DIC PROJECT – Spring 2023
PHASE-1

Thilak Reddy Dharam - thilakre - 50469154
Harsha Vardhan Bitra - hbirta - 50468952

Prediction of Songs Popularity on Spotify Data

Problem Statement:

The main aim of this project is to predict the popularity of the song when a new song is inserted into the dataset. The dataset we are using to solve the above is Spotify data: “SpotifyFeatures.csv” containing features such as “acousticness”, “danceability”, “instrumentalness”, “liveness”, “loudness”, etc.

Background of the Problem:

In the music business, popularity is a key element that may impact a song's success, audience, and financial results. Making educated choices regarding marketing, distribution, and promotion tactics may aid musicians, music firms, and record labels. One of the most widely used music streaming services, Spotify, contains a wealth of information about different songs and their attributes. Variables like "acousticness," "danceability," "instrumentalness," "liveness," "loudness," and others included in the Spotify dataset "SpotifyFeatures.csv" may be utilized to train a machine learning model that can reliably predict a song's popularity. As a new song is added to the dataset, the project seeks to use this dataset and machine learning techniques to create a model that can reliably predict a song's popularity.

Significance of the Problem:

1) Better Recommendation of songs.
2) Marketing and Promotion.
3) Understanding the modern trends.
4) Revenue Generation for the song Industry.

Dataset Description:

The dataset used in this project consists of songs from different artists on most popular music platform “Spotify” which has 232725 records and 18 columns.

Dataset Name: SpotifyFeatures.csv
Data Source: Spotify Features

Attributes Description:

genre: Different genres of songs such as classical, jazz, hiphop, etc.
artist_name: Name of the artist who has composed the song.
track_name: Title of the song.
track_id: Unique ID generated for song by Spotify.
popularity: Popularity of song consisting of values between [0-100].
acousticness: Measures the acoustic of song , values consists between [0-1].
danceability: Describes if the song can be used to dance.
duration_ms: Duration of the song in milliseconds.
energy: Represents intensity and activity of the song.
instrumentalness: Represents the vocals of the song.
key: Overall key of the song, using standard pitch class notation. Ex-> 0-C, 1-C#, 2-D.
liveness: Represents the presence of audience in the song.
loudness: Overall loudness of the song in decibels.
mode: Indicates the modality of the song, representing "Major" for 1 and "Minor" for 0.
speechiness: Describes the measure and perfectness of spoken words in the song.
tempo: Beats for minute of the song.
time_signature: Tells the number of beats in each measure of the song.
valence: Musical positiveness delivered by the song.

Data Cleaning/Processing:

Preprocessing:

1) Checking if any attributes contain null values:

```
spotifyData.isnull().sum()
```

```
genre               0
artist_name         0
track_name          0
track_id            0
popularity          0
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
dtype: int64
```

Checked which attributes contains null values and observed that there are no null values in the dataset.

2) Capitalizing the Genres:

Normalizing the text data to make it consistent.

```
#Making Genre texts capital
print(spotifyData["genre"])
spotifyData["genre"]=spotifyData["genre"].apply(lambda x: x.upper())
print(spotifyData["genre"])
```

```
0         Movie
1         Movie
2         Movie
3         Movie
4         Movie
          ...
232720     Soul
232721     Soul
232722     Soul
232723     Soul
232724     Soul
Name: genre, Length: 232725, dtype: object
0         MOVIE
1         MOVIE
2         MOVIE
3         MOVIE
4         MOVIE
          ...
232720     SOUL
232721     SOUL
232722     SOUL
232723     SOUL
232724     SOUL
Name: genre, Length: 232725, dtype: object
```

3) Removing columns that are unnecessary for our requirement.

Removing the columns that makes no sense for our problem statement that doesn't add value.

```
[244]
#Removing columns which doesn't add much value to the study
column_to_remove=["mode"]
req_column=[col for col in spotifyData.columns if not col in column_to_remove]
spotifyData=spotifyData[req_column]
spotifyData.head()
```

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	speechiness	tempo	time_signature	valence
0	MOVIE	Henri Salvador	C'est beau de faire un Show	0BRjO6ga9RKCKjfDqeFgWV	0	0.611	0.389	99373	0.910	0.000	C#	0.3460	-1.828	0.0525	166.969	4/4	0.814
1	MOVIE	Martin & les fées	Perdu d'avance (par Gad Elmaleh)	0BjC1NfoEOOusryehmNudP	1	0.246	0.590	137373	0.737	0.000	F#	0.1510	-5.559	0.0868	174.003	4/4	0.816
2	MOVIE	Joseph Williams	Don't Let Me Be Lonely Tonight	0CoSDzoNIKCRs124s9uTVy	3	0.952	0.663	170267	0.131	0.000	C	0.1030	-13.879	0.0362	99.488	5/4	0.368
3	MOVIE	Henri Salvador	Dis-moi Monsieur Gordon Cooper	0Gc6TVm52BwZD07Ki6tlvf	0	0.703	0.240	152427	0.326	0.000	C#	0.0985	-12.178	0.0395	171.758	4/4	0.227
4	MOVIE	Fabien Nataf	Ouverture	0lusIXpMROHdEPvSI1fTQK	4	0.950	0.331	82625	0.225	0.123	F	0.2020	-21.150	0.0456	140.576	4/4	0.390

4) Converting datatypes of key and time_signature from object to category

```
genre                   object
artist_name             object
track_name              object
track_id                object
popularity               int64
acousticness           float64
danceability           float64
duration_ms              int64
energy                 float64
instrumentalness       float64
key                     object
liveness               float64
loudness               float64
speechiness            float64
tempo                  float64
time_signature          object
valence                float64
dtype: object


genre                   object
artist_name             object
track_name              object
track_id                object
popularity               int64
acousticness           float64
danceability           float64
duration_ms              int64
energy                 float64
instrumentalness       float64
key                   category
liveness               float64
loudness               float64
speechiness            float64
tempo                  float64
time_signature        category
valence                float64
dtype: object
```

5) Text Filtering: removing duplication because of punctuation.

To show the relevant data and to show the consistency between attribute names, we have removed duplicate attributes.

```
[246]  spotifyData["genre"].unique()

array(['MOVIE', 'R&B', 'A CAPELLA', 'ALTERNATIVE', 'COUNTRY', 'DANCE',
       'ELECTRONIC', 'ANIME', 'FOLK', 'BLUES', 'OPERA', 'HIP-HOP',
       "CHILDREN'S MUSIC", 'CHILDREN’S MUSIC', 'RAP', 'INDIE',
       'CLASSICAL', 'POP', 'REGGAE', 'REGGAETON', 'JAZZ', 'ROCK', 'SKA',
       'COMEDY', 'SOUL', 'SOUNDTRACK', 'WORLD'], dtype=object)
```

```
[247]  #The punctution above is causing duplicate values (Here CHILDREN'S MUSIC is considered as two categories because of puncuation)
def rem_punctuation(st):
  st=st.strip()
  st=re.sub(r'[^\w\s]', '',st)
  return st

spotifyData["genre"]=spotifyData["genre"].apply(lambda x: rem_punctuation(x))
spotifyData["genre"].unique()

array(['MOVIE', 'RB', 'A CAPELLA', 'ALTERNATIVE', 'COUNTRY', 'DANCE',
       'ELECTRONIC', 'ANIME', 'FOLK', 'BLUES', 'OPERA', 'HIPHOP',
       'CHILDRENS MUSIC', 'RAP', 'INDIE', 'CLASSICAL', 'POP', 'REGGAE',
       'REGGAETON', 'JAZZ', 'ROCK', 'SKA', 'COMEDY', 'SOUL', 'SOUNDTRACK',
       'WORLD'], dtype=object)
```

6) Some tracks have multiple genres, so adding columns as genres as list.

Since a single song has multiple genres, adding all the genres to the specific list.

```
[248]  #Single track has multiple genre so we combine it into a list
df_grouped=(spotifyData.groupby("track_id")["genre"].apply(list)).to_frame().reset_index()
spotifyData=pd.merge(spotifyData,df_grouped, left_on='track_id', right_on='track_id', how='left')
spotifyData.head()
```

	genre_x	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	speechiness	tempo	time_signature	valence	genre_y
0	MOVIE	Henri Salvador	C'est beau de faire un Show	0BRjO6ga9RKCKjfDqeFgWV	0	0.611	0.389	99373	0.910	0.000	C#	0.3460	-1.828	0.0525	166.969	4/4	0.814	[MOVIE]
1	MOVIE	Martin & les fées	Perdu d'avance (par Gad Elmaleh)	0BjC1NfoEOOusryehmNudP	1	0.246	0.590	137373	0.737	0.000	F#	0.1510	-5.559	0.0868	174.003	4/4	0.816	[MOVIE]
2	MOVIE	Joseph Williams	Don't Let Me Be Lonely Tonight	0CoSDzoNIKCRs124s9uTVy	3	0.952	0.663	170267	0.131	0.000	C	0.1030	-13.879	0.0362	99.488	5/4	0.368	[MOVIE]
3	MOVIE	Henri Salvador	Dis-moi Monsieur Gordon Cooper	0Gc6TVm52BwZD07Ki6tlvf	0	0.703	0.240	152427	0.326	0.000	C#	0.0985	-12.178	0.0395	171.758	4/4	0.227	[MOVIE]
4	MOVIE	Fabien Nataf	Ouverture	0IusIXpMROHdEPvSI1fTQK	4	0.950	0.331	82625	0.225	0.123	F	0.2020	-21.150	0.0456	140.576	4/4	0.390	[MOVIE]

7) Removing duplicates from "track_id" column.

```
[249]  spotifyData.drop_duplicates(subset="track_id",keep=False, inplace=True)
```

8) Removing invalid columns.

Removing the two invalid attributes “genre_x” and “genre_y” generated in the above step.

```
[250]  spotifyData["genre_list"]=spotifyData["genre_y"]
       spotifyData.drop(["genre_x","genre_y"],axis=1,inplace=True)
```

9) Renaming duration_ms attribute to duration_in_milliseconds.

```
[251]  spotifyData.rename(columns = {"duration_ms" : "duration_in_milliseconds"}, inplace = True)
```

10.Checking for outliers:

checking the outlier values of all the attributes whether the data is logical and deviating or not.

```
spotifyData.describe()
```

	popularity	acousticness	danceability	duration_in_milliseconds	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence
count	141650.000000	141650.000000	141650.000000	1.416500e+05	141650.000000	141650.000000	141650.000000	141650.000000	141650.000000	141650.000000	141650.000000
mean	31.980854	0.436685	0.528496	2.371573e+05	0.544834	0.193319	0.234449	-10.656996	0.134734	116.693988	0.449168
std	15.850546	0.373820	0.193533	1.406640e+05	0.286310	0.338497	0.222889	6.700037	0.221962	31.677253	0.274753
min	0.000000	0.000000	0.056900	1.538700e+04	0.000098	0.000000	0.010200	-52.457000	0.022200	30.379000	0.000000
25%	22.000000	0.054400	0.396000	1.732302e+05	0.307000	0.000000	0.097700	-13.821000	0.037000	91.089250	0.207000
50%	33.000000	0.355000	0.546000	2.186685e+05	0.578000	0.000116	0.132000	-8.669000	0.049000	114.073500	0.437000
75%	43.000000	0.831000	0.674000	2.716898e+05	0.792000	0.196000	0.292000	-5.775000	0.099000	138.120500	0.674000
max	97.000000	0.996000	0.989000	5.552917e+06	0.999000	0.999000	1.000000	3.744000	0.967000	242.903000	1.000000

11.Trimming the data:

trying to trim the top 10% and bottom 10% of data from the specified columns and filter the records containing missing values.

```
lower=0.1
upper=0.9
#Trim Top 10 and bottom 10 percentile data
trim_columns=["valence","liveness","loudness","instrumentalness","tempo"]
quant_df = spotifyData[trim_columns].quantile([lower, upper])
spotifyData[trim_columns] = spotifyData[trim_columns].apply(lambda x: x[(x >= quant_df.loc[lower,x.name]) & (x <= quant_df.loc[upper,x.name])], axis=0)
spotifyData=spotifyData.dropna()
spotifyData=spotifyData.reset_index(drop=True)
spotifyData.describe()
```

	popularity	acousticness	danceability	duration_in_milliseconds	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence
count	57422.000000	57422.000000	57422.000000	5.742200e+04	57422.000000	57422.000000	57422.000000	57422.00000	57422.000000	57422.000000	57422.000000
mean	34.786772	0.372163	0.577919	2.371130e+05	0.558835	0.101310	0.187387	-9.11863	0.095096	115.770611	0.452796
std	16.038214	0.342739	0.162686	1.238481e+05	0.233037	0.235026	0.118189	3.84302	0.143020	22.453504	0.213922
min	0.000000	0.000001	0.079100	1.550900e+04	0.004540	0.000000	0.075200	-20.39500	0.022200	78.513000	0.074200
25%	26.000000	0.046200	0.466000	1.825205e+05	0.381000	0.000000	0.105000	-11.37000	0.034000	96.002000	0.272000
50%	37.000000	0.261000	0.587000	2.216820e+05	0.574000	0.000039	0.134000	-8.09100	0.045400	115.248000	0.450000
75%	46.000000	0.701000	0.698000	2.686795e+05	0.744000	0.014275	0.243000	-6.06000	0.082600	133.912000	0.633000
max	97.000000	0.996000	0.986000	4.791725e+06	0.998000	0.873000	0.626000	-4.16600	0.967000	164.028000	0.841000

12.Clipping the data:

Trying to clip the top 1% and bottom 1% of data from the specified columns, which doesn't allows values to go outside of range.

```
lower=0.01
upper=0.99
#Clip Top 1 and Bottom 1 percentile data to the nearest value.
clip_columns=["valence","liveness","loudness","instrumentalness","tempo"]
lower = spotifyData[clip_columns].quantile(lower)
upper = spotifyData[clip_columns].quantile(upper)
spotifyData[clip_columns] = spotifyData[clip_columns].clip(lower=lower, upper=upper, axis=1)
spotifyData=spotifyData.dropna()
spotifyData=spotifyData.reset_index(drop=True)
spotifyData.describe()
```

	popularity	acousticness	danceability	duration_in_milliseconds	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence
count	57422.000000	57422.000000	57422.000000	5.742200e+04	57422.000000	57422.000000	57422.000000	57422.000000	57422.000000	57422.000000	57422.000000
mean	34.786772	0.372163	0.577919	2.371130e+05	0.558835	0.101239	0.187207	-9.115243	0.095096	115.758099	0.452799
std	16.038214	0.342739	0.162686	1.238481e+05	0.233037	0.234795	0.117519	3.831465	0.143020	22.412582	0.213765
min	0.000000	0.000001	0.079100	1.550900e+04	0.004540	0.000000	0.076900	-19.579790	0.022200	79.385210	0.082500
25%	26.000000	0.046200	0.466000	1.825205e+05	0.381000	0.000000	0.105000	-11.370000	0.034000	96.002000	0.272000
50%	37.000000	0.261000	0.587000	2.216820e+05	0.574000	0.000039	0.134000	-8.091000	0.045400	115.248000	0.450000
75%	46.000000	0.701000	0.698000	2.686795e+05	0.744000	0.014275	0.243000	-6.060000	0.082600	133.912000	0.633000
max	97.000000	0.996000	0.986000	4.791725e+06	0.998000	0.860000	0.588000	-4.266210	0.967000	161.112790	0.832000

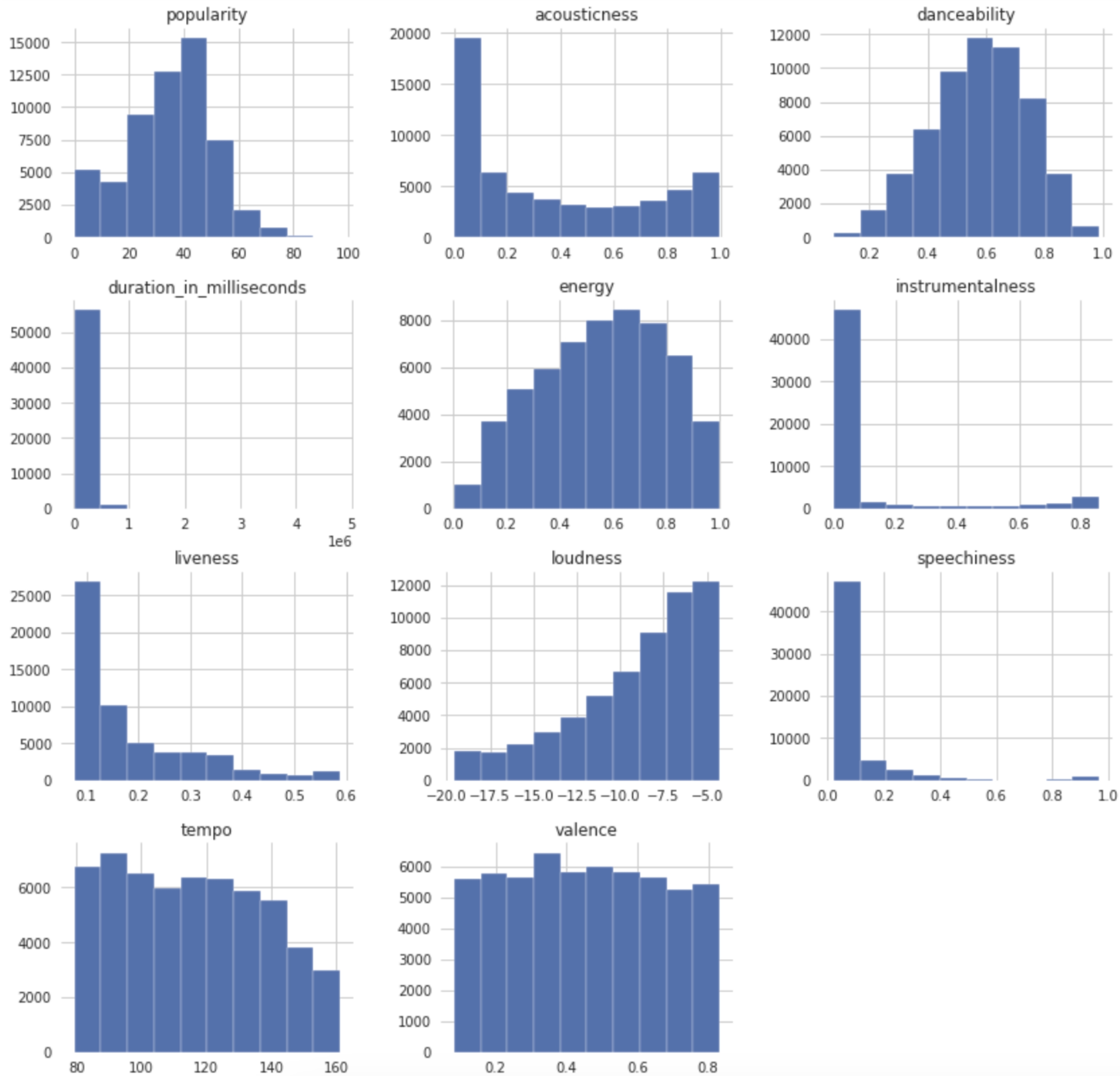
13.Rearranging columns for better readability.

```
[255]
orderList = [2,1,0,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
spotifyData = spotifyData[[spotifyData.columns[i] for i in orderList]]
spotifyData.head()
```

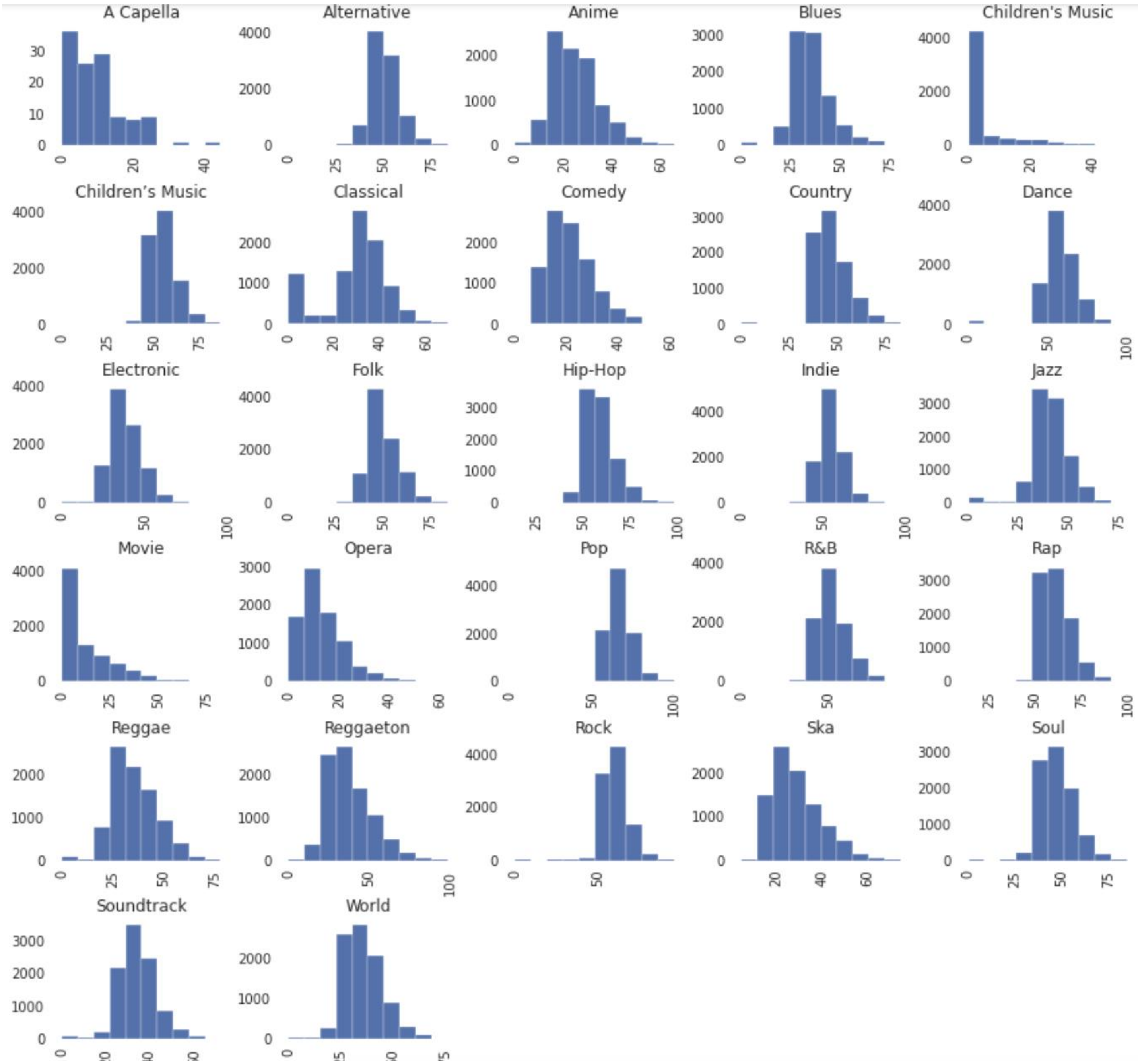
	track_id	track_name	artist_name	popularity	acousticness	danceability	duration_in_milliseconds	energy	instrumentalness	key	liveness	loudness	speechiness	tempo	time_signature	valence	genre_list
0	0CoSDzoNIKCRs124s9uTVy	Don't Let Me Be Lonely Tonight	Joseph Williams	3	0.95200	0.663	170267	0.1310	0.00000	C	0.1030	-13.879	0.0362	99.488	5/4	0.368	[MOVIE]
1	0Mf1jKa8eNAf1a4PwTbizj	Le petit souper aux chandelles	Henri Salvador	0	0.74900	0.578	160627	0.0948	0.00000	C#	0.1070	-14.970	0.1430	87.479	4/4	0.358	[MOVIE]
2	0NUiKYRd6jt1LKMYGkUdnZ	Premières recherches (par Paul Ventimila, Lori...	Martin & les fées	2	0.34400	0.703	212293	0.2700	0.00000	C#	0.1050	-12.675	0.9530	82.873	4/4	0.533	[MOVIE]
3	0PbIF9YVD505GutwotpB5C	Let Me Let Go	Laura Mayne	15	0.93900	0.416	240067	0.2690	0.00000	F#	0.1130	-8.949	0.0286	96.827	4/4	0.274	[MOVIE]
4	0ST6uPfvaPpJLtQwhE6KfC	Helka	Chorus	0	0.00104	0.734	226200	0.4810	0.00086	C	0.0769	-7.725	0.0460	125.080	4/4	0.765	[MOVIE]

EDA (Exploratory Data Analysis):

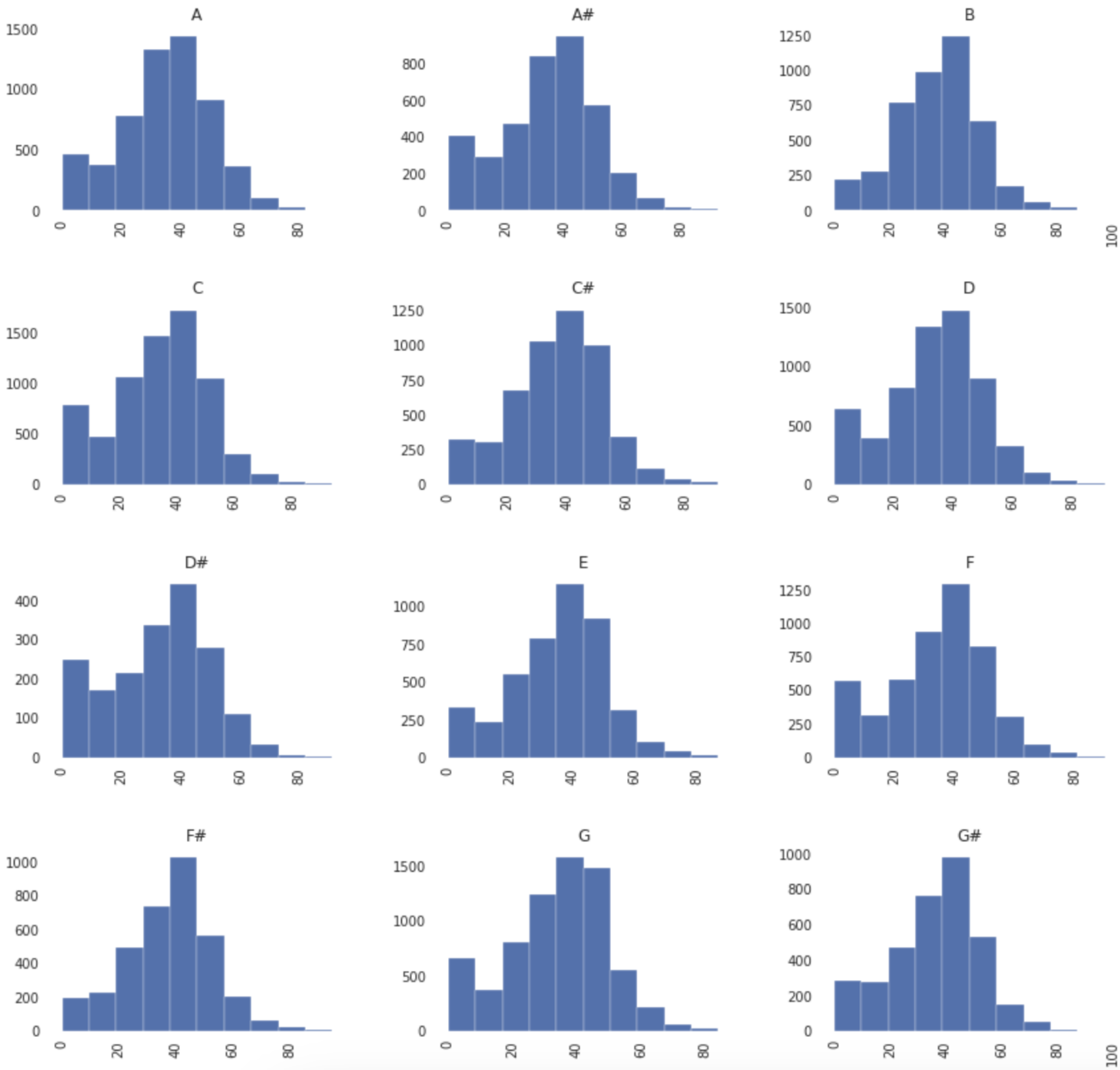
1) Histogram of Each Columns:



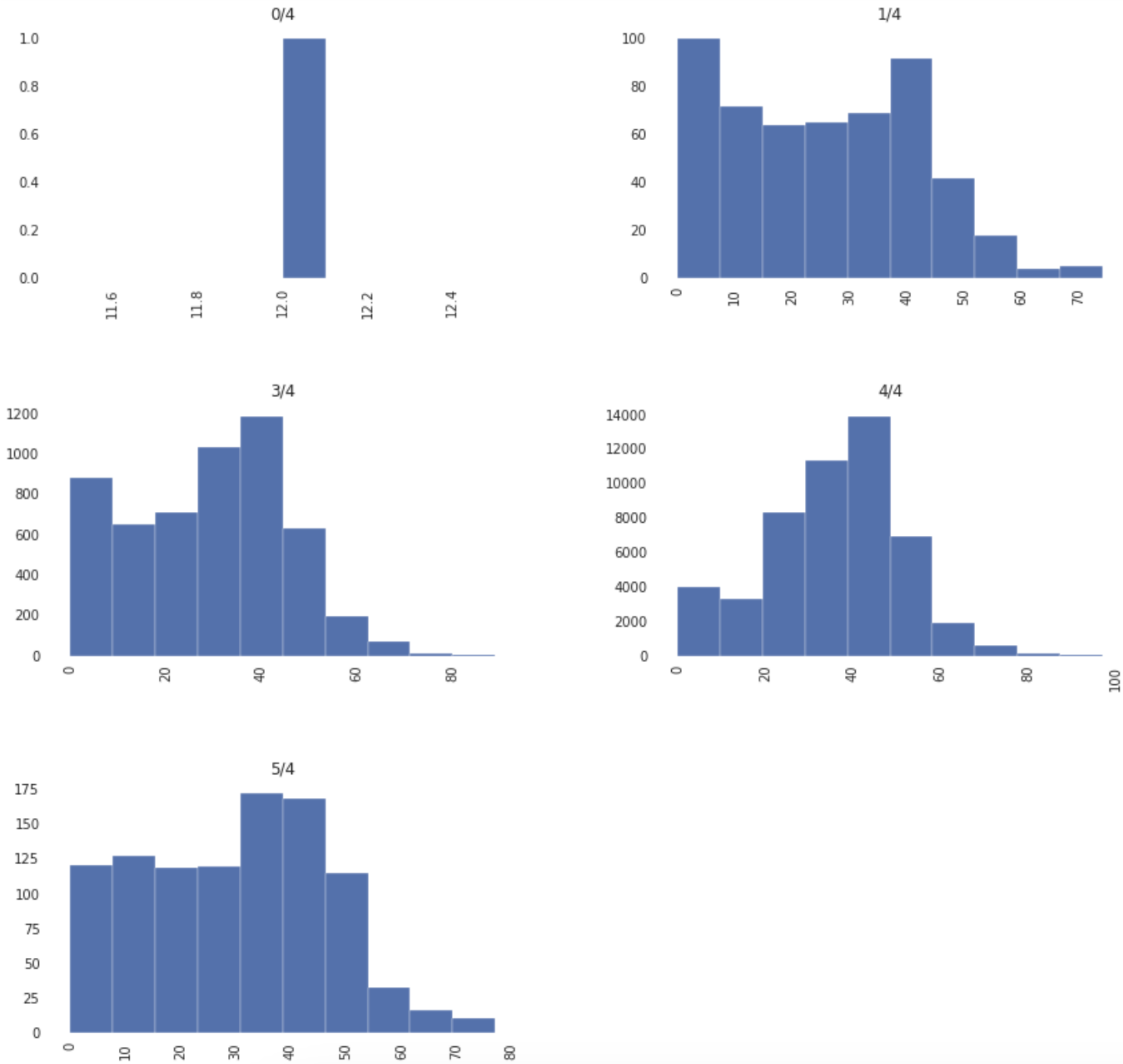
2) Histograms of Genre using original Data:



3) Histogram for Category Columns – "Key":



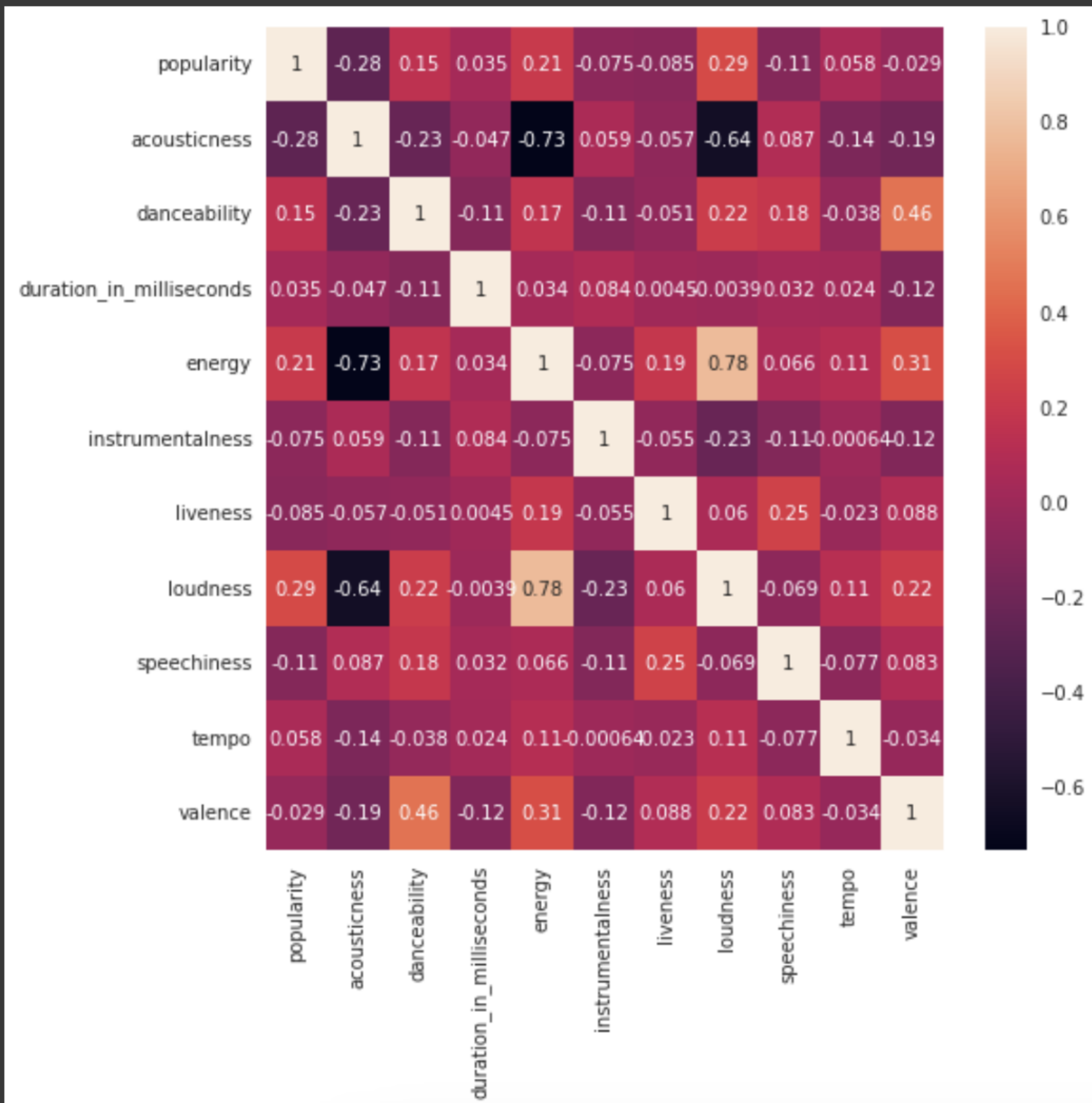
Histogram for Category Columns – "time_signature":



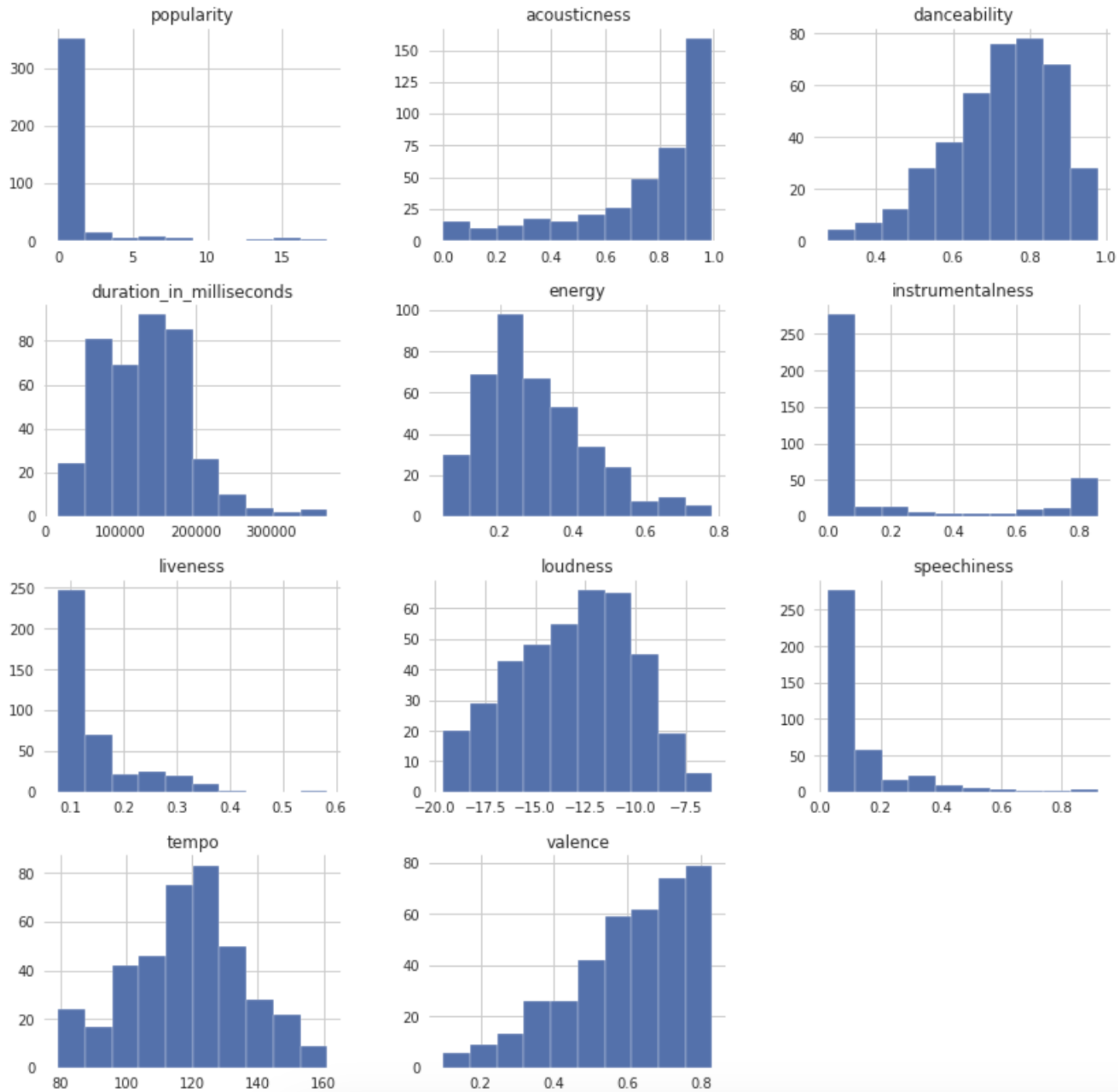
4) Cross-Correlation Matrix:

We have used heatmap to check the correlation between the song features.

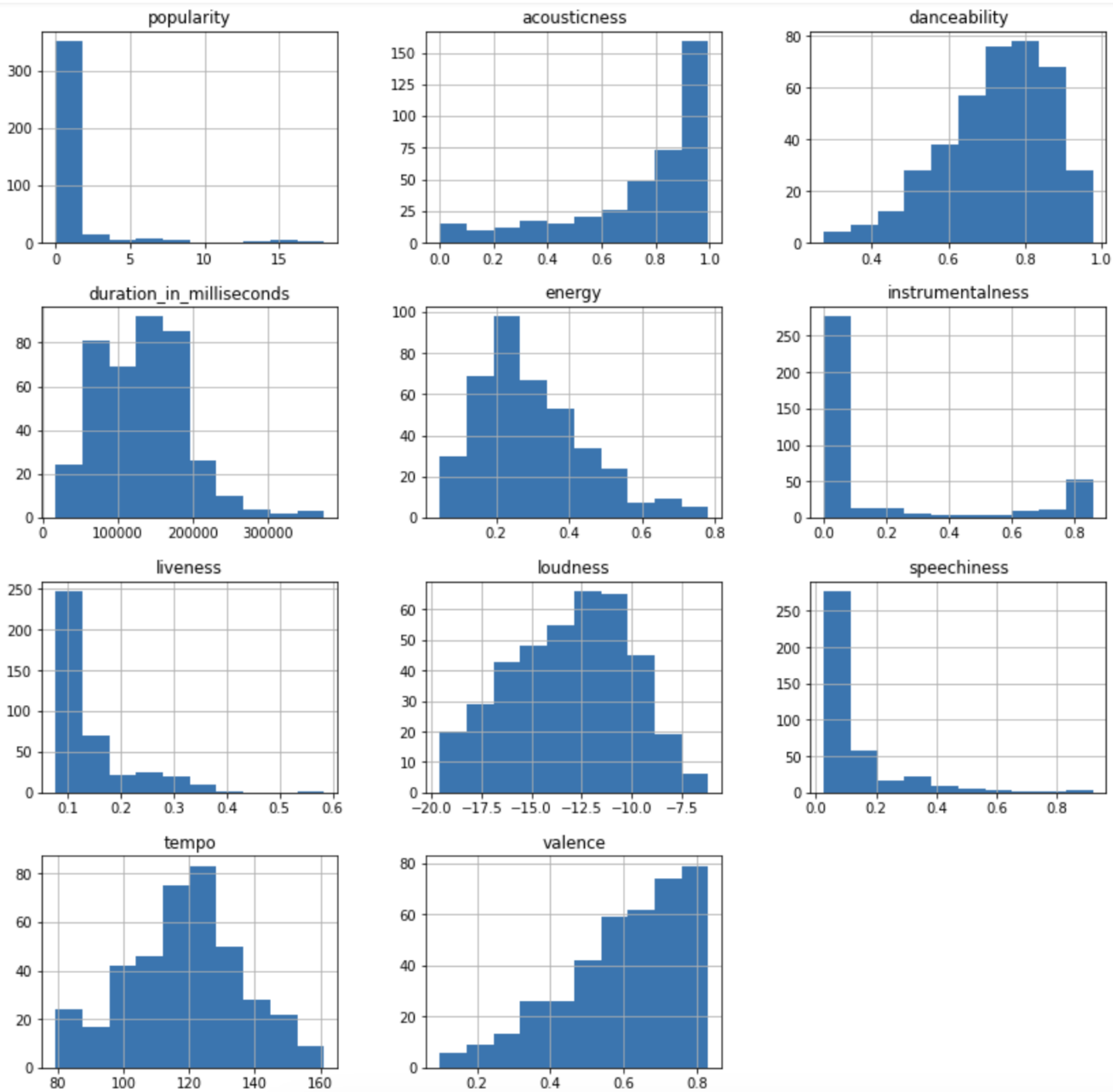
```
corr_matrix = spotifyData.corr()
plt.figure(figsize = (8,8))
sns.heatmap(corr_matrix, annot=True)
plt.show()
```



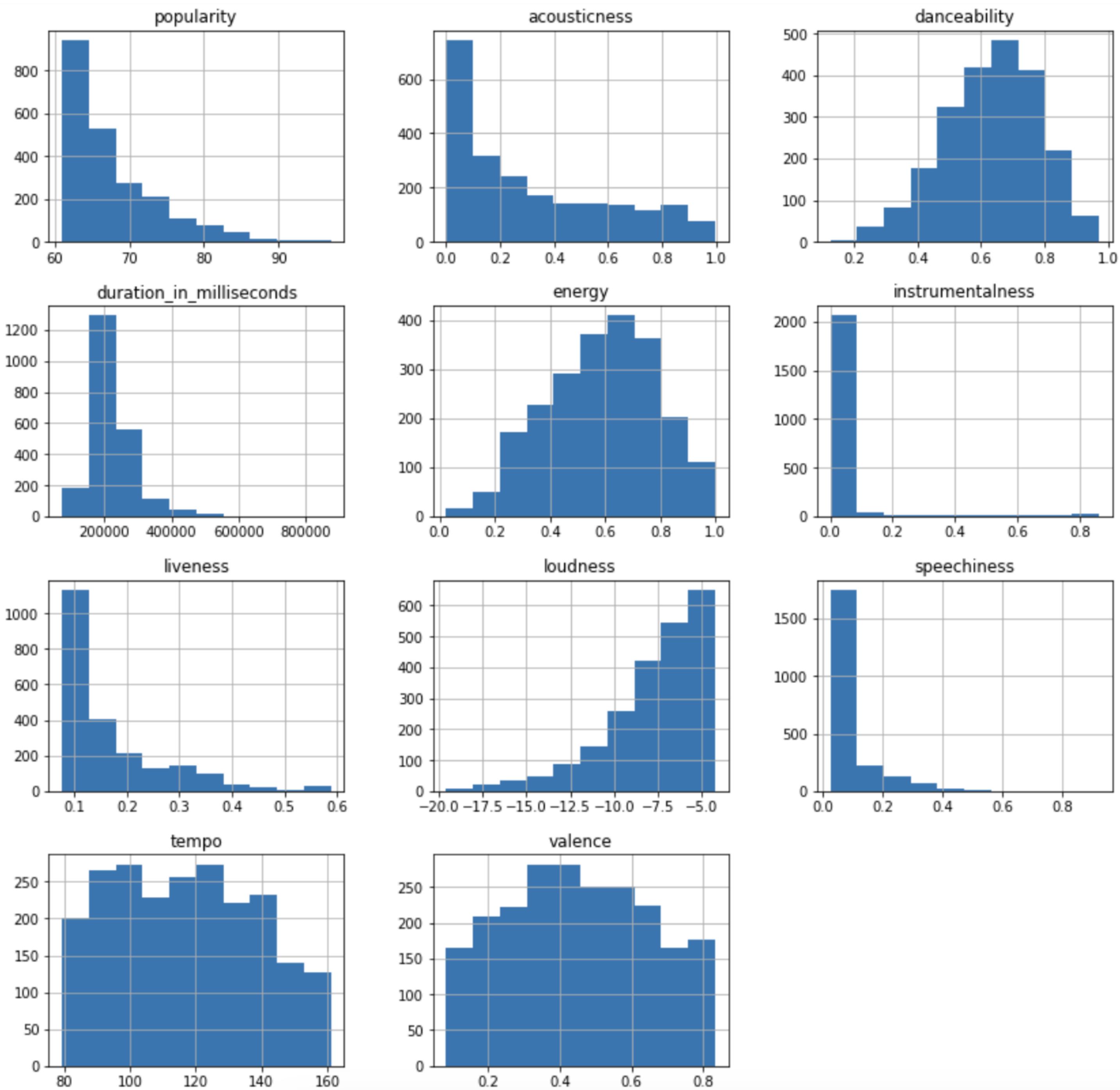
5) Analysis of artists with most songs:



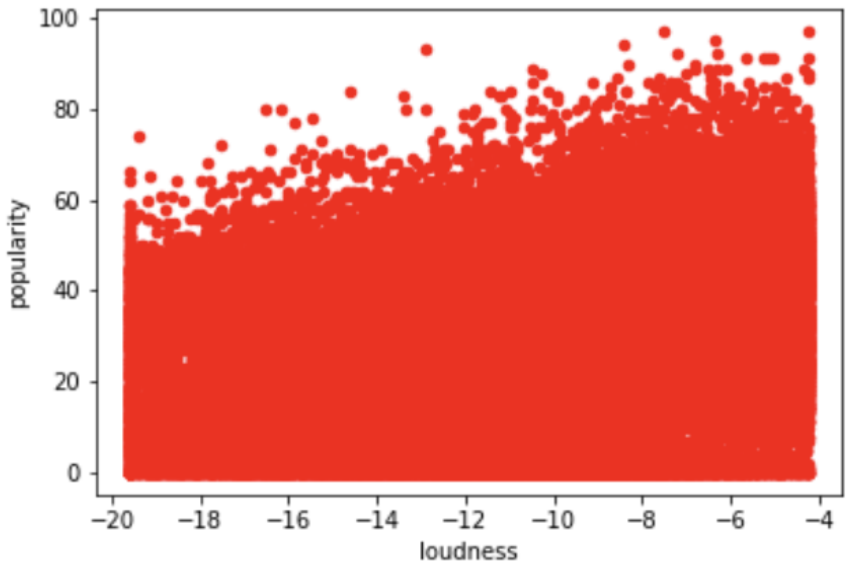
6) Analysis of Artist who's mean popularity is highest:



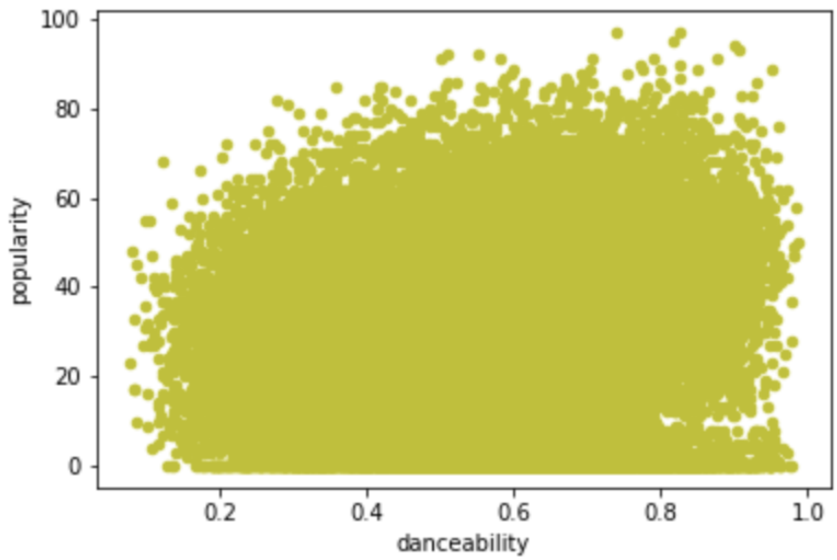
7) Analysis of most popular tracks:



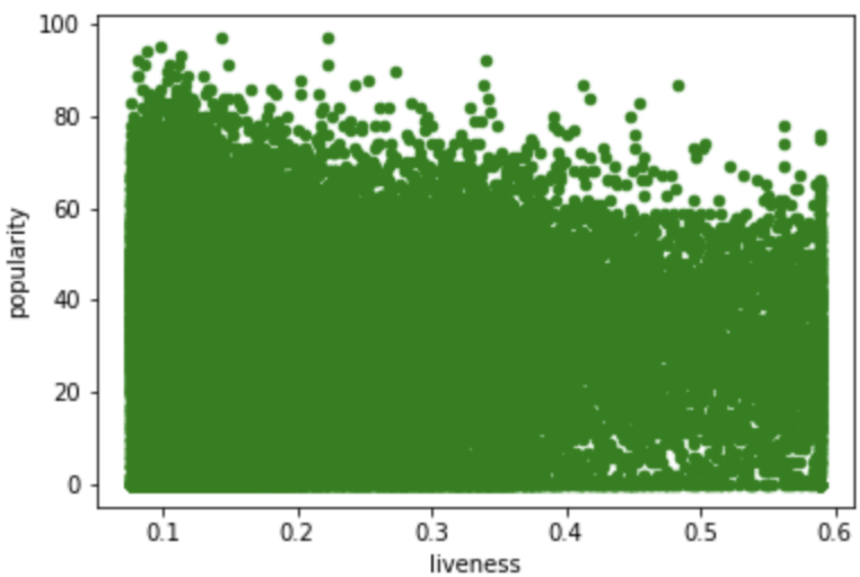
8) Scatter plot of few features with respect to popularity:



popularity
loudness
100
80
60
40
20
0
−20
−18
−16
−14
−12
−10
−8
−6
−4

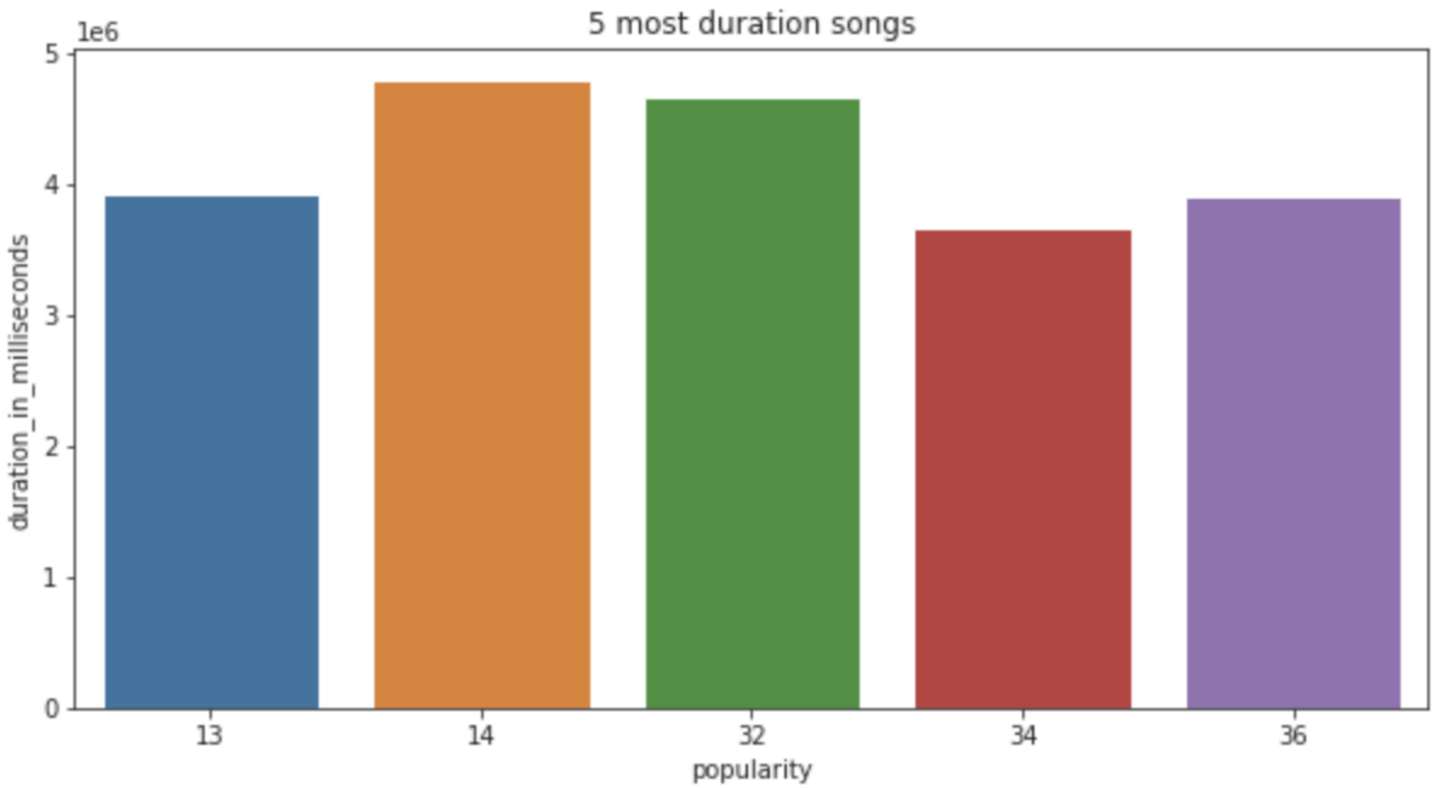


popularity
danceability
100
80
60
40
20
0
0.2
0.4
0.6
0.8
1.0

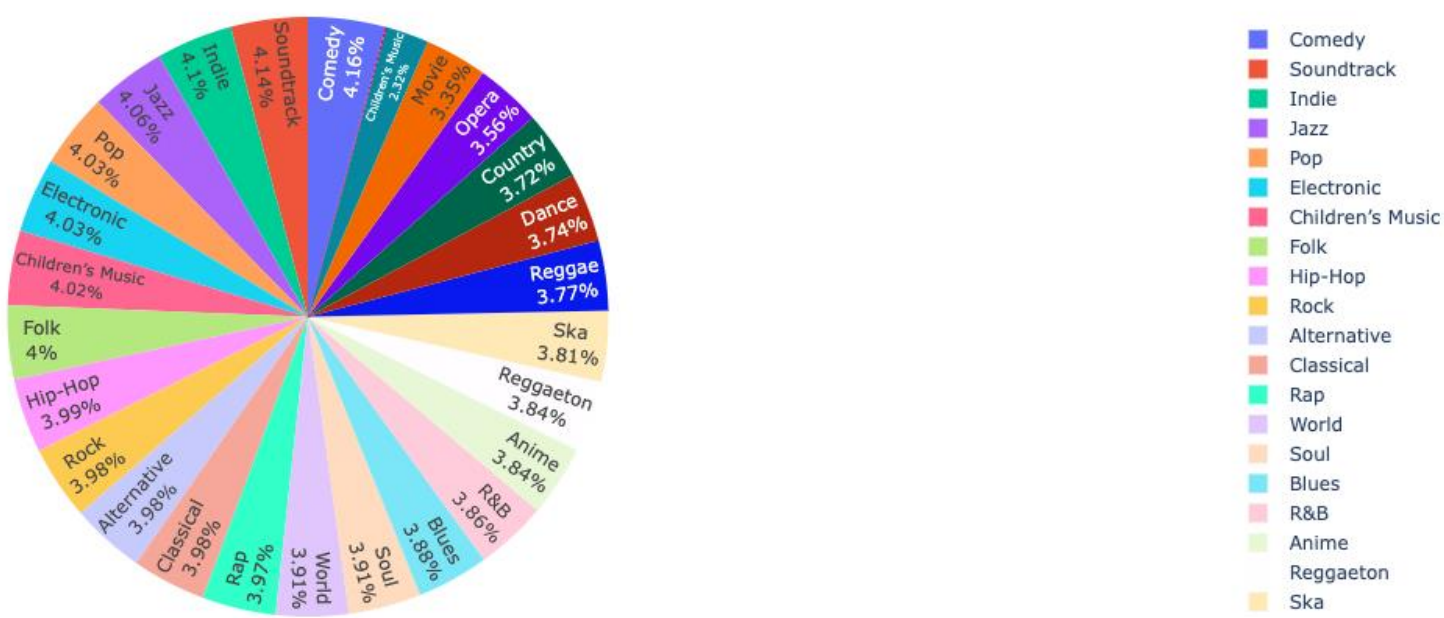


popularity
liveness
100
80
60
40
20
0
0.1
0.2
0.3
0.4
0.5
0.6

9) Bar Graph of 5 most Popular Songs vs Duration in milliseconds:

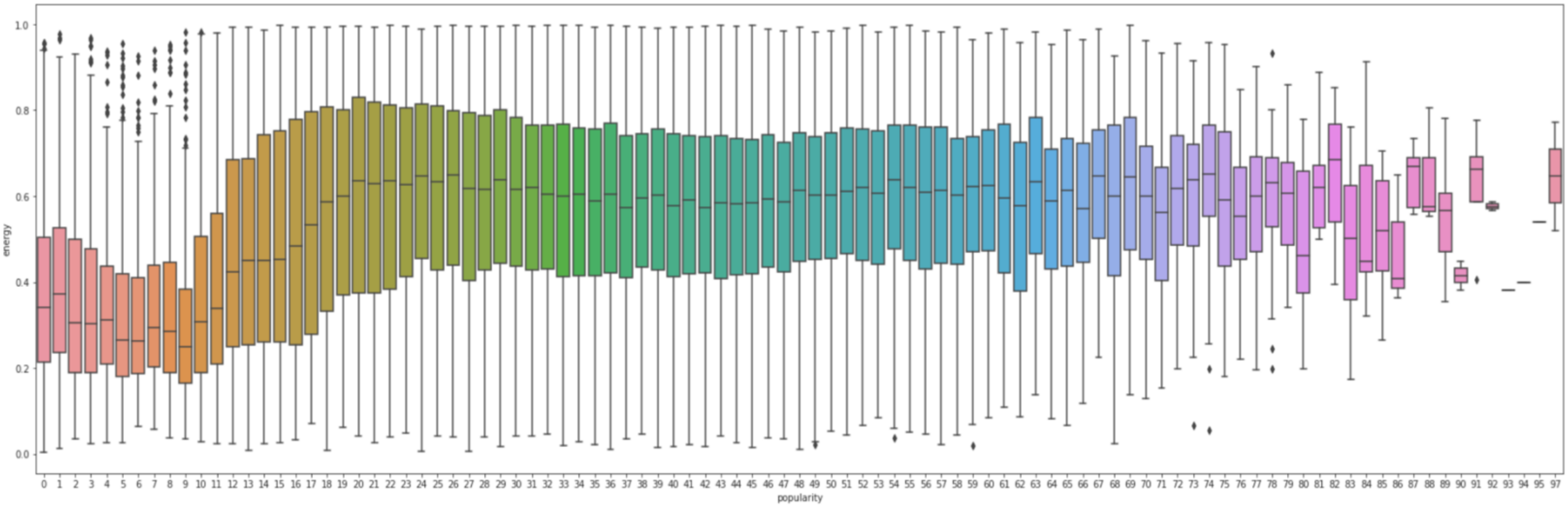


10) Different types of genres and their percentages:



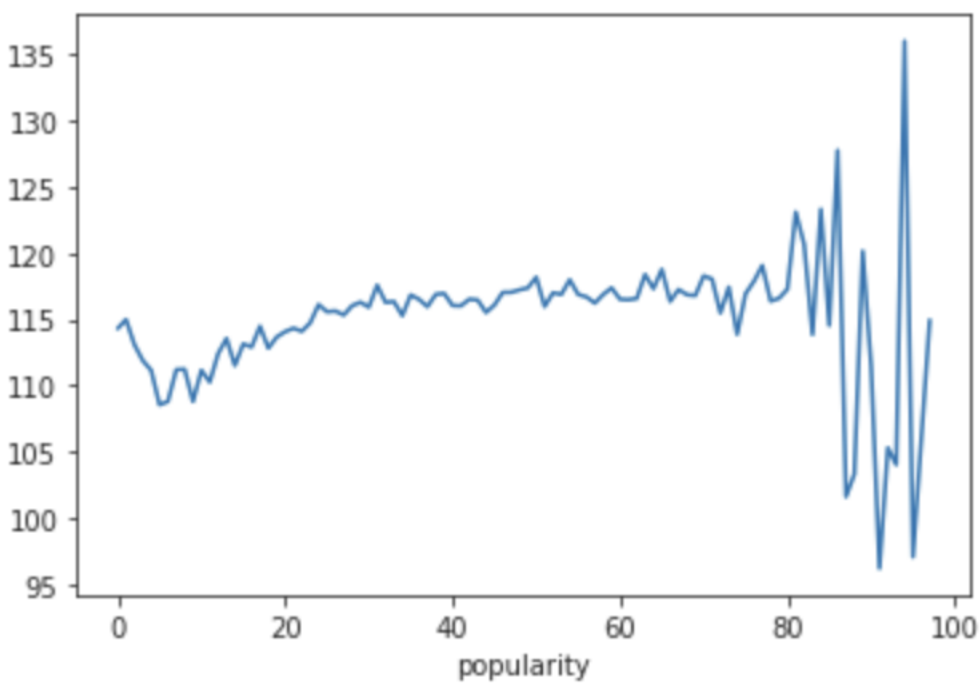
11) Popularity vs energy boxplot:

we can see that 4 of the top 5 popularity have energy constant.



12) Popularity vs Tempo:

from the mean plot we can see that the mean tempo for top popular artists is either low or high, where as its almost constant for less popular artists.



13) Popular Artists and their Instrumentalness:

we can observe that high popular artists have low instrumentalness.