Data Science Intern @Lets Grow More

Author - Priyangshu Sarkar

Task 3 -Music recommender system

--Import Libraries

In [46]:
```
import os
import numpy as np
import pandas as pd

import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.metrics import euclidean_distances
from scipy.spatial.distance import cdist

import warnings
warnings.filterwarnings("ignore")
```

In [47]:
```
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\abc\anaconda3\lib\site-packages (5.8.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\abc\anaconda3\lib\site-packages (from plotly) (8.0.1)
```

Read Data

In [48]:
```
df=pd.read_csv("data.csv")
df1=pd.read_csv("data_by_genres.csv")
df2=pd.read_csv("data_by_year.csv")
```

In [49]:
```
df
```

Out[49]:

	valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key
0	0.0594	1921	0.98200	['Sergei Rachmaninoff', 'James Levine', 'Berli...	0.279	831667	0.211	0	4BJqT0PrAfrxzMOxytFOIz	0.878000	10
1	0.9630	1921	0.73200	['Dennis Day']	0.819	180533	0.341	0	7xPhfUan2yNtyFG0cUWkt8	0.000000	7
2	0.0394	1921	0.96100	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi...	0.328	500062	0.166	0	1o6I8BglA6ylDMrIELygv1	0.913000	3
3	0.1650	1921	0.96700	['Frank Parker']	0.275	210000	0.309	0	3ftBPsC5vPBKxYSee08FDH	0.000028	5
4	0.2530	1921	0.95700	['Phil Regan']	0.418	166693	0.193	0	4d6HGyGT8e121BsdKmw9v6	0.000002	3
...	...	...	...	...	...	...	...	...	...	...	...
170648	0.6080	2020	0.08460	['Anuel AA', 'Daddy Yankee', 'KAROL G', 'Ozuna...	0.786	301714	0.808	0	0KkIkfsLEJbrcIhYsCL7L5	0.000289	7
170649	0.7340	2020	0.20600	['Ashnikko']	0.717	150654	0.753	0	0OStKKAuXlxA0fMH54Qs6E	0.000000	7
170650	0.6370	2020	0.10100	['MAMAMOO']	0.634	211280	0.858	0	4BZXVFYCb76Q0Klojq4piV	0.000009	4
170651	0.1950	2020	0.00998	['Eminem']	0.671	337147	0.623	1	5SiZJoLXp3WOl3J4C8IK0d	0.000008	2
170652	0.6420	2020	0.13200	['KEVVO', 'J Balvin']	0.856	189507	0.721	1	7HmnJHfs0BkFzX4x8j0hkl	0.004710	7

170653 rows × 19 columns

In [50]:
```
df1
```

Out[50]:

	mode	genres	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valenc
0	1	21st century classical	0.979333	0.162883	1.602977e+05	0.071317	0.606834	0.361600	-31.514333	0.040567	75.336500	0.10378
1	1	432hz	0.494780	0.299333	1.048887e+06	0.450678	0.477762	0.131000	-16.854000	0.076817	120.285667	0.22175
2	1	8-bit	0.762000	0.712000	1.151770e+05	0.818000	0.876000	0.126000	-9.180000	0.047000	133.444000	0.97500
3	1	[]	0.651417	0.529093	2.328809e+05	0.419146	0.205309	0.218696	-12.288965	0.107872	112.857352	0.51360
4	1	a cappella	0.676557	0.538961	1.906285e+05	0.316434	0.003003	0.172254	-12.479387	0.082851	112.110362	0.44824
...	...	...	...	...	...	...	...	...	...	...	...	.
2968	1	zolo	0.222625	0.547082	2.580991e+05	0.610240	0.143872	0.204206	-11.295878	0.061088	125.494919	0.59615
2969	0	zouglou	0.161000	0.863000	2.063200e+05	0.909000	0.000000	0.108000	-5.985000	0.081300	119.038000	0.84500
2970	1	zouk	0.263261	0.748889	3.060728e+05	0.622444	0.257227	0.089678	-10.289222	0.038778	101.965222	0.82411
2971	0	zurich indie	0.993000	0.705667	1.984173e+05	0.172667	0.468633	0.179667	-11.453333	0.348667	91.278000	0.73900
2972	1	zydeco	0.421038	0.629409	1.716717e+05	0.609369	0.019248	0.255877	-9.854825	0.050491	126.366087	0.80854

2973 rows × 14 columns

In [51]:
```
df2
```

Out[51]:

	mode	year	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	p
0	1	1921	0.886896	0.418597	260537.166667	0.231815	0.344878	0.205710	-17.048667	0.073662	101.531493	0.379327	
1	1	1922	0.938592	0.482042	165469.746479	0.237815	0.434195	0.240720	-19.275282	0.116655	100.884521	0.535549	
2	1	1923	0.957247	0.577341	177942.362162	0.262406	0.371733	0.227462	-14.129211	0.093949	114.010730	0.625492	
3	1	1924	0.940200	0.549894	191046.707627	0.344347	0.581701	0.235219	-14.231343	0.092089	120.689572	0.663725	
4	1	1925	0.962607	0.573863	184986.924460	0.278594	0.418297	0.237668	-14.146414	0.111918	115.521921	0.621929	
...	...	...	...	...	...	...	...	...	...	...	...	...	
95	1	2016	0.284171	0.600202	221396.510295	0.592855	0.093984	0.181170	-8.061056	0.104313	118.652630	0.431532	5
96	1	2017	0.286099	0.612217	211115.696787	0.590421	0.097091	0.191713	-8.312630	0.110536	117.202740	0.416476	6
97	1	2018	0.267633	0.663500	206001.007133	0.602435	0.054217	0.176326	-7.168785	0.127176	121.922308	0.447921	6
98	1	2019	0.278299	0.644814	201024.788096	0.593224	0.077640	0.172616	-7.722192	0.121043	120.235644	0.458818	6
99	1	2020	0.219931	0.692904	193728.397537	0.631232	0.016376	0.178535	-6.595067	0.141384	124.283129	0.501048	6

100 rows × 14 columns

In [52]:
```python
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170653 entries, 0 to 170652
Data columns (total 19 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   valence           170653 non-null  float64
 1   year              170653 non-null  int64  
 2   acousticness      170653 non-null  float64
 3   artists           170653 non-null  object 
 4   danceability      170653 non-null  float64
 5   duration_ms       170653 non-null  int64  
 6   energy            170653 non-null  float64
 7   explicit          170653 non-null  int64  
 8   id                170653 non-null  object 
 9   instrumentalness  170653 non-null  float64
 10  key               170653 non-null  int64  
 11  liveness          170653 non-null  float64
 12  loudness          170653 non-null  float64
 13  mode              170653 non-null  int64  
 14  name              170653 non-null  object 
 15  popularity        170653 non-null  int64  
 16  release_date      170653 non-null  object 
 17  speechiness       170653 non-null  float64
 18  tempo             170653 non-null  float64
dtypes: float64(9), int64(6), object(4)
memory usage: 24.7+ MB
None
```

In [53]:
```python
print(df1.info())
```

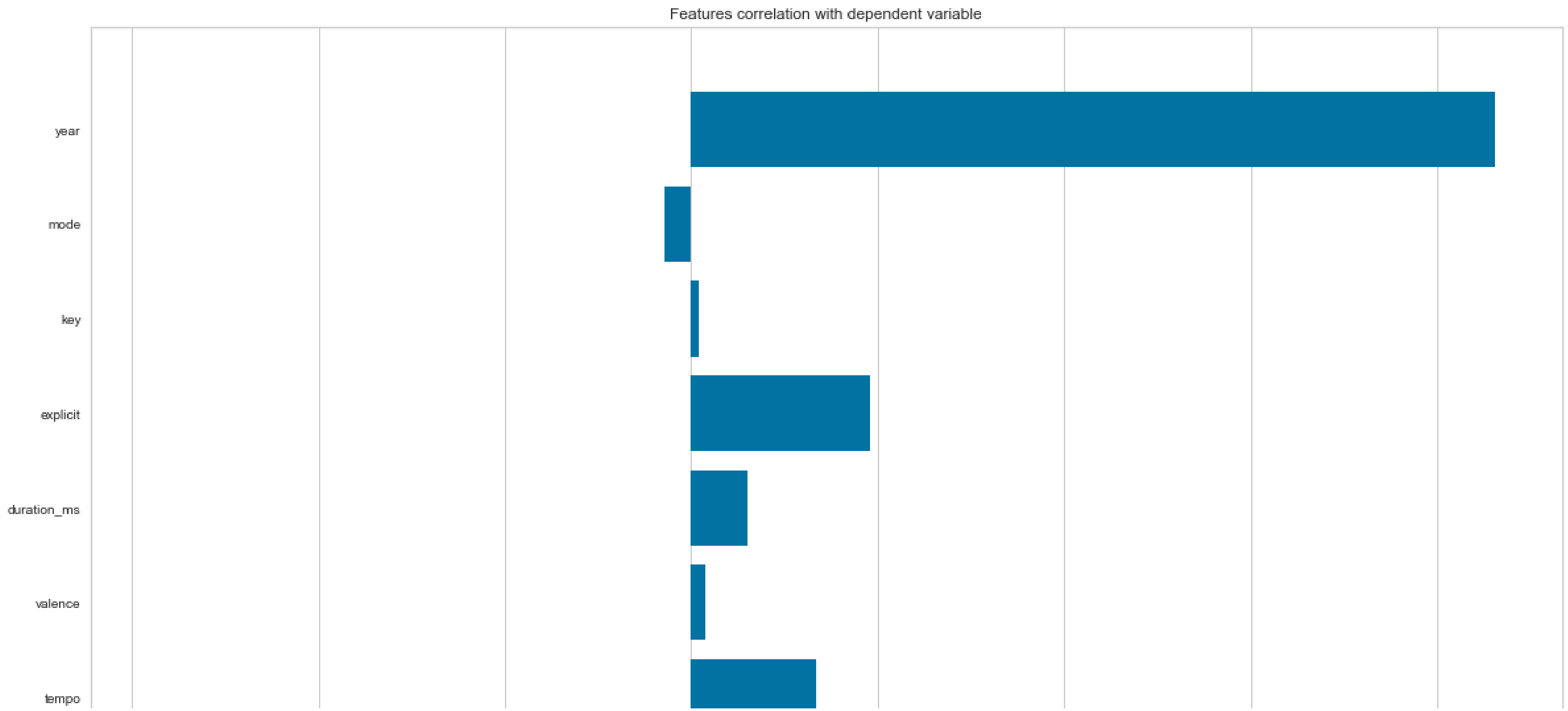
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2973 entries, 0 to 2972
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   mode              2973 non-null   int64  
 1   genres            2973 non-null   object 
 2   acousticness      2973 non-null   float64
 3   danceability      2973 non-null   float64
 4   duration_ms       2973 non-null   float64
 5   energy            2973 non-null   float64
 6   instrumentalness  2973 non-null   float64
 7   liveness          2973 non-null   float64
 8   loudness          2973 non-null   float64
 9   speechiness       2973 non-null   float64
 10  tempo             2973 non-null   float64
 11  valence           2973 non-null   float64
 12  popularity        2973 non-null   float64
 13  key               2973 non-null   int64  
dtypes: float64(11), int64(2), object(1)
memory usage: 325.3+ KB
None
```

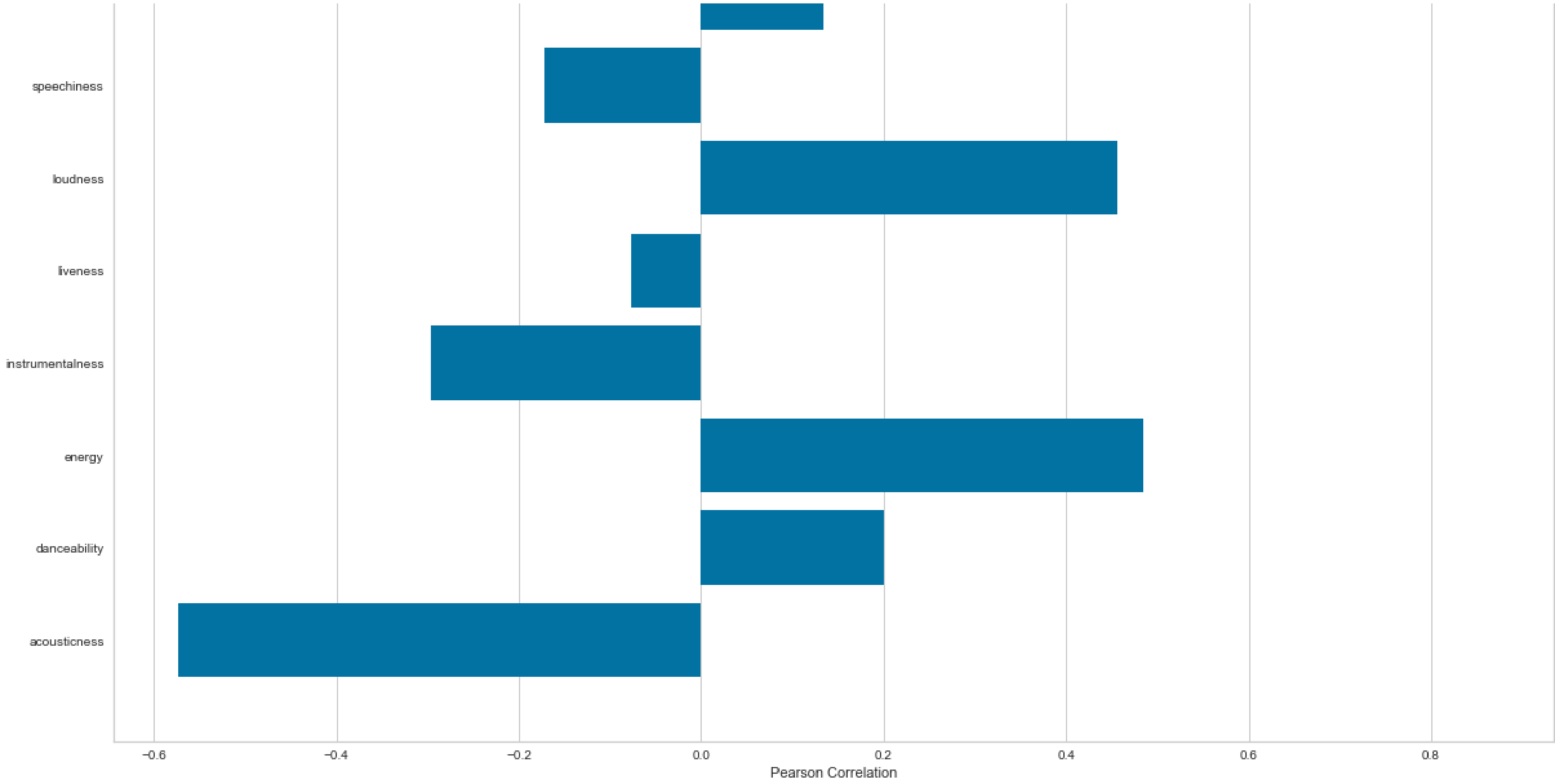
In [54]:
```
print(df2.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   mode              100 non-null    int64  
 1   year              100 non-null    int64  
 2   acousticness      100 non-null    float64
 3   danceability      100 non-null    float64
 4   duration_ms       100 non-null    float64
 5   energy            100 non-null    float64
 6   instrumentalness  100 non-null    float64
 7   liveness          100 non-null    float64
 8   loudness          100 non-null    float64
 9   speechiness       100 non-null    float64
 10  tempo             100 non-null    float64
 11  valence           100 non-null    float64
 12  popularity        100 non-null    float64
 13  key               100 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 11.1 KB
None
```

In [58]:

```
from yellowbrick.target import FeatureCorrelation

feature_names = ['acousticness', 'danceability', 'energy', 'instrumentalness',
       'liveness', 'loudness', 'speechiness', 'tempo', 'valence','duration_ms','explicit','key','mode','year']

X, y = df[feature_names], df['popularity']

# Create a list of the feature names
features = np.array(feature_names)

# Instantiate the visualizer
visualizer = FeatureCorrelation(labels=features)

plt.rcParams['figure.figsize']=(20,20)
visualizer.fit(X, y)     # Fit the data to the visualizer
visualizer.show()
```





Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x15f8d7afc40>

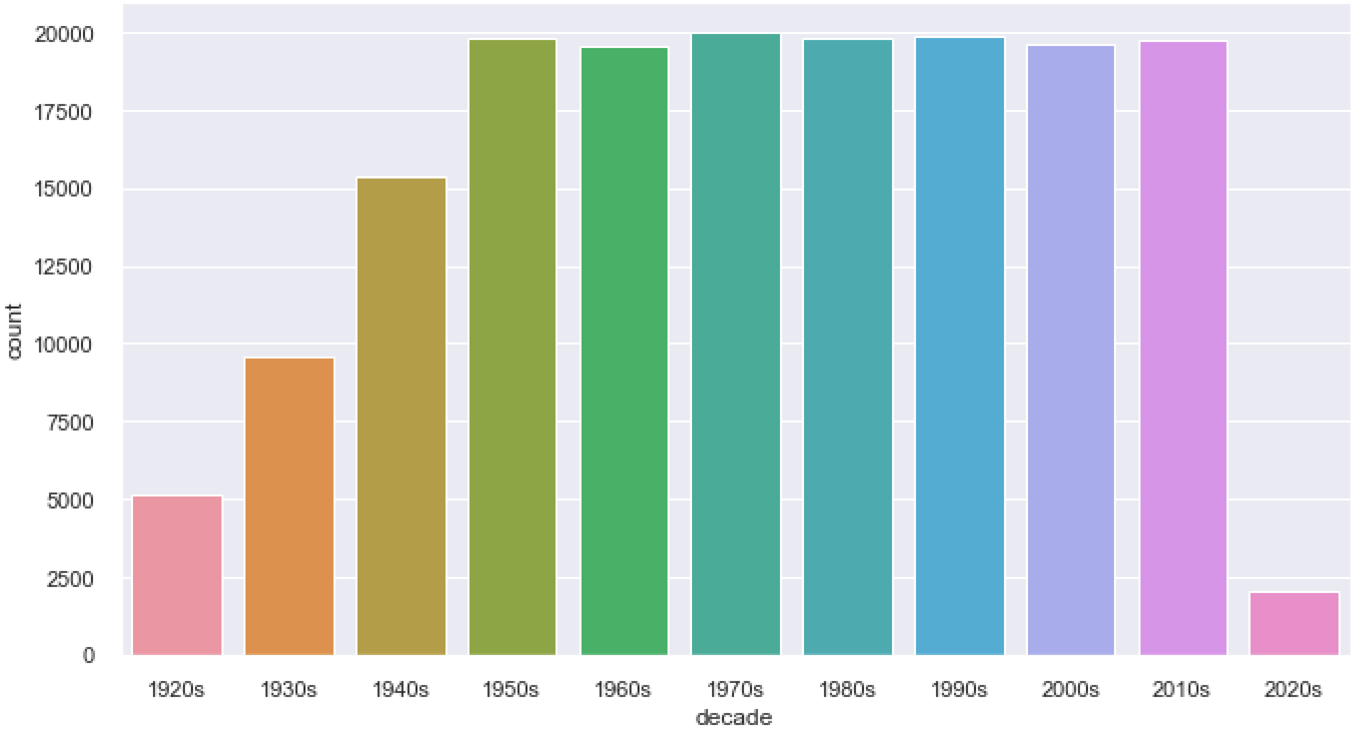
Data Understanding by Visualization and EDA

Music Over Time

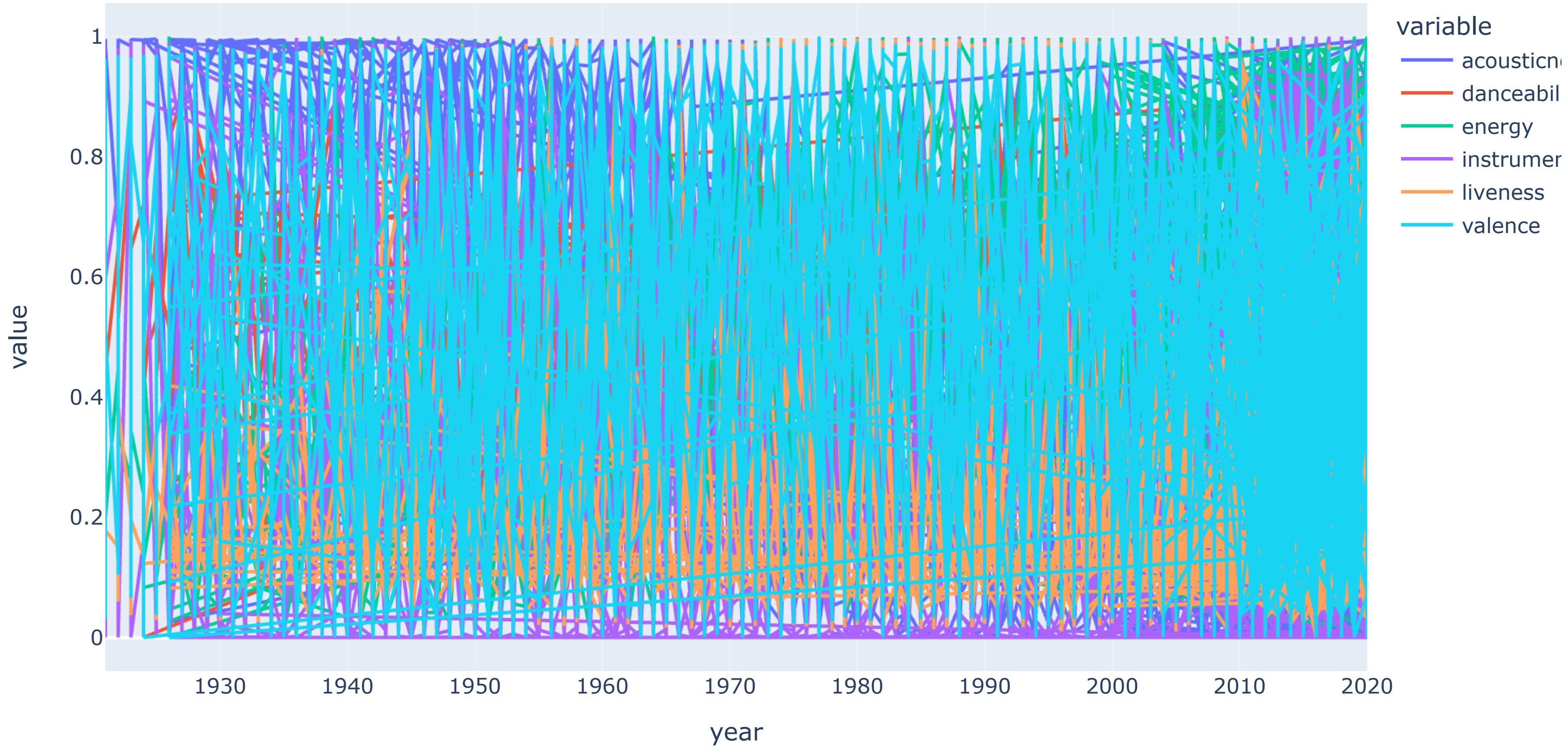
Using the data grouped by year, we can understand how the overall sound of music has changed from 1921 to 2020.

In [59]:
```python
def get_decade(year):
    period_start = int(year/10) * 10
    decade = '{}s'.format(period_start)
    return decade

df['decade'] = df['year'].apply(get_decade)

sns.set(rc={'figure.figsize':(11 ,6)})
sns.countplot(df['decade'])
```

Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x15f91ff3ee0>



In [62]:
```
sound_features = ['acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'valence']
fig = px.line(df, x='year', y=sound_features)
fig.show()
```



Clustering Genres with K-Means

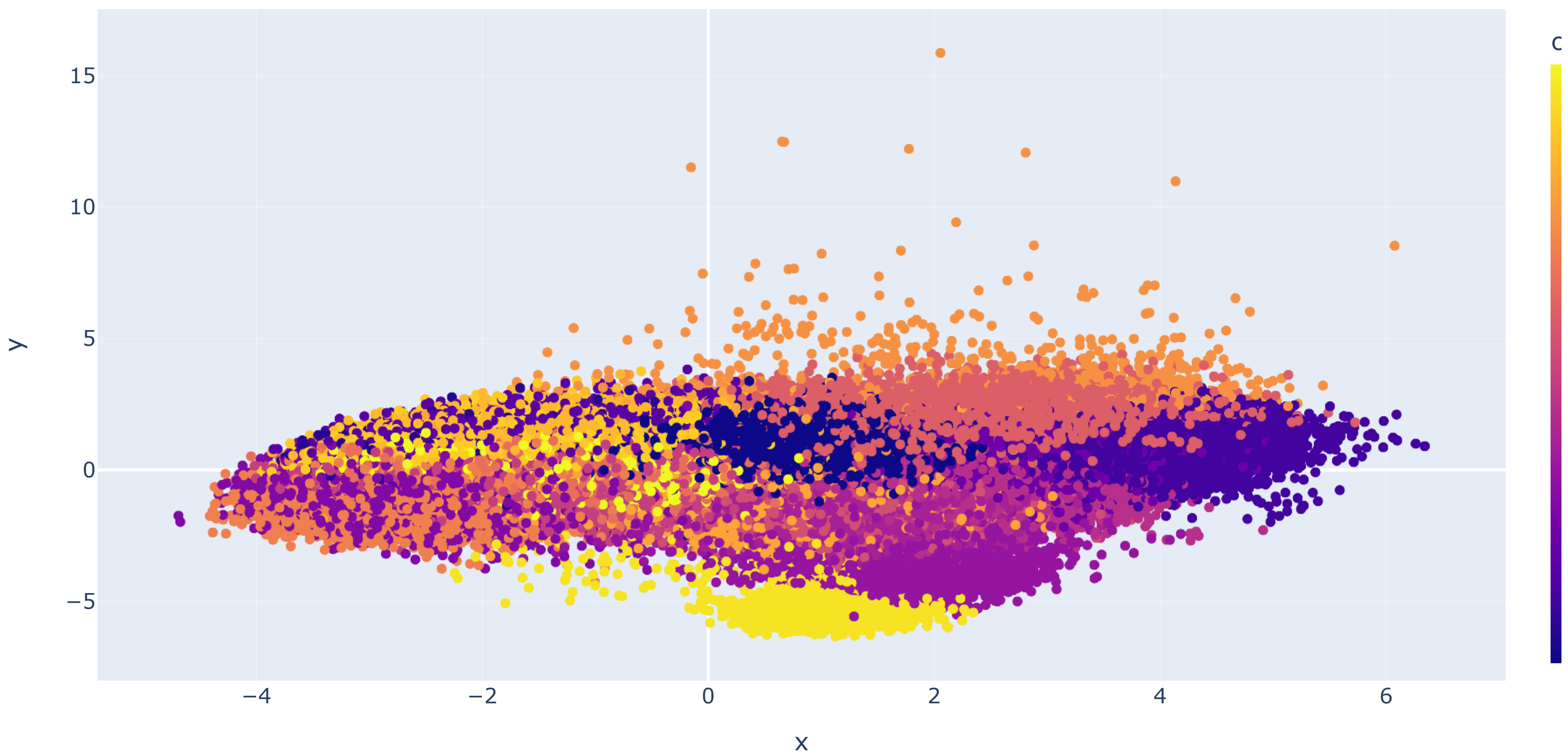
Here, the simple K-means clustering algorithm is used to divide the genres in this dataset into ten clusters based on the numerical audio features of each genres.

In [69]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

cluster_pipeline = Pipeline([('scaler', StandardScaler()), ('kmeans', KMeans(n_clusters=10, n_jobs=-1))])
X = df.select_dtypes(np.number)
cluster_pipeline.fit(X)
df['cluster'] = cluster_pipeline.predict(X)
```

In [74]:
```
song_cluster_pipeline = Pipeline([('scaler', StandardScaler()),
                                  ('kmeans', KMeans(n_clusters=20,
                                   verbose=False, n_jobs=4))
                                 ], verbose=False)

X = df.select_dtypes(np.number)
number_cols = list(X.columns)
song_cluster_pipeline.fit(X)
song_cluster_labels = song_cluster_pipeline.predict(X)
df['cluster_label'] = song_cluster_labels
```

Visualizing the Clusters with PCA

In [76]:
```
from sklearn.decomposition import PCA

pca_pipeline = Pipeline([('scaler', StandardScaler()), ('PCA', PCA(n_components=2))])
song_embedding = pca_pipeline.fit_transform(X)
projection = pd.DataFrame(columns=['x', 'y'], data=song_embedding)
projection['title'] = df['name']
projection['cluster'] = df['cluster_label']

fig = px.scatter(
    projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'title'])
fig.show()
```



Build Recommender System

In [77]:
```
!pip install spotipy
```

```
Collecting spotipy
  Downloading spotipy-2.19.0-py3-none-any.whl (27 kB)
Requirement already satisfied: six>=1.15.0 in c:\users\abc\anaconda3\lib\site-packages (from spotipy) (1.15.0)
Collecting requests>=2.25.0
  Downloading requests-2.27.1-py2.py3-none-any.whl (63 kB)
     ---------------------------------------- 63.1/63.1 kB ? eta 0:00:00
Collecting urllib3>=1.26.0
  Downloading urllib3-1.26.9-py2.py3-none-any.whl (138 kB)
     -------------------------------------- 139.0/139.0 kB 8.6 MB/s eta 0:00:00
Requirement already satisfied: certifi>=2017.4.17 in c:\users\abc\anaconda3\lib\site-packages (from requests>=2.25.0->s
potipy) (2022.5.18.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\abc\anaconda3\lib\site-packages (from requests>=2.25.0->spotip
y) (2.10)
Collecting charset-normalizer~=2.0.0
  Downloading charset_normalizer-2.0.12-py3-none-any.whl (39 kB)
Installing collected packages: urllib3, charset-normalizer, requests, spotipy
  Attempting uninstall: urllib3
    Found existing installation: urllib3 1.25.9
    Uninstalling urllib3-1.25.9:
      Successfully uninstalled urllib3-1.25.9
  Attempting uninstall: requests
    Found existing installation: requests 2.24.0
    Uninstalling requests-2.24.0:
      Successfully uninstalled requests-2.24.0
Successfully installed charset-normalizer-2.0.12 requests-2.27.1 spotipy-2.19.0 urllib3-1.26.9

ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behavio
ur is the source of the following dependency conflicts.
conda 4.13.0 requires ruamel_yaml_conda>=0.11.14, which is not installed.
```

In [92]:
```python
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
from collections import defaultdict

#sp = spotipy.Spotify(auth_manager=SpotifyClientCredentials(client_id=os.environ["SPOTIFY_CLIENT_ID"],
                                #client_secret=os.environ["SPOTIFY_CLIENT_SECRET"]))

def find_song(name, year):
    song_data = defaultdict()
    results = df.search(q= 'track: {} year: {}'.format(name,year), limit=1)
    if results['tracks']['items'] == []:
        return None

    results = results['tracks']['items'][0]
    track_id = results['id']
    audio_features = df.audio_features(track_id)[0]

    song_data['name'] = [name]
    song_data['year'] = [year]
    song_data['explicit'] = [int(results['explicit'])]
    song_data['duration_ms'] = [results['duration_ms']]
    song_data['popularity'] = [results['popularity']]

    for key, value in audio_features.items():
        song_data[key] = value

    return pd.DataFrame(song_data)
```

In [95]:
```python
from collections import defaultdict
from sklearn.metrics import euclidean_distances
from scipy.spatial.distance import cdist
import difflib

number_cols = ['valence', 'year', 'acousticness', 'danceability', 'duration_ms', 'energy', 'explicit',
 'instrumentalness', 'key', 'liveness', 'loudness', 'mode', 'popularity', 'speechiness', 'tempo']


def get_song_data(song, spotify_data):

    try:
        song_data = spotify_data[(spotify_data['name'] == song['name'])
                                & (spotify_data['year'] == song['year'])].iloc[0]
        return song_data

    except IndexError:
        return find_song(song['name'], song['year'])


def get_mean_vector(song_list, spotify_data):

    song_vectors = []

    for song in song_list:
        song_data = get_song_data(song, spotify_data)
        if song_data is None:
            print('Warning: {} does not exist in Spotify or in database'.format(song['name']))
            continue
        song_vector = song_data[number_cols].values
        song_vectors.append(song_vector)

    song_matrix = np.array(list(song_vectors))
    return np.mean(song_matrix, axis=0)


def flatten_dict_list(dict_list):

    flattened_dict = defaultdict()
    for key in dict_list[0].keys():
        flattened_dict[key] = []
```

```python
    for dictionary in dict_list:
        for key, value in dictionary.items():
            flattened_dict[key].append(value)

    return flattened_dict


def recommend_songs( song_list, spotify_data, n_songs=10):

    metadata_cols = ['name', 'year', 'artists']
    song_dict = flatten_dict_list(song_list)

    song_center = get_mean_vector(song_list, spotify_data)
    scaler = song_cluster_pipeline.steps[0][1]
    scaled_data = scaler.transform(spotify_data[number_cols])
    scaled_song_center = scaler.transform(song_center.reshape(1, -1))
    distances = cdist(scaled_song_center, scaled_data, 'cosine')
    index = list(np.argsort(distances)[:, :n_songs][0])

    rec_songs = spotify_data.iloc[index]
    rec_songs = rec_songs[~rec_songs['name'].isin(song_dict['name'])]
    return rec_songs[metadata_cols].to_dict(orient='records')
```