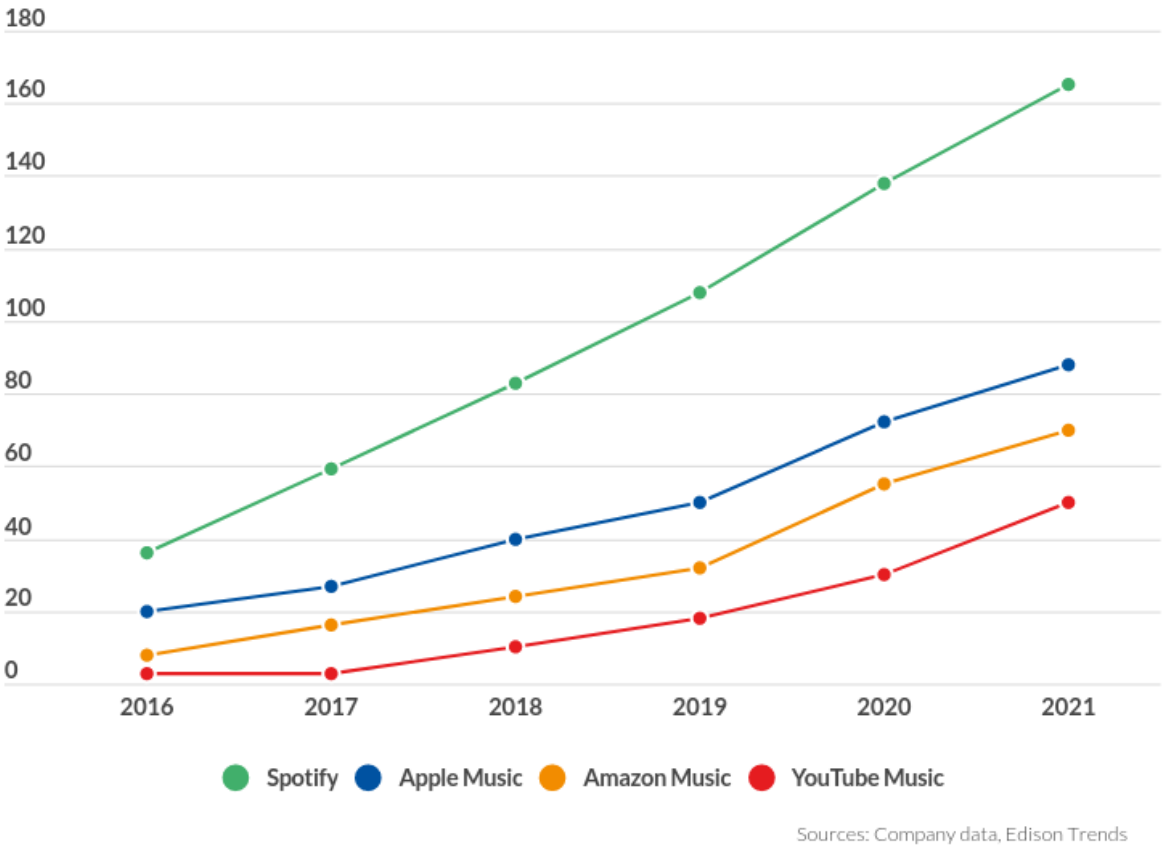
Laporan Proyek Akhir KASDD - Kelompok NumPy

Spotify Songs

Piawai Said Umbara (1806235933), Arkha Sayoga Mayadi (1606890252), Mohammad Dwikurnia Apriadi (1606889490), Fadhil Sulthoni (1606918572)

1. Latar Belakang

Saat ini musik telah menjadi industri yang sangat menjanjikan dengan memberikan ketenaran dan profit yang besar. Terlebih saat ini sudah mulai bermunculan media layanan mendengarkan musik online seperti Play Music, ITunes, Spotify, SoundCloud. Saat ini Spotify menjadi salah satu media layanan streaming musik terbesar di dunia dengan jumlah pengguna langganan terbanyak yaitu 165 juta di tahun 2021. Oleh karena itu kami akan menggunakan data dari Spotify untuk melakukan analisa dengan mengolah daftar putar lagu terkenal oleh Spotify.



2. Tujuan

Dapat memberikan wawasan yang baik terhadap dataset yang dimiliki. Wawasan dapat berupa prediksi, clustering, visualisasi data. Diharapkan dari sini bisa dijadikan basis data oleh produsen musik ketika membuat lagu baru. Memberikan wawasan tentang artis atau lagu yang populer di tahun-tahun tertentu.

3. Dataset yang Digunakan

Dataset yang kami gunakan terdiri dari 8 kumpulan dataset (dalam bentuk file .csv). Tujuh dari delapan dataset tersebut merepresentasikan playlist yang ada di Spotify. Daftar playlist tersebut berisi kumpulan lagu paling populer dari setiap dekade. File kedelapan (top10s.csv) berisi daftar lagu dari tahun 2010 dan seterusnya yang berada di peringkat teratas Billboard. Masing-masing lagu memiliki 15 atribut berikut:

#	Attribute Name	Description
1	title	Judul lagu
2	artist	Nama penyanyi
3	genre	Genre lagu
4	year	Tahun rilis
5	bpm	Jumlah beats per menit
6	nrgy	Enerji yang terkandung di sebuah lagu. Semakin tinggi nilainya, semakin enerjik lagu tersebut.
7	dnce	Semakin tinggi nilainya, semakin mudah untuk menari mengikuti alunan lagu tersebut.
8	dB	Semakin tinggi nilainya, semakin keras lagu tersebut (dalam satuan dB).
9	live	Semakin tinggi nilainya, semakin besar kemungkinan lagu tersebut adalah rekaman langsung (live).
10	val	Semakin tinggi nilainya, semakin positif mood yang terkandung dalam lagu tersebut.
11	dur	Durasi lagu
12	acous	Semakin tinggi nilainya, semakin akustik lagu tersebut.
13	spch	Semakin tinggi nilainya, semakin banyak jumlah kata yang terkandung dalam lagu tersebut.
14	popularity	Semakin tinggi nilainya, semakin populer lagu tersebut.
15	has_win_award	Nilai boolean dalam atribut ini menyatakan apakah lagu tersebut pernah memenangi suatu award. Jika lagu tersebut pernah memenangi suatu award atribut ini akan bernilai 1. Sebaliknya, jika lagu tersebut tidak pernah memenangi suatu award, atribut ini akan bernilai 0,

Dataset tersebut dapat diakses melalui tautan berikut:
https://drive.google.com/drive/folders/1Zaw2DFOL6c0Df7ySRv0YmIRJOyQtbdvM?usp=sharing

4. Task yang akan Dikerjakan

- **Exploratory Data Analysis & Visualization**

  Setelah melakukan preprocessing pada seluruh dataset, dilakukan sejumlah exploratory data analysis untuk menjawab beberapa pertanyaan terkait dataset berikut:

  - Siapa saja artist (penyanyi) yang paling populer?
  - Genre lagu apa yang paling terkenal pada era-era tertentu?
  - Apakah setiap artist memiliki suatu genre lagu spesifik?
  - Apakah fitur-fitur yang mendeskripsikan sebuah lagu memiliki suatu keterhubungan?
  - Bagaimana pengaruh fitur durasi waktu ke fitur popularitas lagu? Apakah terdapat durasi tertentu yang dapat membuat popularitas lagu semakin tinggi?

- **Clustering**

  Pengelompokkan lagu menjadi beberapa cluster berdasarkan fitur yang cocok dengan dataset yang digunakan. Lalu, melakukan analisis berdasarkan hasil clustering yang didapat (apakah cluster yang dihasilkan dapat merepresentasikan lagu berdasarkan suatu fakta?).

- **Klasifikasi**

  Melakukan klasifikasi genre lagu berdasarkan fitur yang cocok dengan dataset yang digunakan.

- **Regresi**

  Membuat model regresi untuk memprediksi popularitas dari sebuah lagu.

5. Langkah-langkah Pengerjaan Proyek

- Preprocessing
  - Mencari nilai yang hilang
  - Mencari duplikasi data
  - Melakukan normalisasi data
  - Mencari nilai outlier
- Visualisasi
  - Melakukan visualisasi data berdasarkan tren tahun atau genre atau musisi
  - Melakukan visualisasi berdasarkan data hasil clustering atau klasifikasi
- Clustering, Klasifikasi, Prediksi
  - Melakukan implementasi algoritma clustering dengan K-Means
  - Melakukan implementasi algoritma K-Neighbour untuk klasifikasi
  - Melakukan implementasi algoritma Linear Regression untuk melakukan prediksi

6. Timeline Pengerjaan Proyek

	4-10 April 2022	11-17 April 2022	18-24 April 2022	25 April-2 Mei 2022
EDA				
Clustering				
Klasifikasi				
Regresi				

7. Hasil

- **Exploratory Data Analysis & Visualization**

  - Data Preprocessing

    Menggabungkan seluruh dataset

```
df_all = pd.concat(\
    [df_1950,df_1960\
        ,df_1970,df_1980\
        ,df_1990,df_2000\
        ,df_2010,df_top10],ignore_index=True, sort=False)

df_all.to_csv('all_sort_from_1950_to_top10.csv',index=False)

# drop duplicate
df_all = df_all.drop_duplicates()
# drop all nan
df_clean = df_all.dropna()
# ubah data has_win_award dari float ke boolean
df_all = df_all.astype({"has_win_award":bool})
df_clean = df_clean.astype({"has_win_award":bool})

df_all.to_csv('all_no_duplicates.csv',index=False)
df_clean.to_csv('all_clean.csv',index=False)
```

Helper function untuk mengkategorisasi suatu genre ke dalam super-genrenya

```
def super_genre_helper(genre):
    genre_dict = {'blues': ['blues','african blues','blues rock','blues shouter','british blues','canadian blues','chicago blues','classic fema
        'country': ['country','alternative country','cowpunk','americana','australian country','bakersfield sound','bluegrass','progressive blu
        'easy listening': ['easy listening','background music','elevator music (muzak)','barococo','beautiful music','chill-out','furniture mus
        'electronic': ['electronic','ambient','ambient dub','dark ambient','ambient industrial','dungeon synth','isolationism','drone','illbien
        'contemporary folk': ['contemporary folk','american folk revival','americana','anti-folk','british folk revival','celtic music','chalga
        'hip hop': ['hip hop','alternative hip hop','hipster hop','boom bap','bounce','british hip hop','road rap','chopped and screwed','chopp
        'jazz': ['jazz','acid jazz','afro-cuban jazz','alt-jazz','avant-garde jazz','bebop','boogie-woogie','bossa nova','brazilian jazz','brit
        'pop': ['pop','adult album alternative','adult contemporary','ambient pop','arabic pop','art pop','baroque pop','bedroom pop','brill bu
        'r&b and soul': ['r&b and soul','alternative r&b','contemporary r&b','disco','freestyle','go-go','funk','deep funk','minneapolis sound'
        'rock': ['rock','afro rock','alternative rock','alternative dance','britpop','post-britpop','dream pop','goth rock','shoegaze','blackga
        'metal': ['metal','alternative metal','funk metal','nu metal','rap metal','avant-garde metal','black metal','blackened death metal','at
        'punk': ['punk','anarcho punk','crust punk','d-beat','art punk','christian punk','deathrock','digital hardcore','folk punk','celtic pun
    genre_list = ''
    found = False
    for k,v in genre_dict.items():
        if genre in v:
            if genre_list == '': genre_list+=k
            else: genre_list += '|'+k
            found = True
    if not found: return genre
    return genre_list

def super_genre(df):
    df_tmp = df.copy()
    for i in tqdm.tqdm(df_tmp.index):
        df_tmp.loc[i,'genre'] = super_genre_helper(df_tmp.loc[i,'genre'])
    return df_tmp

genre_all = ['blues','country','easy listening','electronic','contemporary folk','hip hop','jazz','pop','r&b and soul','rock','metal','punk']
```

Eksekusi fungsi super_genre_helper pada dataset

```
df_all_super_genre = super_genre(df_clean)

multi_genre = df_all_super_genre.loc[df_all_super_genre['genre'].str.contains('\|', case=False)]
df_all_super_genre = df_all_super_genre[df_all_super_genre["genre"].str.contains('\|') == False]

for i, r in tqdm.tqdm(multi_genre.iterrows()):
    genre_list = r['genre'].split('|')
    new_row = r.to_dict()
    for g in genre_list:
        new_row['genre'] = g
        df_all_super_genre = df_all_super_genre.append(new_row, ignore_index=True)
```

Super-genre yang terdapat pada dataset

In [9]:
```
df_all_super_genre.to_csv('all_super_genre.csv',index=False)
df_all_super_genre['genre'].unique()
```

Out[9]:
```
array(['pop', 'r&b and soul', 'blues', 'jazz', 'rock',
       'contemporary folk', 'country', 'electronic', 'metal', 'hip hop'],
      dtype=object)
```

Memeriksa apakah terdapat duplikasi data pada dataset

In [10]:
```
# cek duplikasi
print(sum(df_all.duplicated()))
```

```
0
```

Memeriksa apakah terdapat missing value pada dataset

In [11]:
```
# cek null di kolom yang mana
def cek_apakah_ada_null(data_frame):
    cek = data_frame.isnull()
    columns = list(cek.columns)
    c_null = []
    for i in columns:
        if sum(cek[i]) != 0:
            c_null.append(i)
    return c_null
```

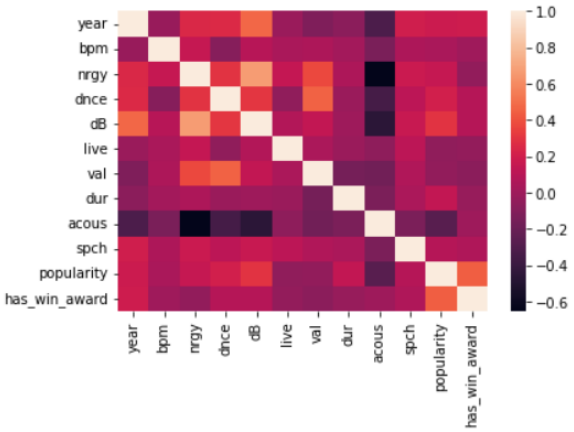
In [12]:
```
print(cek_apakah_ada_null(df_all))
```

```
['genre']
```

Memeriksa korelasi antar atribut pada dataset

In [13]:
```
sns.heatmap(df_all.corr())
```

Out[13]:
```
<AxesSubplot:>
```



Hanya energy, dB, dan acoustic saja yang dapat mendeskripsikan sebuah lagu

In [14]:
```
attribute_spotify_high_corr = ["nrgy","dB","acous"]
```

Berdasarkan visualisasi Heatmap diketahui bahwa hanya tiga atribut saja yang memiliki korelasi tertinggi (energy dengan acoustic dan dB dengan acoustic).

Setelah melakukan preprocessing pada seluruh dataset, dilakukan sejumlah exploratory data analysis untuk menjawab beberapa pertanyaan terkait dataset berikut:

- Siapa saja artist (penyanyi) yang paling populer?

  Mencari daftar artis yang populer berdasarkan jumlah skor popularitas untuk setiap lagu dari artis tersebut.

Artis yang populer

In [22]:
```
sum(df_top10.duplicated())
```

Out[22]:
```
0
```

In [23]:
```
artist_total = pd.unique(df_all.loc[:,'artist'])
print("Total Artist : %d" % (len(artist_total)))
```

```
Total Artist : 589
```

In [24]:
```
genre_total = pd.unique(df_all.loc[:,'genre'])
print("Total Genre : %d" % (len(genre_total)))
```

```
Total Genre : 137
```

In [25]:
```
df_all['artist'].value_counts()
```

Out[25]:
```
Rihanna              21
Katy Perry           20
Justin Bieber        18
Maroon 5             16
Bruno Mars           15
                     ..
Van Halen             1
Eric Carmen           1
Michael Sembello      1
Bruce Springsteen     1
R3HAB                 1
Name: artist, Length: 589, dtype: int64
```

Artis yang paling populer dapat dihitung dari jumlah nilai lagu yang populer

In [26]:
```
def dict_artist_popularity(df):
    artist_popularity = dict()
    for i in df.index:
        artist = df['artist'][i]
        popularity = df['popularity'][i]
        artist_popularity[artist] = artist_popularity.setdefault(artist,0)\
            + popularity
    return artist_popularity
```

In [27]:
```
artis_populer = dict_artist_popularity(df_all.loc[df_all['year'] == 2010])
max(artis_populer, key=artis_populer.get)
```

Out[27]:
```
'Bruno Mars'
```

Berdasarkan eksekusi fungsi dict_artist_popularity (fungsi untuk menghitung skor popularitas lagu dari tiap artis) diketahui bahwa artis yang mendapatkan skor popularitas paling tinggi adalah **Bruno Mars**.

- Genre lagu apa yang paling terkenal pada era-era tertentu?

  Mencari genre lagu yang populer berdasarkan skor popularitas untuk setiap lagu dari genre tersebut.

Mencari genre lagu yang populer dapat menggunakan hal yang sama saat mencari artis

In [28]:
```
def dict_genre_popularity(df):
    genre_popularity = dict()
    for i in df.index:
        genre = df['genre'][i]
        popularity = df['popularity'][i]
        genre_popularity[genre] = genre_popularity.setdefault(genre,0)\
            + popularity
    return genre_popularity
```

In [29]:
```
genre_populer = dict_genre_popularity(df_top10[df_top10['year'] == 2010])
max(genre_populer, key=genre_populer.get)
```

Out[29]:
```
'dance pop'
```

Berdasarkan eksekusi fungsi dict_genre_popularity (fungsi untuk menghitung skor popularitas lagu dari tiap genre) diketahui bahwa genre yang mendapatkan skor popularitas paling tinggi adalah **dance pop**.

- Apakah setiap artist memiliki suatu genre lagu spesifik?

  Fungsi yang menghasilkan output dictionary setiap artis dengan kumpulan genre dari lagu yang mereka bawakan.

```
def dict_artist_genre(df):
    artist_genre = dict()
    for i in df.index:
        artist = df['artist'][i]
        genre = df['genre'][i]
        if isNaN(genre):
            artist_genre.setdefault(artist,set())
            continue
        artist_genre.setdefault(artist,set()).add(genre)
    return artist_genre
```

  Fungsi yang menghasilkan output dictionary setiap artis dengan kumpulan lagu yang mereka bawakan.

```
def dict_artist_music(df):
    artist_music = dict()
    for i in df.index:
        artist = df['artist'][i]
        music = df['title'][i]
        if isNaN(music):
            artist_music.setdefault(artist,set())
            continue
        artist_music.setdefault(artist,set()).add(music)
    return artist_music
```

  Fungsi yang menghasilkan output dictionary setiap lagu dengan kumpulan genrenya.

```
def dict_genre_music(df):
    genre_music = dict()
    for i in df.index:
        genre = df['genre'][i]
        music = df['title'][i]
        if isNaN(music):
            genre_music.setdefault(genre,set())
            continue
        genre_music.setdefault(genre,set()).add(music)
    return genre_music
```

Mencari jumlah genre maksimum dari seorang artis di dataset.

In [33]:
```
artist_genre = dict_artist_genre(df_all)

for i in artist_genre.keys():
    artist_genre[i] = len(artist_genre[i])

genre_max = max(artist_genre.values())
genre_max
```

Out[33]: 2

Berdasarkan output dari code di atas, diketahui bahwa jumlah genre terbanyak dari seorang artis adalah 2.

In [34]:
```
music = dict_artist_music(df_all)
df_all.loc[df_all['artist'] == 'Taylor Swift']
```

Out[34]:

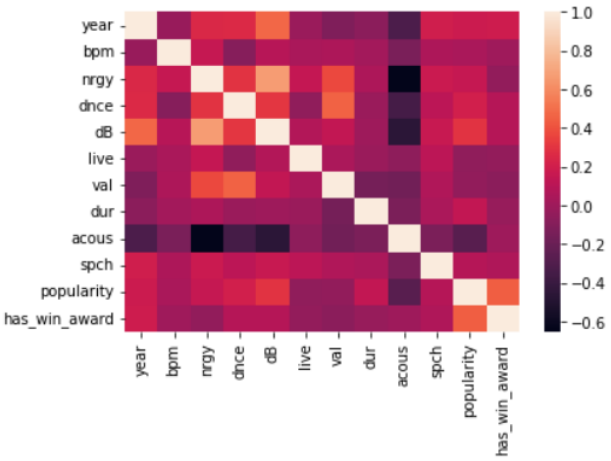
	title	artist	genre	year	bpm	nrgy	dnce	dB	live	val	dur	acous	spch	popularity	has_win_award
494	I Knew You Were Trouble.	Taylor Swift	dance pop	2012	77	47	62	-7	3	68	220	0	4	75	False
508	We Are Never Ever Getting Back Together	Taylor Swift	dance pop	2012	86	68	63	-6	12	75	193	1	9	73	False
509	Love Story	Taylor Swift	dance pop	2008	119	74	62	-4	8	31	236	13	3	73	False
552	Teardrops On My Guitar - Radio Single Remix	Taylor Swift	dance pop	2006	100	42	62	-7	12	29	203	29	2	61	False
645	Wildest Dreams	Taylor Swift	dance pop	2014	140	66	55	-7	11	47	220	7	7	71	False
776	I Knew You Were Trouble.	Taylor Swift	pop	2012	77	47	62	-7	3	68	220	0	4	77	False
780	We Are Never Ever Getting Back Together	Taylor Swift	pop	2012	86	68	63	-6	12	75	193	1	9	75	False
819	We Are Never Ever Getting Back Together	Taylor Swift	pop	2013	86	68	63	-6	12	75	193	1	9	75	False
882	Shake It Off	Taylor Swift	pop	2014	160	80	65	-5	33	94	219	6	17	78	False
1074	Out Of The Woods	Taylor Swift	pop	2016	92	84	55	-7	34	34	236	0	4	66	False
1204	Look What You Made Me Do	Taylor Swift	pop	2018	128	71	77	-6	13	51	212	20	12	75	False
1216	End Game	Taylor Swift	pop	2018	159	59	65	-6	11	15	245	1	6	70	False
1236	...Ready For It? - BloodPop® Remix	Taylor Swift	pop	2018	160	84	58	-5	10	50	190	13	22	52	False

Contoh output code untuk genre dari setiap lagu yang dibawakan oleh Taylor Swift.

- Apakah fitur-fitur yang mendeskripsikan sebuah lagu memiliki suatu keterhubungan?

```
In [13]: sns.heatmap(df_all.corr())
```

```
Out[13]: <AxesSubplot:>
```



Hanya energy, dB, dan acoustic saja yang dapat mendeskripsikan sebuah lagu

```
In [14]: attribute_spotify_high_corr = ["nrgy","dB","acous"]
```

Seperti yang sudah dijelaskan sebelumnya, melalui visualisasi Heatmap diketahui bahwa hanya tiga atribut saja yang memiliki korelasi tertinggi (energy dengan acoustic dan dB dengan acoustic).

- Bagaimana pengaruh fitur durasi waktu ke fitur popularitas lagu? Apakah terdapat durasi tertentu yang dapat membuat popularitas lagu semakin tinggi?

Out[35]:

	dur	popularity
dur	1.00000	0.13945
popularity	0.13945	1.00000

Berdasarkan koefisien korelasi Pearson dari atribut duration dan popularity, diketahui bahwa korelasi kedua atribut tersebut sangat lemah (weak correlation).

- **Clustering**

Scaling data menggunakan min-max scaler.

```
# buat scaler
def scaler_spotify(df, attribute_spotify):
    sc = MinMaxScaler()
    data_scaled = sc.fit_transform(df.loc[:,attribute_spotify])
    return data_scaled

def scaler_spotify_with_PCA(df, attribute_spotify):
    sc = MinMaxScaler()
    data_scaled = sc.fit_transform(df.loc[:,attribute_spotify])
    pca = PCA(n_components=0.95)
    pca.fit(data_scaled)
    data_scaled = pca.transform(data_scaled)
    return data_scaled
```

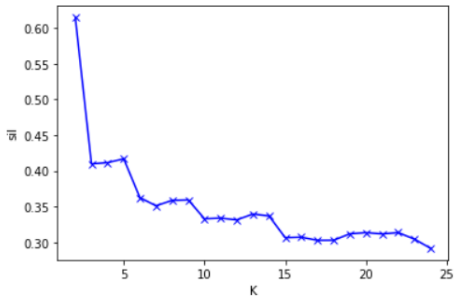
Mencari nilai k optimal menggunakan elbow method.

```
# menggunakan clustering kmeans
# mencari nilai optimal dari k
def optimal_kmeans(scaled_data):
    sil = []
    k = range(2,25)
    for i in k:
        cluster_data = KMeans(n_clusters=i)\
            .fit(scaled_data)
        sil.append(silhouette_score(scaled_data\
            ,cluster_data.labels_,metric='euclidean'))
    plt.plot(k,sil, "bx-")
    plt.xlabel("K")
    plt.ylabel("sil")
    plt.show()
    return sil.index(max(sil))+2
```

Eksekusi fungsi optimal_kmeans pada dataset (didapat nilai k optimal = 2)

```
attribute_spotify = ['title','artist','genre','year','bpm','nrgy','dnce','dB','live','val','dur','acous','spch','popularity','has_win_award']
```

```
df_all_std = scaler_spotify(df_all, attribute_spotify_high_corr)
k = optimal_kmeans(df_all_std)
df_all_cd = KMeans(n_clusters=k).fit_predict(df_all_std)
df_all_cluster = df_all.copy()
df_all_cluster['cluster'] = df_all_cd
print(k)
```




```
2
```

Silhoutte score dari clustering yang telah dilakukan.

In [19]:
```
from sklearn.metrics import silhouette_score
silhouette_score(df_all.loc[:,attribute_spotify_high_corr], df_all_cd, metric='euclidean')
```

Out[19]:
```
0.6203765854379533
```

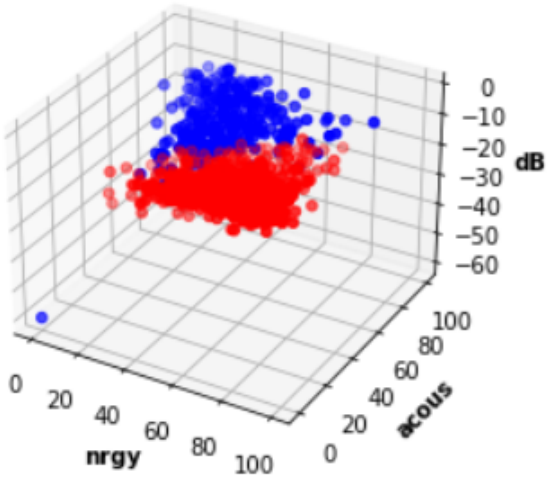
Implementasi code untuk plotting hasil clustering berdasarkan nilai k yang sebelumnya didapat dari elbow method (k = 2).

```
label_0 = df_all_cluster.loc[df_all_cluster['cluster'] == 0]
label_1 = df_all_cluster.loc[df_all_cluster['cluster'] == 1]
```

```
plot3d = plt.axes(projection='3d')
plot3d.set_xlabel('nrgy', fontweight ='bold')
plot3d.set_ylabel('acous', fontweight ='bold')
plot3d.set_zlabel('dB', fontweight ='bold')
plot3d.scatter3D(label_0['nrgy'], label_0['acous'], label_0['dB'], color='red')
plot3d.scatter3D(label_1['nrgy'], label_1['acous'], label_1['dB'], color='blue')
plt.show
```

Plot hasil clustering dari tiga atribut yang memiliki korelasi tertinggi (nrgy, acous, dan dB).

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



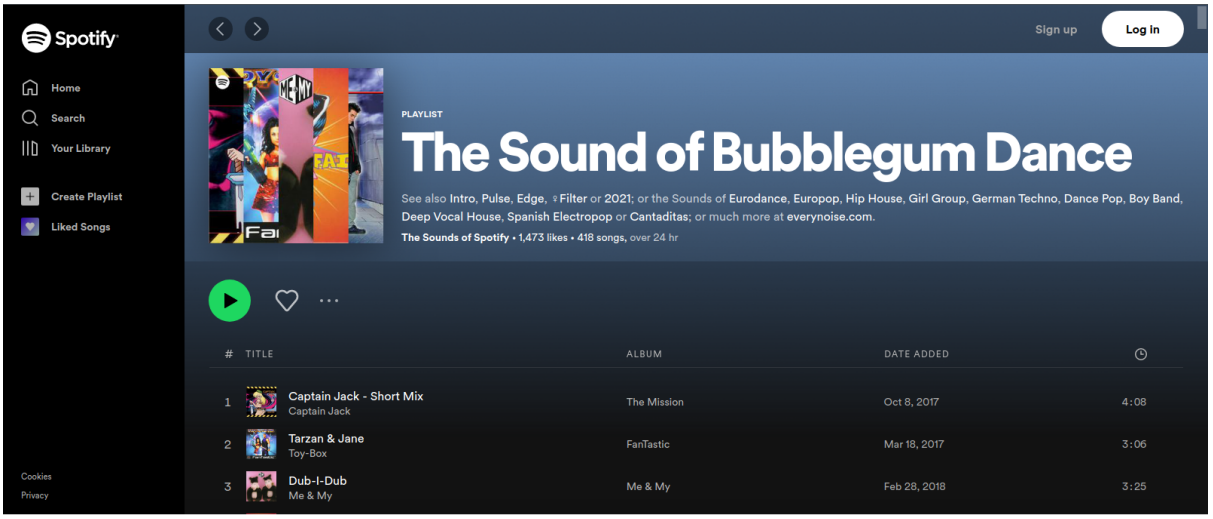
- **Klasifikasi**

  Dalam tahap klasifikasi kami melakukan 2 hal yaitu klasifikasi berdasarkan genre dan pemenang penghargaan.

  1. Genre

     Dari data genre yang ada, ada banyak jenis genre di Spotify. Oleh karena itu dilakukan konversi genre ke bentuk super genre yang didapatkan dari en.wikipedia.org/wiki/List_of_music_genres_and_styles. Ada 12 genre utama yaitu blues, country, easy listening, electronic, contemporary folk, hip hop, jazz, pop, r&b and soul, rock, metal, punk. Namun masih ada genre-genre yang belum terklasifikasi

sehingga perlu melakukan pencarian manual melalui situs spotify. Proses pencarian dilakukan dengan mencari playlist spotify dengan nama genre. Contoh saat mencari genre bubblegum dance maka melakukan pencarian playlist spotify bernama “The Sound of Bubblegum Dance”



Dari playlist ini ada informasi “or the Sounds of …” dan diikuti genre-genre yang terurut dari genre dengan keterkaitan terkuat hingga terlemah. Dalam hal ini genre bubblegum dance sangat berkaitan erat dengan eurodance yang merupakan bagian dari super genre electronic. Sehingga bubblegum dance dapat dikategorikan dalam super genre electronic.

Satu genre dapat masuk kedalam 2 kategori super genre. Contoh adalah genre country pop yang masuk dalam kategori super genre country dan pop. Untuk mengatasi hal ini data dengan double genre ini akan dilakukan penambahan data dengan data musik yang sama namun berbeda genre ke dalam database.

Hasil dari pengubahan genre ini adalah sebagai berikut.

```
blues 41
country 11
easy listening 0
electronic 128
contemporary folk 15
hip hop 62
jazz 5
pop 804
r&b and soul 93
rock 171
metal 6
punk 0
```

Musik dengan genre pop memiliki jumlah yang lebih banyak dari genre lainnya sehingga jika dilakukan klasifikasi akan terjadi klasifikasi tidak seimbang. Untuk menyeimbangkan data dilakukan oversampling menggunakan RandomOverSampler, SMOTE, dan ADASYN dengan teknik RepeatedStratifiedKFold dan DecisionTree sebagai model klasifikasinya. Dari semua metode oversampling didapatkan bahwa masih mendapatkan nilai F1 yang rendah. Hal ini dikarenakan jumlah musik dengan genre jazz dan metal sangat sedikit.

```
X = df_popular.loc[:,attribute_spotify_genre]
y = df_popular['genre']

# define pipeline
steps = [('over', ADASYN(n_neighbors=3)), ('model', DecisionTreeClassifier())]
pipeline = Pipeline(steps=steps)
# evaluate pipeline
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(pipeline, X, y, scoring='f1_micro', cv=cv, n_jobs=-1)
print('F1 Score: %.3f' % np.mean(scores))
```

```
F1 Score: 0.405
```

2. Has Win Award

   Melakukan klasifikasi musik yang sudah memiliki penghargaan atau belum. Sama seperti proses klasifikasi genre, ada ketidak seimbangan data antar data.

   ```
   has not win 1194
   has win 142
   ```

   Proses yang dilakukan kurang lebih sama saat klasifikasi genre yaitu oversampling data dengan metode RandomOverSampling, SMOTE, dan ADASYN. Dari hasil yang didapatkan ternyata cukup memuaskan dengan nilai F1 sebesar 0.98 di semua metode oversampling. Jika dibandingkan dengan data klasifikasi genre dengan penghargaan ini, banyak data dari pemenang penghargaan cukup banyak yaitu 142. Jumlah ini cukup baik untuk melakukan metode over sampling dengan berbagai metode.

```
X = df_all.loc[:,attribute_spotify_award]
y = df_all['has_win_award']

# define pipeline
steps = [('over', SMOTE()), ('model', DecisionTreeClassifier())]
pipeline = Pipeline(steps=steps)
# evaluate pipeline
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(pipeline, X, y, scoring='f1_micro', cv=cv, n_jobs=-1)
print('F1 Score: %.3f' % np.mean(scores))
```

```
F1 Score: 0.984
```

- **Regresi**

  Melakukan regresi popularitas dari musik. Ada tantangan tersendiri ketika melakukan regresi popularitas suatu musik karena dari database yang dimiliki didapatkan popularitas memiliki keterkaitan kecil dengan fitur-fitur lainnya. Hanya fitur acous, dB, dan has_win_award saja yang memiliki nilai keterkaitan yang tinggi dengan nilai 0.2. Hal ini akan berdampak saat melakukan regresi yang kurang baik. Namun sebelum dilakukan regresi, melakukan pembersihan data terlebih dahulu dengan metode interquartile range. Metode regresi yang digunakan adalah LinearRegression. Hasil yang didapatkan sesuai dengan ekspektasi yang tidak begitu baik hasil regresinya dan nilai R2 sebesar 0.266

```
X = np.array(df_popularity.loc[:,['acous','dB','has_win_award']])
y = np.array(df_popularity.loc[:,['popularity']])

linear_r = LinearRegression()
# evaluate pipeline
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(linear_r, X, y, scoring='r2', cv=cv, n_jobs=-1)
print('R2: %.3f' % np.mean(scores))
```

```
R2: 0.266
```