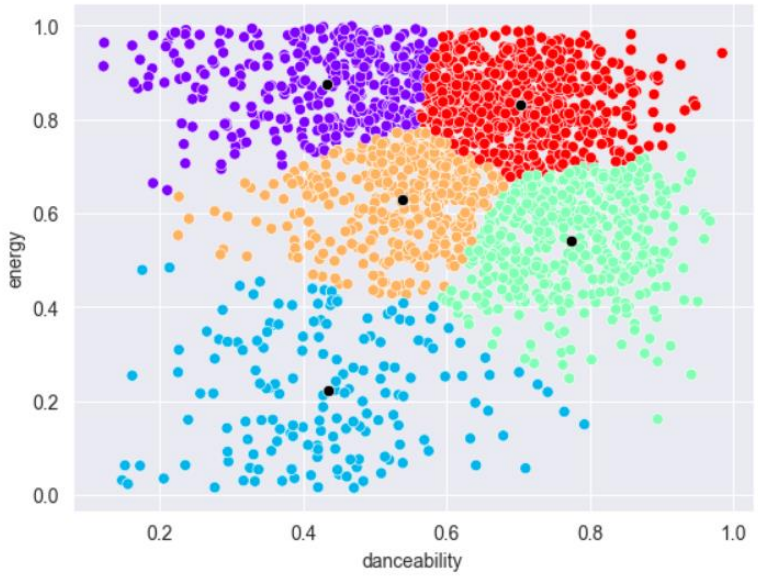
KMeans(n_clusters=5)

```
[51]: print(model.cluster_centers_)
```

```
[[0.78329909 0.54740411]
 [0.43407242 0.86755153]
 [0.43615432 0.21140802]
 [0.54375381 0.60488579]
 [0.69810843 0.82975   ]]
```

```
[71]: sns.scatterplot (data = xi, x="danceability", y="energy",  c=model.labels_, cmap = 'rainbow')
      sns.scatterplot(x=model.cluster_centers_[:, 0], y=model.cluster_centers_[:, 1], c=['black'])
```

[71]: <Axes: xlabel='danceability', ylabel='energy'>




```
[112]: xii = A.filter(["acousticness", "danceability", "energy", "duration_ms", "instrumentalness",
                       "key", "liveness", "loudness", "mode", "speechiness", "tempo","time_signature",
                       "valence", "target"], axis = 1)
```

```
[113]: xii.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2017 entries, 0 to 2016
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   acousticness      2017 non-null   float64
 1   danceability      2017 non-null   float64
 2   energy            2017 non-null   float64
 3   duration_ms       2017 non-null   int64
 4   instrumentalness  2017 non-null   float64
 5   key               2017 non-null   int64
 6   liveness          2017 non-null   float64
 7   loudness          2017 non-null   float64
 8   mode              2017 non-null   int64
 9   speechiness       2017 non-null   float64
 10  tempo             2017 non-null   float64
 11  time_signature    2017 non-null   float64
 12  valence           2017 non-null   float64
 13  target            2017 non-null   int64
dtypes: float64(10), int64(4)
memory usage: 220.7 KB
```

[114]:
```
model = KMeans(n_clusters=5)
model.fit(xii)
```

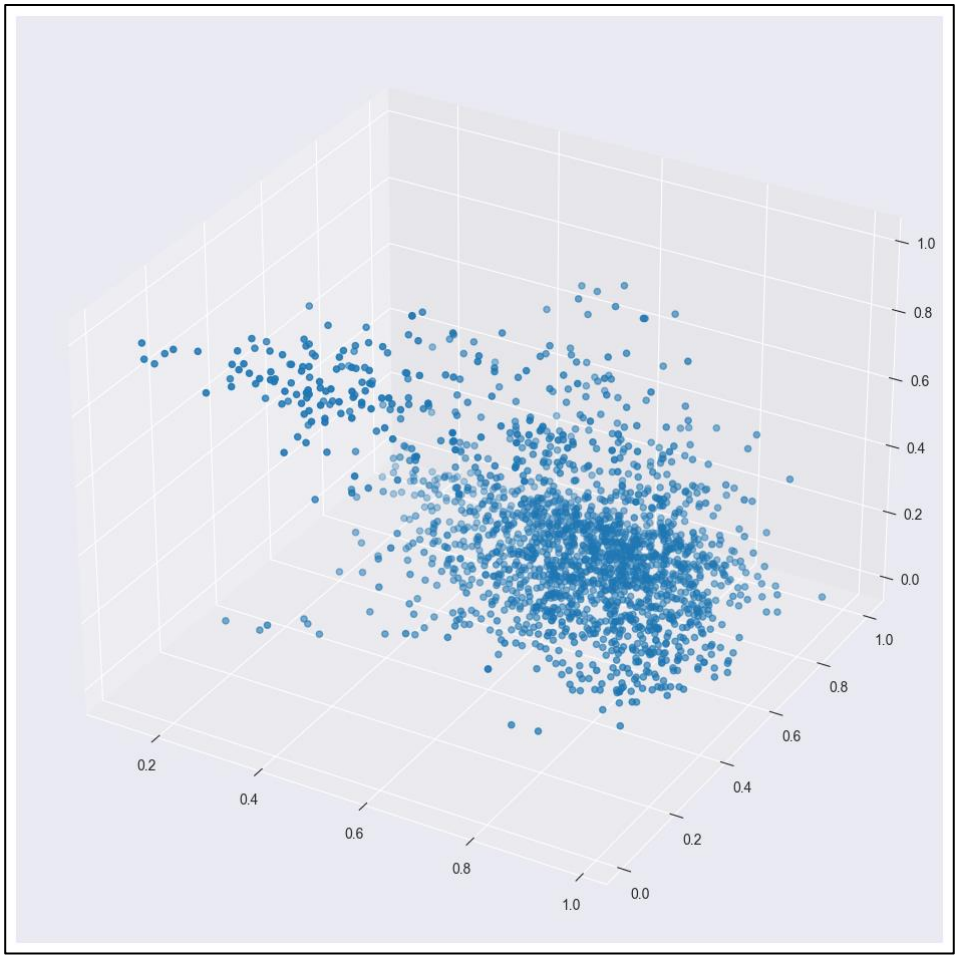
[114]:
```
KMeans
KMeans(n_clusters=5)
```

[115]:
```
print(model.cluster_centers_)
```

```
[[ 1.68501873e-01  6.15379350e-01  6.89916473e-01  2.34970255e+05
   8.49298959e-02  5.34918794e+00  1.89133643e-01 -6.64398608e+00
   6.25290023e-01  8.99537123e-02  1.21843194e+02  3.97795824e+00
   4.86303364e-01  4.54756381e-01]
 [ 2.53300797e-01  6.57943548e-01  5.90810484e-01  4.17840073e+05
   4.26390304e-01  5.67741935e+00  1.64004032e-01 -1.04763065e+01
   5.32258065e-01  6.74709677e-02  1.17457419e+02  3.96774194e+00
   4.58454032e-01  7.58064516e-01]
 [ 1.96398028e-01  6.20175194e-01  7.05781705e-01  1.80349462e+05
   8.92426804e-02  5.11317829e+00  1.90197674e-01 -6.45820310e+00
   6.43410853e-01  1.01347132e-01  1.23130724e+02  3.97054264e+00
   5.45309767e-01  4.52713178e-01]
 [ 1.83011743e-01  6.14252778e-01  6.55661111e-01  3.02907000e+05
   2.08438187e-01  5.67500000e+00  1.98371944e-01 -7.82518333e+00
   5.63888889e-01  9.44252778e-02  1.20705544e+02  3.94444444e+00
   4.51904722e-01  6.19444444e-01]
 [ 3.51931923e-01  5.45038462e-01  5.96361538e-01  6.56582615e+05
   3.90602581e-01  4.61538462e+00  2.87357692e-01 -1.08815000e+01
   4.61538462e-01  6.28961538e-02  1.07958962e+02  3.92307692e+00
   4.47061538e-01  7.30769231e-01]]
```

[116]:
```
plt.figure (figsize=(15,12))
plot = plt.axes(projection = '3d')

seaborn_plot = plt.axes (projection='3d')
print (type (seaborn_plot))
seaborn_plot.scatter3D (A["danceability"], A["energy"], A["acousticness"])
plt.show()
```

```
<class 'mpl_toolkits.mplot3d.axes3d.Axes3D'>
```


1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0