Recommendation System for Spotify Music Dataset

Chandler Ebrahimi

The Problem

For people who listen to music and want to find new songs. People want to find new songs that are similar to a song they already know; a recommendation system can be used to find songs similar to a song they already know.

Who might care?

Average users, general public

People who want to expand their music taste

Data Acquisition

In the dataset there were 42,305 entries of songs.

All formatted in a csv file.

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	...	id	uri
0	0.831	0.814	2	-7.364	1	0.4200	0.0598	0.013400	0.0556	0.3890	...	2Vc6NJ9PW9gD9q343XFRKx	spotify:track:2
1	0.719	0.493	8	-7.230	1	0.0794	0.4010	0.000000	0.1180	0.1240	...	7pgJBLVz5VmnL7uGHmRj6p	spotify:track:7
2	0.850	0.893	5	-4.783	1	0.0623	0.0138	0.000004	0.3720	0.0391	...	0vSWgAlfpye0WCGeNmuNhy	spotify:track:0
3	0.476	0.781	0	-4.710	1	0.1030	0.0237	0.000000	0.1140	0.1750	...	0VSXnJqQkwuH2ei1nOQ1nu	spotify:track:0
4	0.798	0.624	2	-7.668	1	0.2930	0.2170	0.000000	0.1660	0.5910	...	4jCeguq9rMTlbMmPHuO7S3	spotify:track:4

The listed category columns names that was needed to be cleaned.

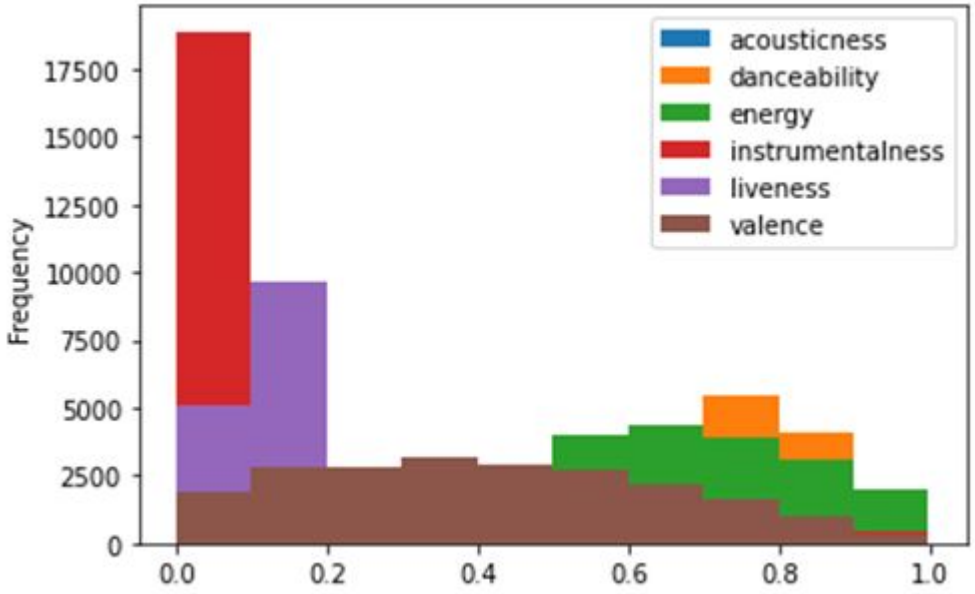
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42305 entries, 0 to 42304
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   danceability      42305 non-null  float64
 1   energy            42305 non-null  float64
 2   key               42305 non-null  int64  
 3   loudness          42305 non-null  float64
 4   mode              42305 non-null  int64  
 5   speechiness       42305 non-null  float64
 6   acousticness      42305 non-null  float64
 7   instrumentalness  42305 non-null  float64
 8   liveness          42305 non-null  float64
 9   valence           42305 non-null  float64
 10  tempo             42305 non-null  float64
 11  type              42305 non-null  object 
 12  id                42305 non-null  object 
 13  uri               42305 non-null  object 
 14  track_href        42305 non-null  object 
 15  analysis_url      42305 non-null  object 
 16  duration_ms       42305 non-null  int64  
 17  time_signature    42305 non-null  int64  
 18  genre             42305 non-null  object 
 19  song_name         21519 non-null  object 
 20  Unnamed: 0        20780 non-null  float64
 21  title             20780 non-null  object 
dtypes: float64(10), int64(4), object(8)
memory usage: 7.1+ MB
```

There weren't too many null values that needed to be addressed in the columns

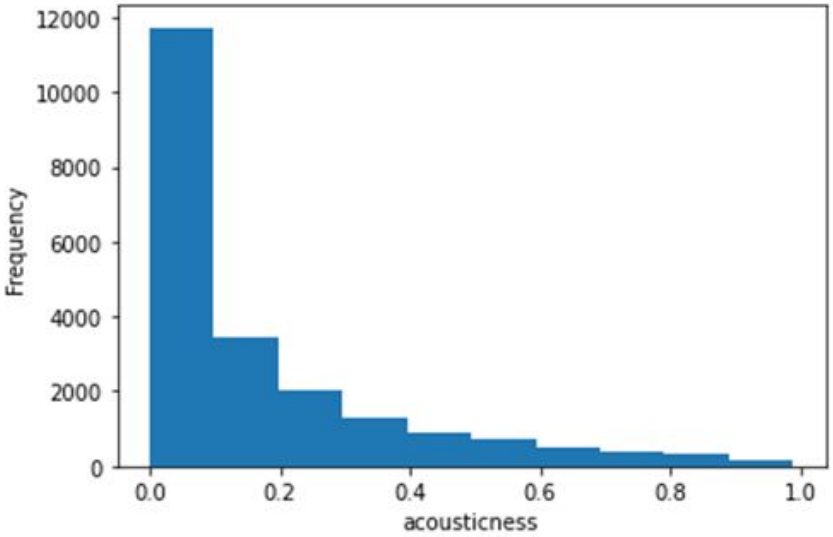
After getting rid of irrelevant and missing value columns we ended with 14 columns to do exploratory data analysis with, with an end total of 21,519 songs.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21519 entries, 0 to 21524
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   danceability      21519 non-null  float64
 1   energy            21519 non-null  float64
 2   key               21519 non-null  int64
 3   loudness          21519 non-null  float64
 4   mode              21519 non-null  int64
 5   speechiness       21519 non-null  float64
 6   acousticness      21519 non-null  float64
 7   instrumentalness  21519 non-null  float64
 8   liveness          21519 non-null  float64
 9   valence           21519 non-null  float64
 10  tempo             21519 non-null  float64
 11  duration_ms       21519 non-null  int64
 12  time_signature    21519 non-null  int64
 13  genre             21519 non-null  object
 14  song_name         21519 non-null  object
dtypes: float64(9), int64(4), object(2)
memory usage: 2.6+ MB
```

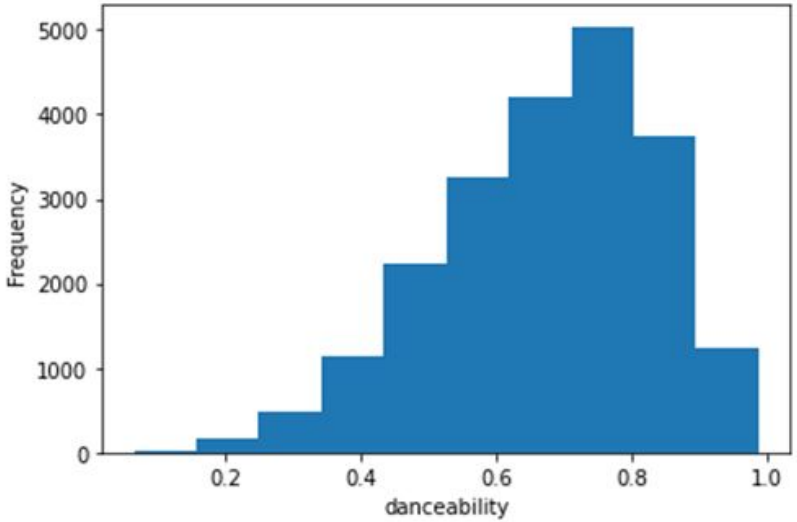
EDA (Exploratory Data Analysis)



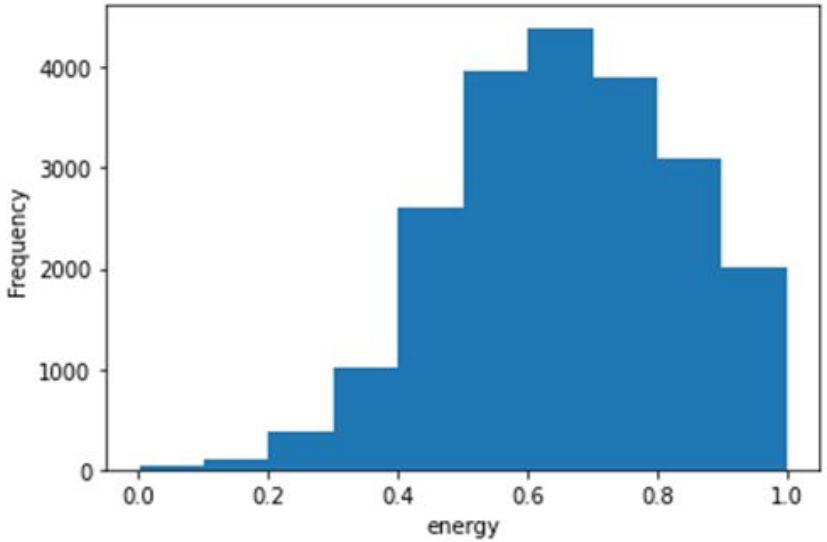
Acousticness



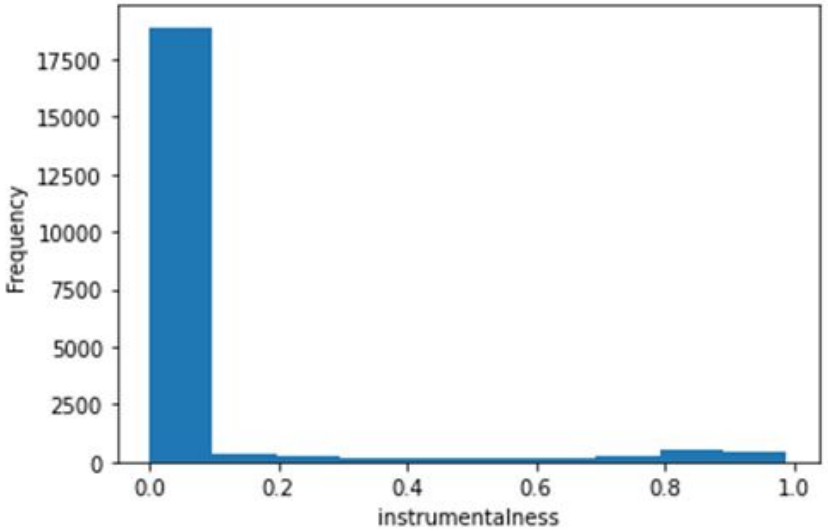
Danceability



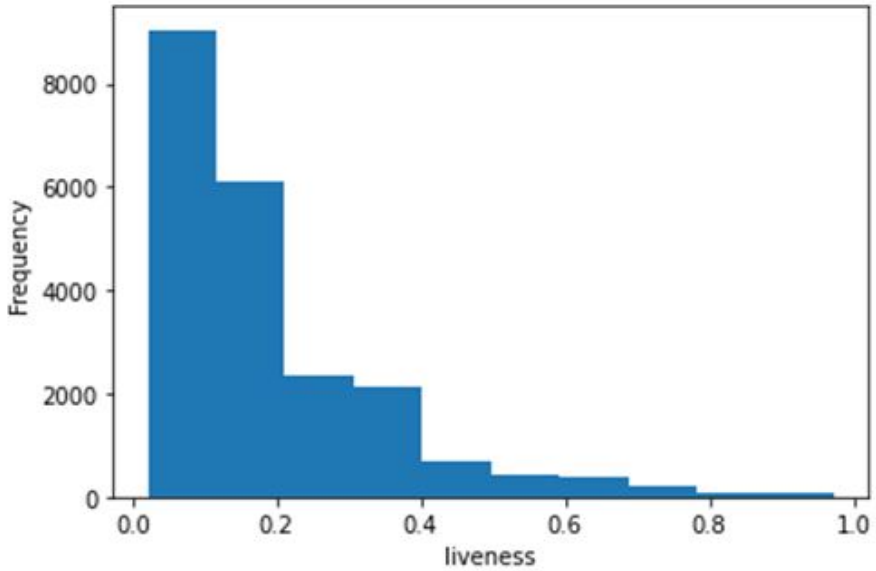
Energy



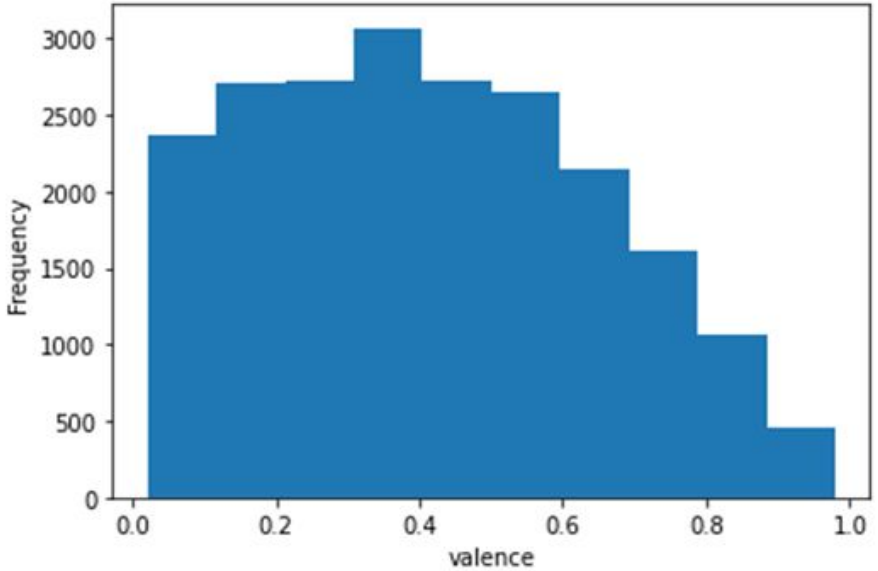
Instrumentalness



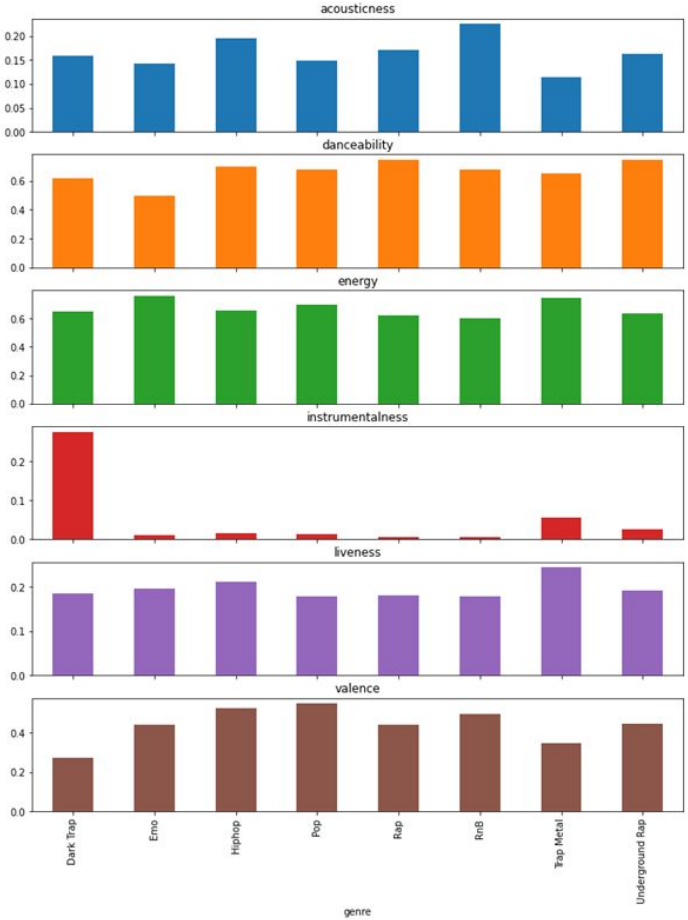
Liveness

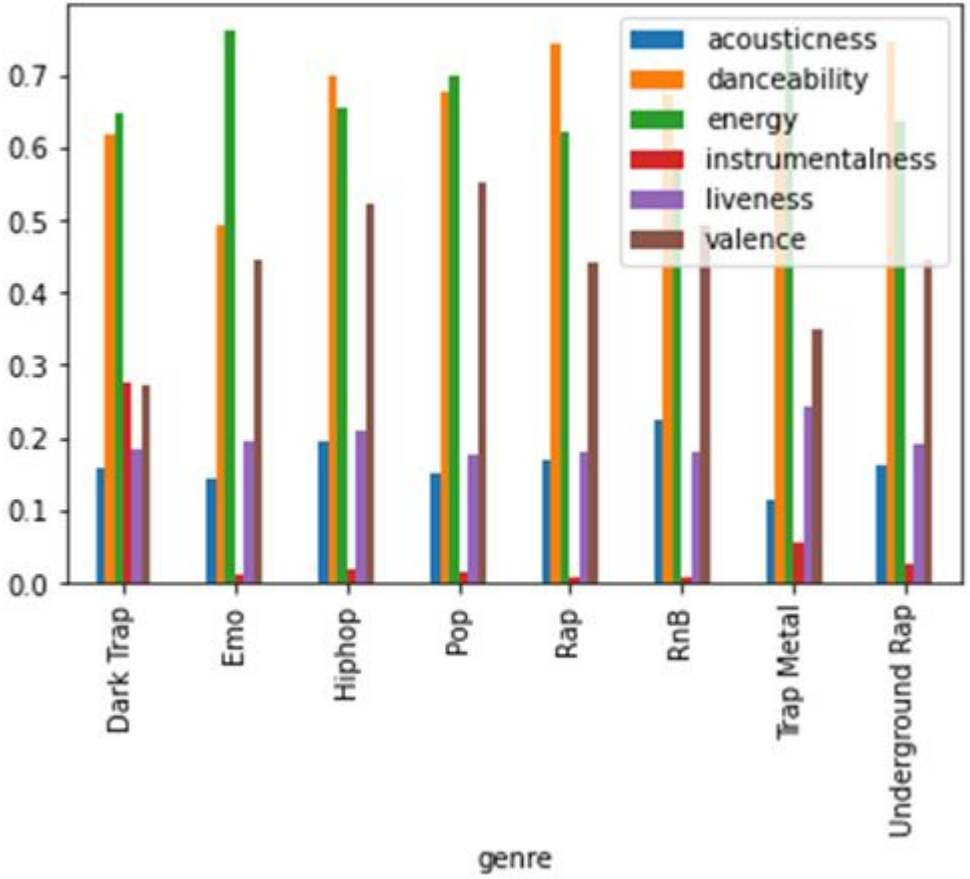


Valence



Average of sound feature by genre




acousticness
danceability
energy
instrumentalness
liveness
valence
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Dark Trap
Emo
Hiphop
Pop
Rap
RnB
Trap Metal
Underground Rap
genre

Pre-processing

I took the data and preformed steps:

- Scaled it using Standard Scaler
- Then Kmeans the genres
- Dimension reduced

```
: 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 = data.select_dtypes(np.number)
cluster_pipeline.fit(X)
data['cluster'] = cluster_pipeline.predict(X)
```

Modeling

For my recommendation system I used a mean vector solution to sum up the sound features, then KMean to find nearest neighbors of that song.

```
def get_mean_vector(song_list, genre, data):

    """
    Gets the mean vector for a list of songs.
    """

    song_vectors = []

    for song in song_list:
        song_data = get_song_data(song, genre, data)
        if song_data is None:
            print('Warning:' + song +  'does not exist in the database')
            return None
        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)
```

Results

A recommendation system, that a user inputs the song and genre and a default of 10 songs will be outputted as a result

```
recommend_songs(['P.I.M.P.'], 'Hiphop', data)
```

```
<bound method BaseEstimator.get_params of StandardScaler()>
```

	song_name	genre
19201	P.I.M.P.	Hiphop
17717	P.I.M.P.	RnB
7520	P.I.M.P.	Underground Rap
20927	Taking over (feat. Djak, Big Stalks & Gfunk)	Hiphop
16026	Wanna Get To Know You	RnB
21360	Jungle	Hiphop
18147	This Is What You Came For (feat. Rihanna)	Pop
20793	My Money	Hiphop
20839	G'D Up	Hiphop
12666	Na Na Na (Na Na Na Na Na Na Na Na Na)	Emo