Analytics for Big Data

Project Milestone Report

Spotify - Trend and Sentiment Analysis with Recommendation System

Rajendran Karthick Sharan

Project Overview:

With this study, we plan to develop a ML model to identify and predict the sentiment behind the lyrics and attributes (acousticness, danceability, valence etc.) of a track. Also, we plan to analyze if having certain sentiments and attributes would give a song an edge over other to top the charts. Finally, based on our finding we will develop a recommendation system which suggest songs closest to the mood/sentiment conveyed by a particular track.

Data Extraction:

The data was extracted from Spotify using a developer account and Spotify api. Once we get the developer credentials (client id, and secret), we can use those to authenticate the session and create a 'Spotify' object with the below codes, which we will be using to analyze the tracks.

```
id_df = pd.read_csv('Spotify_id.txt')
```

```
#Authentication - without user
cid = id_df.iloc[0,0]
secret = id_df.iloc[1,0]
client_credentials_manager = SpotifyClientCredentials(client_id=cid, client_secret=secret)
sp = spotipy.Spotify(client_credentials_manager = client_credentials_manager)
```

(Track URI: We will use a track's URI (uniform resource identifiers) to read the song's attributes)

Below we used the global top 50 songs playlist from spotify as a sample and extracted the track URI for all the songs in the playlist.

```
playlist_link = "https://open.spotify.com/playlist/37i9dQZEVXbNG2KDcFcKOF?si=1333723a6eff4b7f"
playlist_URI = playlist_link.split("/")[-1].split("?")[0]
track_uris = [x["track"]["uri"] for x in sp.playlist_tracks(playlist_URI)["items"]]
```

```
track_uris
```

```
['spotify:track:0V3wPSX9ygBnCm8psDIegu',
 'spotify:track:3nqQXoyQOWXiESFLlDF1hG',
 'spotify:track:5jQI2r1RdgtuT8S3iG8zFC',
 'spotify:track:3rWDp9tBPQR9z6U5YyRSK4',
 'spotify:track:4uUG5RXrOk84mYEfFvj3cK',
```

The function ‘autio_features’ of the spotipy package is used to read the attributes of the tracks.

```
sp.audio_features(track_uri)[0]
```

```
{'danceability': 0.804,
 'energy': 0.674,
 'key': 5,
 'loudness': -5.453,
 'mode': 0,
 'speechiness': 0.0333,
 'acousticness': 0.294,
 'instrumentalness': 1.18e-06,
 'liveness': 0.115,
 'valence': 0.292,
 'tempo': 99.968,
 'type': 'audio_features',
 'id': '6Xom58OOXk2SoU711L2IXO',
 'uri': 'spotify:track:6Xom58OOXk2SoU711L2IXO',
 'track_href': 'https://api.spotify.com/v1/tracks/6Xom58OOXk2SoU711L2IXO',
 'analysis_url': 'https://api.spotify.com/v1/audio-analysis/6Xom58OOXk2SoU711L2IXO',
 'duration_ms': 245940,
 'time_signature': 4}
```

We used the below code to retrieve the meta data for a particular track:

```
for track in sp.playlist_tracks(playlist_URI)["items"]:
    #URI
    track_uri = track["track"]["uri"]

    #Track name
    track_name = track["track"]["name"]

    #Main Artist
    artist_uri = track["track"]["artists"][0]["uri"]
    artist_info = sp.artist(artist_uri)

    #Name, popularity, genre
    artist_name = track["track"]["artists"][0]["name"]
    artist_pop = artist_info["popularity"]
    artist_genres = artist_info["genres"]

    #Album
    album = track["track"]["album"]["name"]

    #Popularity of the track
    track_pop = track["track"]["popularity"]
```

```
print('track_name:',track_name)
print('artist_name:',artist_name)
print('album:',album)
```

```
track_name: Moscow Mule
artist_name: Bad Bunny
album: Un Verano Sin Ti
```

Finally, to prepare the dataset, we extracted a **million** playlist (which was the upper limit) from Spotify using the api, we found that the average number of songs in a playlist is **67.**

```
tot = 0
for i in df['tracks']:
    tot += len(i) # total no. of songs across all playlists
print("Average songs in a playlist (For our dataset) :", tot/len(df['tracks']))
```

```
Average songs in a playlist (For our dataset) : 67.503
```

Since, we have 1million playlist and each playlist has 67 songs on average, the scope of data we’ll be working with will become too large, hence, we decided to consider only the **top 100,000 songs** according to their ‘populariy’ score which we extracted in previous step.

The Metadata for each track is being stored in a different DataFrame since we don't need the metadata for our analysis.

The below screenshots illustrates the 1000 .json files, each containing 1000 playlists and the format the playlist and track information are stored inside each .json file.

Quick access
OneDrive
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Network

Name	Date modified	Type	Size
mpd.slice.0-999.json	1/16/2018 6:24 AM	JSON File	33,320 KB
mpd.slice.1000-1999.json	1/16/2018 6:24 AM	JSON File	32,892 KB
mpd.slice.2000-2999.json	1/16/2018 6:29 AM	JSON File	32,411 KB
mpd.slice.3000-3999.json	1/16/2018 6:33 AM	JSON File	32,832 KB
mpd.slice.4000-4999.json	1/16/2018 6:37 AM	JSON File	33,593 KB
mpd.slice.5000-5999.json	1/16/2018 6:41 AM	JSON File	33,307 KB
mpd.slice.6000-6999.json	1/16/2018 6:45 AM	JSON File	32,488 KB
mpd.slice.7000-7999.json	1/16/2018 6:50 AM	JSON File	33,357 KB
mpd.slice.8000-8999.json	1/16/2018 6:54 AM	JSON File	32,586 KB
mpd.slice.9000-9999.json	1/16/2018 6:58 AM	JSON File	31,268 KB
mpd.slice.10000-10999.json	1/16/2018 6:24 AM	JSON File	32,903 KB
mpd.slice.11000-11999.json	1/16/2018 6:25 AM	JSON File	32,064 KB
mpd.slice.12000-12999.json	1/16/2018 6:26 AM	JSON File	34,267 KB
mpd.slice.13000-13999.json	1/16/2018 6:26 AM	JSON File	32,599 KB
mpd.slice.14000-14999.json	1/16/2018 6:27 AM	JSON File	32,850 KB
mpd.slice.15000-15999.json	1/16/2018 6:27 AM	JSON File	32,143 KB
mpd.slice.16000-16999.json	1/16/2018 6:27 AM	JSON File	33,196 KB
mpd.slice.17000-17999.json	1/16/2018 6:28 AM	JSON File	33,826 KB
mpd.slice.18000-18999.json	1/16/2018 6:28 AM	JSON File	33,180 KB
mpd.slice.19000-19999.json	1/16/2018 6:29 AM	JSON File	33,044 KB
mpd.slice.20000-20999.json	1/16/2018 6:29 AM	JSON File	32,874 KB
mpd.slice.21000-21999.json	1/16/2018 6:29 AM	JSON File	32,701 KB
mpd.slice.22000-22999.json	1/16/2018 6:30 AM	JSON File	34,518 KB
mpd.slice.23000-23999.json	1/16/2018 6:30 AM	JSON File	33,162 KB
mpd.slice.24000-24999.json	1/16/2018 6:31 AM	JSON File	32,835 KB
mpd.slice.25000-25999.json	1/16/2018 6:31 AM	JSON File	34,099 KB
mpd.slice.26000-26999.json	1/16/2018 6:31 AM	JSON File	32,180 KB

1,000 items 1,000 items selected

mpd.slice.0-999.json - Notepad

File Edit Format View Help


```
{
    "info": {
        "generated_on": "2017-12-03 08:41:42.057563",
        "slice": "0-999",
        "version": "v1"
    },
    "playlists": [
        {
            "name": "Throwbacks",
            "collaborative": "false",
            "pid": 0,
            "modified_at": 1493424000,
            "num_tracks": 52,
            "num_albums": 47,
            "num_followers": 1,
            "tracks": [
                {
                    "pos": 0,
                    "artist_name": "Missy Elliott",
                    "track_uri": "spotify:track:0UaMYEvWZi0ZqiDOoHU3YI",
                    "artist_uri": "spotify:artist:2wIVse2owClT7go1WT98tk",
                    "track_name": "Lose Control (feat. Ciara & Fat Man Scoop)",
                    "album_uri": "spotify:album:6vV5UrXcfyQD1wu4Qo2I9K",
                    "duration_ms": 226863,
                    "album_name": "The Cookbook"
                },
                {
                    "pos": 1,
                    "artist_name": "Britney Spears",
                    "track_uri": "spotify:track:6I9VzXrHxO9rA9A5euc8Ak",
                    "artist_uri": "spotify:artist:26dSoYclwsYLMAKD3tpOr4",
                    "track_name": "Toxic",
                    "album_uri": "spotify:album:0z7pVBGOD7HCIB7S8eLkLI",
                    "duration_ms": 198800,
```


The following code was used to read the .jsons in dataframe, however we can observe that the tracks inside a playlist are nested, hence they're displayed as a dictionary in a single columns:

```
with open(path,'r') as f:
    data = json.loads(f.read())

# Normalizing data
df = pd.json_normalize(data['playlists'])
```

```
df
```

	name	collaborative	pid	modified_at	num_tracks	num_albums	num_followers	tracks	num_edits	duration_ms	num_artists	descripti
0	Throwbacks	false	0	1493424000	52	47	1	[{'pos': 0, 'artist_name': 'Missy Elliott', 't...	6	11532414	37	Na
1	Awesome Playlist	false	1	1506556800	39	23	1	[{'pos': 0, 'artist_name': 'Survivor', 'track_...	5	11656470	21	Na
2	korean	false	2	1505692800	64	51	1	[{'pos': 0, 'artist_name': 'Hoody', 'track_uri...	18	14039958	31	Na
3	mat	false	3	1501027200	126	107	1	[{'pos': 0, 'artist_name': 'Camille Saint-Saën...	4	28926058	86	Na

We extracted the tracks and their metadata into a DataFrame using the below code:

```
meta = pd.DataFrame([[0,0,0,0,0,0,0,0]], columns=['pos','artist_name','track_uri','artist_uri','track_name','album_uri','duration
```

```
for i in playlist:
    tracks = len(i)
    for j in range(tracks):
        lst = []
        for k in meta.columns:
            lst.append(i[j][k])
        print(lst)
        #meta = meta.append(pd.Series(lst, index=['pos','artist_name','track_uri','artist_uri','track_name','album_uri','duration
        meta.loc[len(meta)] = lst
```

```
[0, 'Missy Elliott', 'spotify:track:0UaMYEvWZi0ZqiDOoHU3YI', 'spotify:artist:2wIVse2owClT7go1WT98tk', 'Lose Control (feat. Ciar
a & Fat Man Scoop)', 'spotify:album:6vV5UrXcfyQD1wu4Qo2I9K', 226863, 'The Cookbook']
[1, 'Britney Spears', 'spotify:track:6I9VzXrHxO9rA9A5euc8Ak', 'spotify:artist:26dSoYclwsYLMAKD3tpOr4', 'Toxic', 'spotify:album:
0z7pVBGOD7HCIB7S8eLkLI', 198800, 'In The Zone']
```

The resulting metadata Dataframe is as below (We'll drop the first row with '0' values during cleaning):

```
meta
```

	pos	artist_name	track_uri	artist_uri	track_name	album_uri	d
0	0	0	0	0	0	0	
1	0	Missy Elliott	spotify:track:0UaMYEvWZi0ZqiDOoHU3YI	spotify:artist:2wIVse2owClT7go1WT98tk	Lose Control (feat. Ciara & Fat Man Scoop)	spotify:album:6vV5UrXcfyQD1wu4Qo2I9K	
2	1	Britney Spears	spotify:track:6I9VzXrHxO9rA9A5euc8Ak	spotify:artist:26dSoYclwsYLMAKD3tpOr4	Toxic	spotify:album:0z7pVBGOD7HCIB7S8eLkLI	
3	2	Beyoncé	spotify:track:0WqIKmW4BTrj3eJFmnCKMv	spotify:artist:6vWDO969PvNqNYHIOW5v0m	Crazy In Love	spotify:album:25hVFAxTlDvXbx2X2QkUkE	
4	3	Justin Timberlake	spotify:track:1AWQoqb9bSvzTjaLralEkT	spotify:artist:31TPClRtHm23RisEBtV3X7	Rock Your Body	spotify:album:6QPkyl04rXwTGlGlcYaRoW	

Once we obtained this dataframe, we used the 'autio_features' function of the sp object we saw earlier on the track_uri's , to retrieve the various track attributes such as 'danceability', 'energy', 'acousticness', 'valence', 'tempo', etc. and store those values into our final dataframe which we will perform our analysis on.

Below are the code used to generate the dataset, and a snippet of the resulting dataframe:

```
tracks = []
for i in meta['track_uri']:
    tracks.append(i)
```

```
df_tracks = pd.DataFrame(sp.audio_features(tracks[0:100]))
for i in range(200,10000,100):
    df_tracks = df_tracks.append(pd.DataFrame(sp.audio_features(tracks[i:(i+100)])))
```

```
df_tracks
```

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	type	
0	0.904	0.813	4	-7.105	0	0.1210	0.03110	0.006970	0.0471	0.810	125.461	audio_features	0UaMYEvWZi0ZqiDOoH
1	0.774	0.838	5	-3.914	0	0.1140	0.02490	0.025000	0.2420	0.924	143.040	audio_features	6I9VzXrHxO9rA9A5eu
2	0.664	0.758	2	-6.583	0	0.2100	0.00238	0.000000	0.0598	0.701	99.259	audio_features	0WqIKmW4BTrj3eJFmnC
3	0.892	0.714	4	-6.055	0	0.1410	0.20100	0.000234	0.0521	0.817	100.972	audio_features	1AWQoqb9bSvzTjaLra
4	0.853	0.606	0	-4.596	1	0.0713	0.05610	0.000000	0.3130	0.654	94.759	audio_features	1lzr43nnXAijlGYnCT8

Description of dataset:

1. The working dataset has **100,000 rows** which we selected as the top 100,000 songs.
2. The dataset has **23 columns**, most of them being numerical with just 2 categorical variables of interest - **'genre'** and **'time_signature'**.
3. We'll drop the unnecessary columns such as **track_id, audio_features** etc. during **EDA.**
4. For trend analysis we'll use the **'popularity'** score as a target variable, and for sentiment analysis and recommendation system we'll use the other attributes such as '**speechiness'**, '**valence'**, etc.

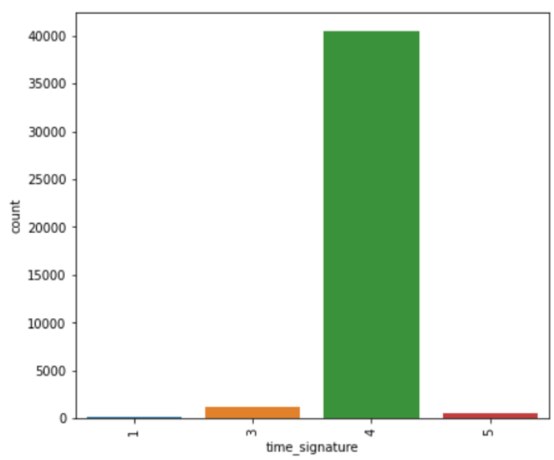
Techniques Used:

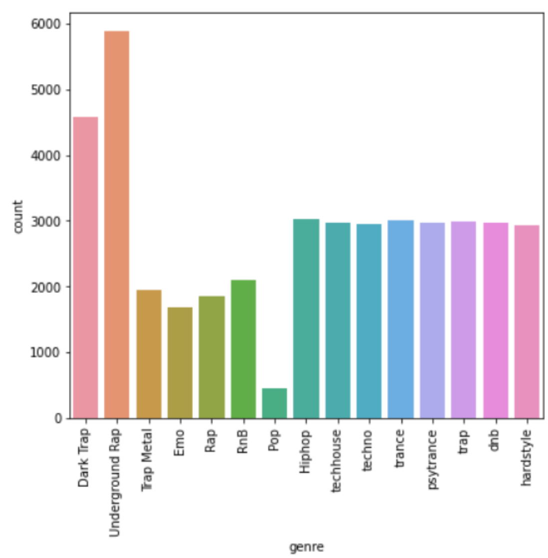
1. Soptipy package was used to extract song metadata from global Spotify playlists
2. In-built functions from spotipy package was used to generate the dataset with top 100,000 songs based on their popularity score.
3. We'll be using matplotlib and seaborn packages to illustrate our analysis during EDA.
4. Inferences will be drawn from EDA which will aid in building and selecting ML models.

Exploratory Data Analysis:

There are 2 categorical columns of interest in our dataset, So, to start with our EDA we will be plotting the countplot for these two categorical variables : "genre" and "time_signature"

```
categorical_cols = ['time_signature', 'genre']
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(15,6))
axs = np.ravel(axs)
for i, col in enumerate(categorical_cols):
    plt.sca(axs[i])
    sns.countplot(data=df, x=col)
    plt.xticks(rotation=90)
plt.show()
```





We can draw the below inferences from the above plots:

1. Time_signature 4 dominates our dataset, more than 90% of tracks out of the top 100,000 has a time_signature of 4. This mean people are generally more receptive towards songs with certain beats per minute which qualifies as time_signature: 4.

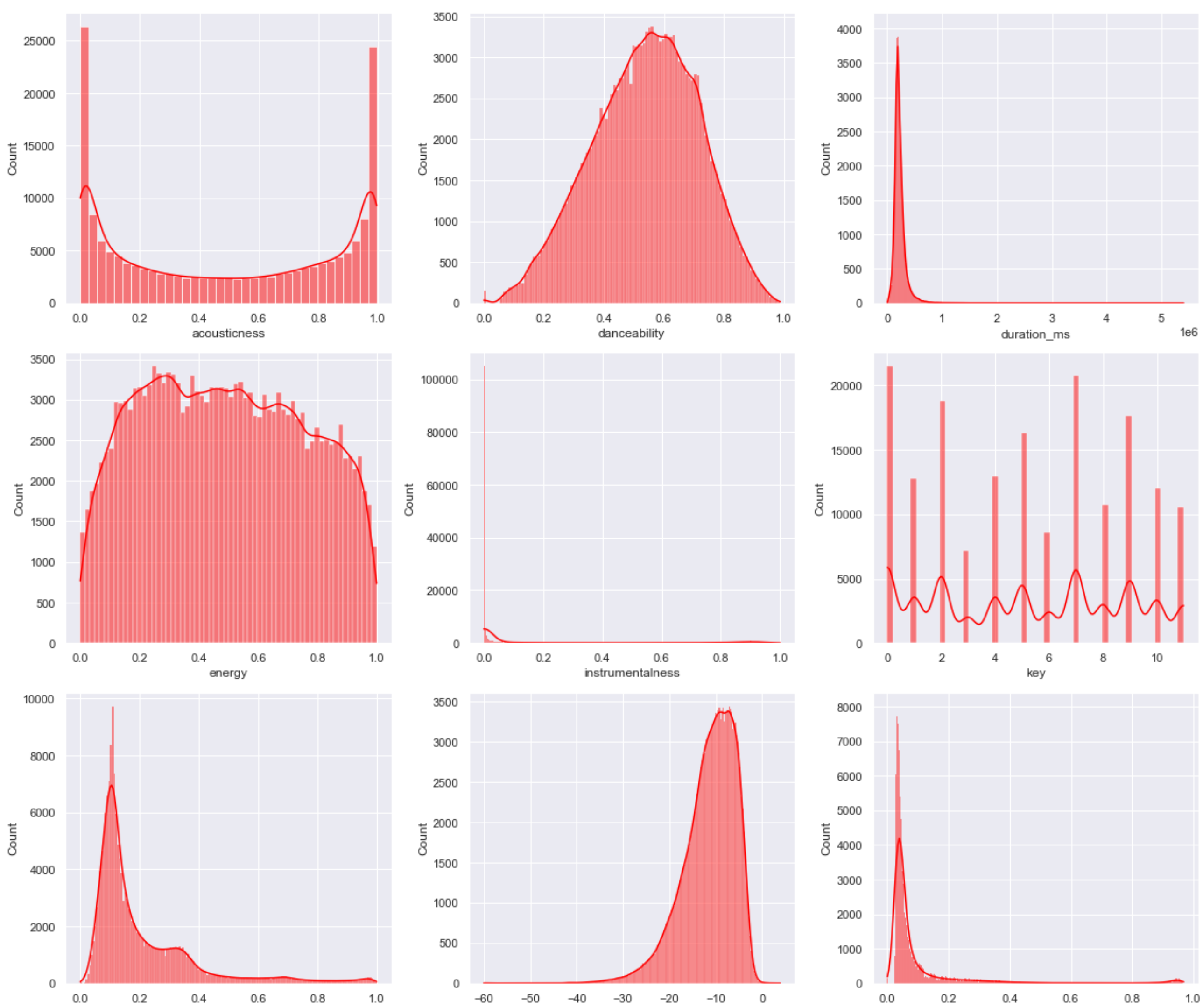
2. Rap and Trap songs contribute the most to the top 100,000 songs, while pop has the least count among all other genres.
3. Genres like Hiphop, Techno, Freestyle, Trance have around equal no. of instances, hence there are equally popular.

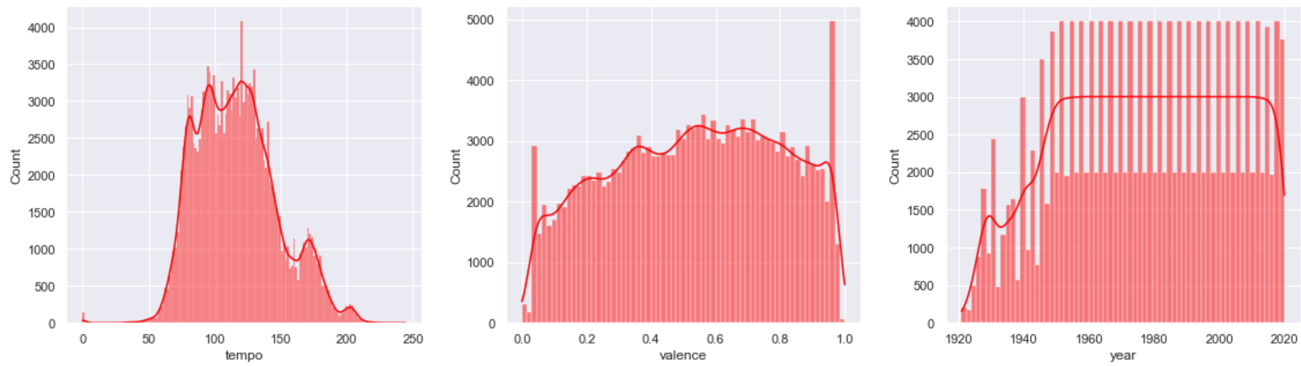
Plotting the distribution of Numerical variables:

```
fig, axs = plt.subplots(nrows=4, ncols=3, figsize=(16,18))
axs = np.ravel(axs)

numerical_cols = data.select_dtypes('number').drop('mode',1).drop('explicit',1).drop('popularity',1).columns.to_list()
for i, col in enumerate(numerical_cols):
    plt.sca(axs[i])
    sns.histplot(data=data, x=col, kde=True, fill=True, color='red')

plt.tight_layout()
plt.show()
```





From the above plots we can infer:

1. Instrumentality and liveliness are heavily right skewed and have a positive bias, we can keep this in mind when building our recommendation engine.
2. Valence shows a steady decrease in count as with respect to it's value, this again is an interesting observation for our recommendation system. Most songs in the top charts have a valence from 0-0.5, which illustrates that neutral or negative emotions actually have a higher staying power, while the positive ones are short lived in people's minds.
3. We can also observe that loudness, danceability and tempo are somewhat close to a normal distribution, this also means that the average dominates in these attributes.

Correlation Matrix for numeric variables:

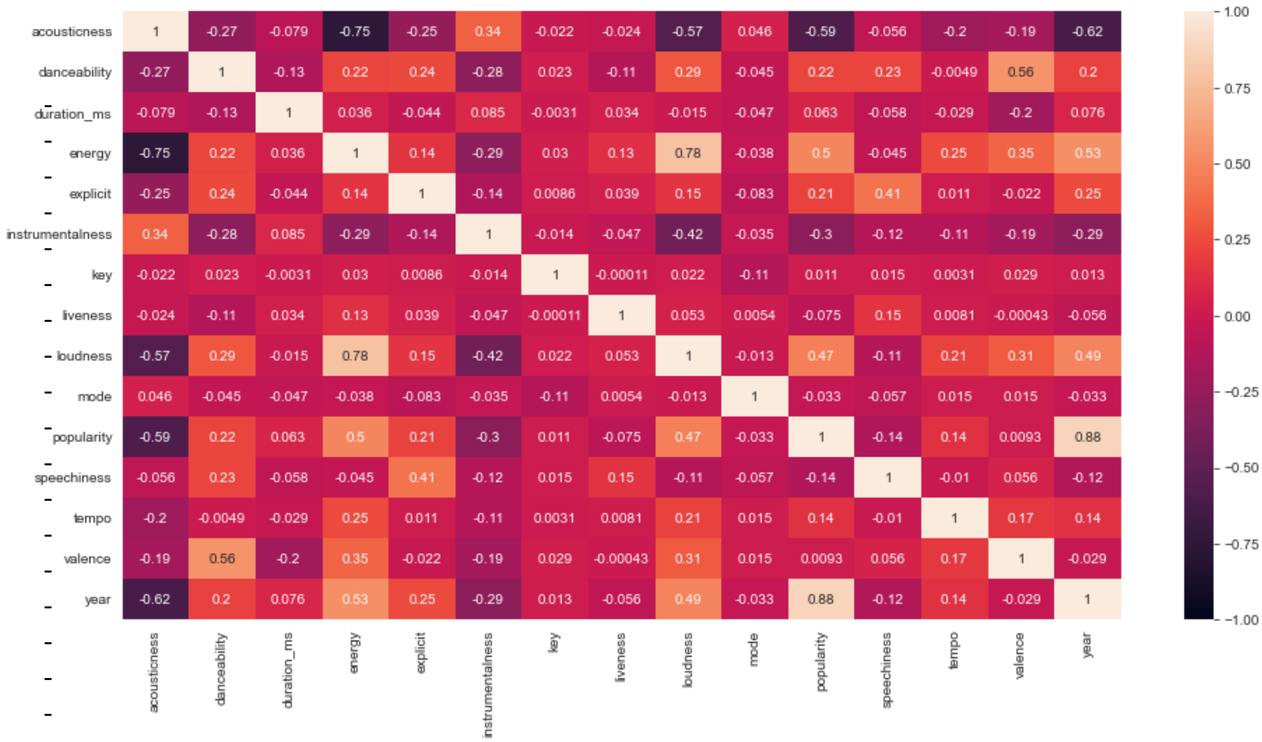
```
data.corr()
```

	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness	loudness	mode	popularity	speech
acousticness	1.000000	-0.265950	-0.079311	-0.750283	-0.253690	0.335821	-0.021686	-0.023871	-0.567072	0.046475	-0.593345	-0.0
danceability	-0.265950	1.000000	-0.134500	0.220569	0.241891	-0.281429	0.022599	-0.105532	0.294170	-0.045306	0.221077	0.2
duration_ms	-0.079311	-0.134500	1.000000	0.036396	-0.043811	0.084814	-0.003116	0.034270	-0.014687	-0.046981	0.063292	-0.0
energy	-0.750283	0.220569	0.036396	1.000000	0.142677	-0.287692	0.029984	0.126293	0.782982	-0.038355	0.497488	-0.0
explicit	-0.253690	0.241891	-0.043811	0.142677	1.000000	-0.138292	0.008578	0.039272	0.152695	-0.083221	0.214044	0.4
instrumentalness	0.335821	-0.281429	0.084814	-0.287692	-0.138292	1.000000	-0.014268	-0.047397	-0.417033	-0.035051	-0.299829	-0.1
key	-0.021686	0.022599	-0.003116	0.029984	0.008578	-0.014268	1.000000	-0.000106	0.021920	-0.112766	0.010675	0.0
liveness	-0.023871	-0.105532	0.034270	0.126293	0.039272	-0.047397	-0.000106	1.000000	0.052985	0.005393	-0.075293	0.1
loudness	-0.567072	0.294170	-0.014687	0.782982	0.152695	-0.417033	0.021920	0.052985	1.000000	-0.013147	0.466546	-0.1
mode	0.046475	-0.045306	-0.046981	-0.038355	-0.083221	-0.035051	-0.112766	0.005393	-0.013147	1.000000	-0.032854	-0.0
popularity	-0.593345	0.221077	0.063292	0.497488	0.214044	-0.299829	0.010675	-0.075293	0.466546	-0.032854	1.000000	-0.1
speechiness	-0.056077	0.225305	-0.058449	-0.045226	0.413074	-0.115735	0.015225	0.147667	-0.105796	-0.057493	-0.135707	1.0
tempo	-0.204982	-0.004872	-0.028816	0.249936	0.011484	-0.107570	0.003148	0.008124	0.211114	0.014539	0.135047	-0.0
valence	-0.185540	0.560242	-0.198760	0.350086	-0.022327	-0.193929	0.029064	-0.000426	0.308418	0.014727	0.009327	0.0
year	-0.624550	0.203430	0.076293	0.532419	0.245227	-0.291571	0.012503	-0.055839	0.490118	-0.033084	0.880724	-0.1

We can draw the below inferences from the correlation matrix:

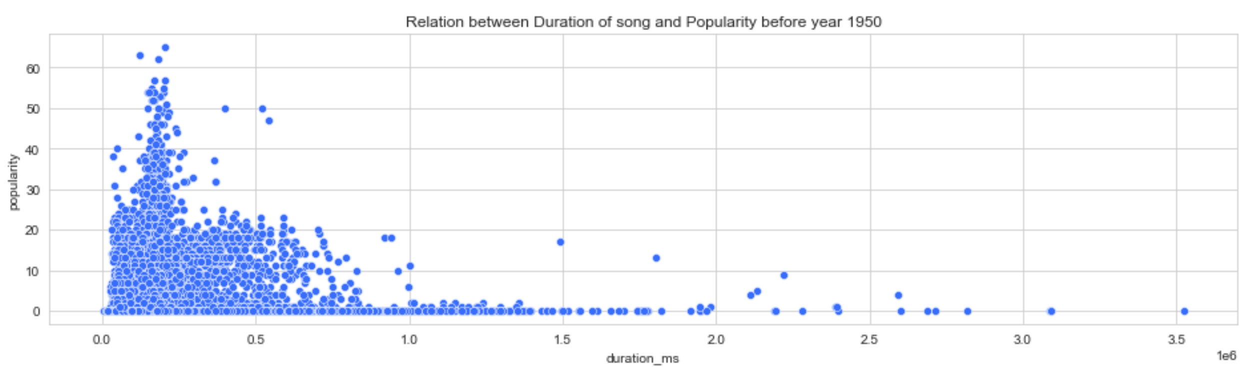
- 'loudness' and 'energy' seem to have a relatively strong positive correlation
- 'energy' and 'accousticness' have an equally strong negative correlation

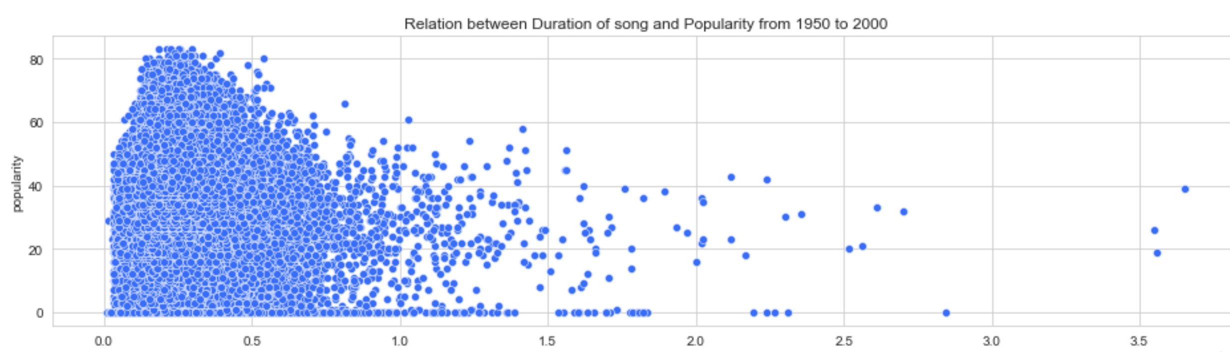
Heatmap of correlation:

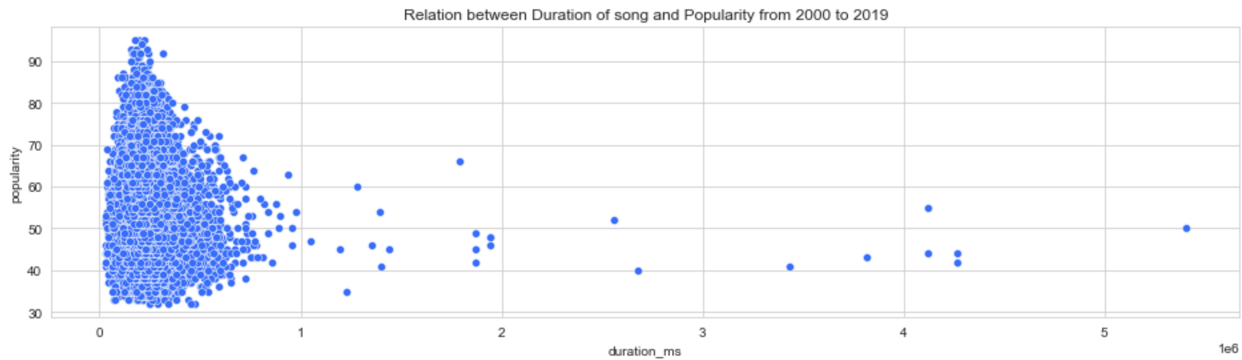


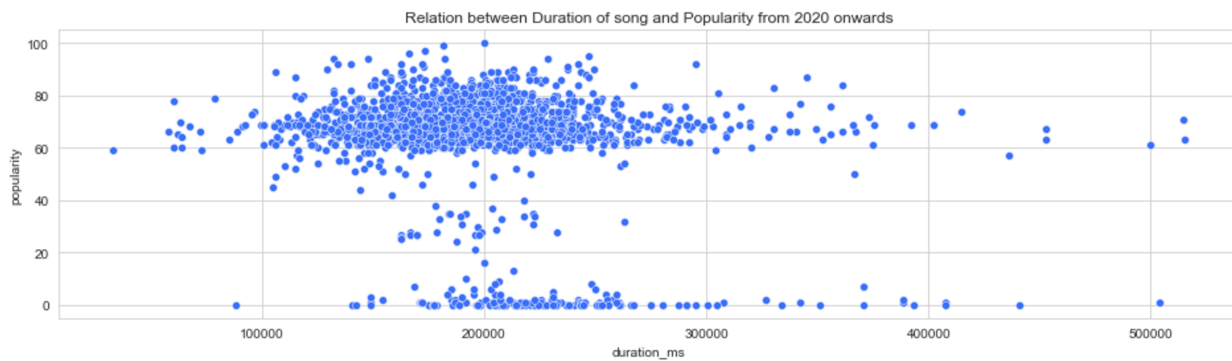
- Additional to our earlier inferences we can observe that 'Accousticness' has a strong negative correlation with 'year', which illustrates that less acoustics songs are released with each passing year.

Trend of Popularity and Song duration over the years:









We can observe a gradual shift in the impact of song duration over the years on popularity. While shorter songs were the clear winner in earlier years, song duration's impact seemed to have waned for over the years with songs released after 2020 have a wide spectrum of duration for their most popular songs.

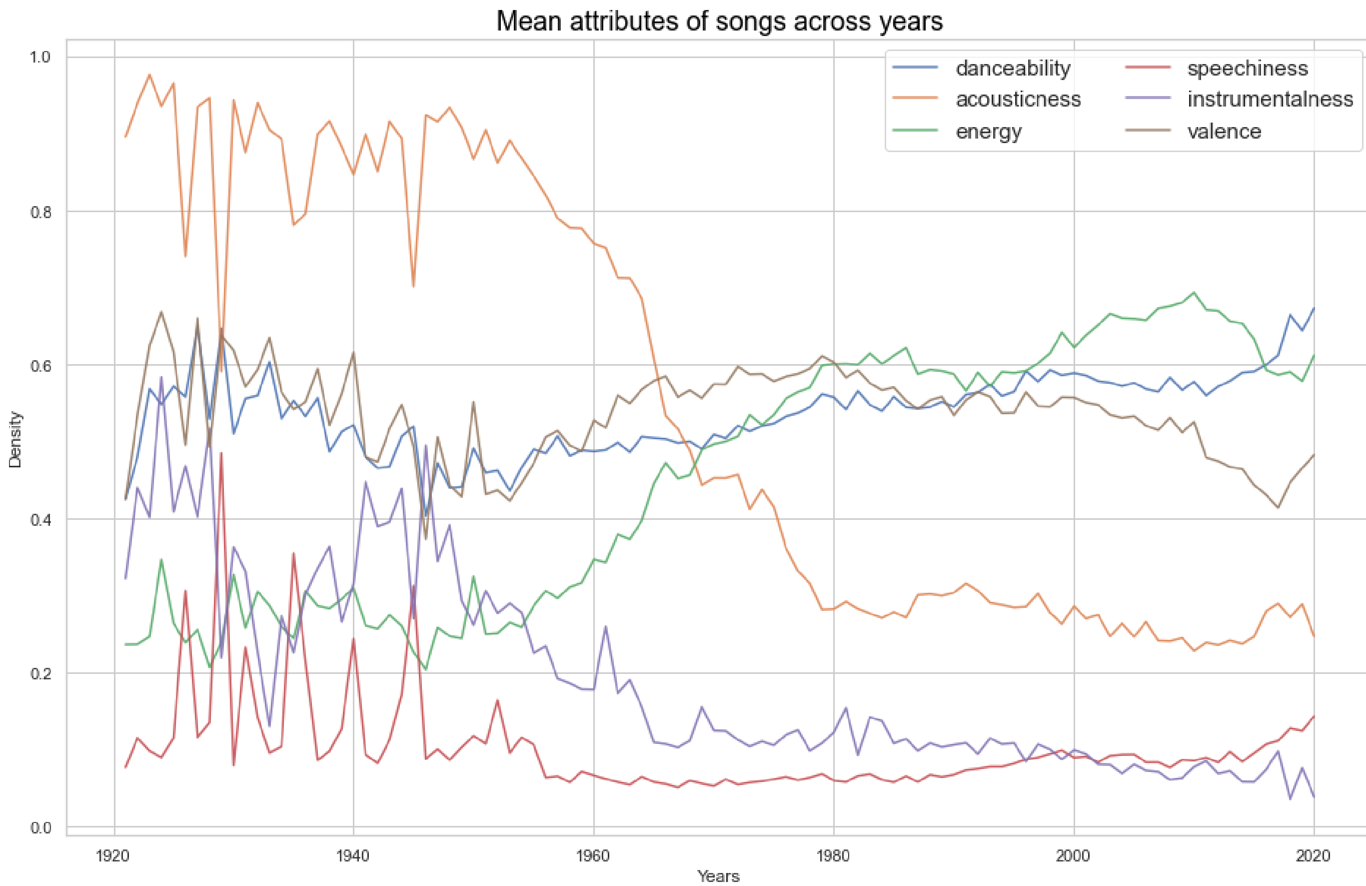
Mean Attributes of Songs over the Years:

The below dataframe was created to store the mean attribute of

- "danceability",
- "acousticness",
- "energy",
- "speechiness",
- "instrumentalness",
- "valence

```
df_med = data[["danceability","acousticness","energy","speechiness","instrumentalness","valence"]].groupby(data.year).mean()
df_med
```

	danceability	acousticness	energy	speechiness	instrumentalness	valence
year						
1921	0.425661	0.895823	0.236784	0.077258	0.322330	0.425495
1922	0.480000	0.939236	0.237026	0.115419	0.440470	0.534056
1923	0.568462	0.976329	0.246936	0.098619	0.401932	0.624788
1924	0.548654	0.935575	0.347033	0.090210	0.583955	0.668574
1925	0.571890	0.965422	0.264373	0.115457	0.408893	0.616430
...	...	...	...	...	...	...
2016	0.599976	0.280290	0.592877	0.107298	0.074646	0.430769
2017	0.612286	0.289916	0.586739	0.111752	0.098209	0.414465
2018	0.664930	0.271941	0.590591	0.128140	0.035948	0.447141
2019	0.644215	0.289298	0.578796	0.124799	0.076518	0.465856
2020	0.673077	0.247374	0.611914	0.143505	0.039052	0.482755



We can see the shift of the mean value of various song attributes across a span of 100 years. Overall, the danceability, energy and valence of a song has gone up while other have been in a steady decline with ‘accousticness’ having the greatest fall over the decades.

Next Steps:

- We plan to do one last deep dive and finalize the EDA, this would be followed by data scaling, encoding and other pre-processing to make it suitable for model building.
- Once we prepare a proper data for model, we’ll build 3 baseline models and evaluate the various features with respect to popularity of a song. Based on findings we’ll proceed with feature selection/deletion/extraction.
- We’ll evaluate our base-model performance based on our feature engineering and select the best base model.
- After selecting our model, we’ll start with tuning hyperparameter and other metrics to get the best performance possible and move to prediction and final model evaluation.
- Finally, to create a recommendation system, we’ll start will building the best clustering model possible and then based on clusters and a song’s attribute values we’ll recommend the nearest best match songs.
- Dimension reduction techniques such as PCA/LDA and deep feature synthesis for feature extraction will be used when needed.