Spotify_EDA_Project

April 16, 2023

0.1 Importing Libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
sns.set_style("darkgrid") #Setting the style as darkgrid
```

0.2 Loading Dataset

0.2.1 Spotify API Reference

https://developer.spotify.com/documentation/web-api/reference/get-audio-features

[3]:
```
df = pd.read_csv('data.csv')
```

[ ]:
```
df
```

[ ]:

	Unnamed: 0	acousticness	danceability	duration_ms	energy	\
0	0	0.01020	0.833	204600	0.434	
1	1	0.19900	0.743	326933	0.359	
2	2	0.03440	0.838	185707	0.412	
3	3	0.60400	0.494	199413	0.338	
4	4	0.18000	0.678	392893	0.561	
…	…	…	…	…	…	
2012	2012	0.00106	0.584	274404	0.932	
2013	2013	0.08770	0.894	182182	0.892	
2014	2014	0.00857	0.637	207200	0.935	
2015	2015	0.00164	0.557	185600	0.992	
2016	2016	0.00281	0.446	204520	0.915	

	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	\
0	0.021900	2	0.1650	-8.795	1	0.4310	150.062	
1	0.006110	1	0.1370	-10.401	1	0.0794	160.083	
2	0.000234	2	0.1590	-7.148	1	0.2890	75.044	
3	0.510000	5	0.0922	-15.236	1	0.0261	86.468	
4	0.512000	5	0.4390	-11.648	0	0.0694	174.004	

```
…                …  …       …       …  …           …       …
2012          0.002690    1    0.1290    -3.501     1       0.3330   74.976
2013          0.001670    1    0.0528    -2.663     1       0.1310  110.041
2014          0.003990    0    0.2140    -2.467     1       0.1070  150.082
2015          0.677000    1    0.0913    -2.735     1       0.1330  150.011
2016          0.000039    9    0.2180    -6.221     1       0.1410  190.013

      time_signature  valence  target                              song_title  \
0                4.0    0.286       1                               Mask Off
1                4.0    0.588       1                                 Redbone
2                4.0    0.173       1                            Xanny Family
3                4.0    0.230       1                          Master Of None
4                4.0    0.904       1                          Parallel Lines
…                …      …     …                                          …
2012             4.0    0.211       0   Like A Bitch - Kill The Noise Remix
2013             4.0    0.867       0                                 Candy
2014             4.0    0.470       0  Habit - Dack Janiels & Wenzday Remix
2015             4.0    0.623       0                          First Contact
2016             4.0    0.402       0                   I Wanna Get Better

                artist
0               Future
1     Childish Gambino
2               Future
3          Beach House
4          Junior Boys
…                  …
2012    Kill The Noise
2013    Dillon Francis
2014          Rain Man
2015         Twin Moons
2016          Bleachers

[2017 rows x 17 columns]
```

[5]:
```
#Droping the column as it same as index value and not important for the analysis

df.drop("Unnamed: 0", axis=1, inplace=True)
```

[6]:
```
df.head()
```

[6]:
```
   acousticness  danceability  duration_ms  energy  instrumentalness  key  \
0        0.0102         0.833       204600   0.434          0.021900    2
1        0.1990         0.743       326933   0.359          0.006110    1
2        0.0344         0.838       185707   0.412          0.000234    2
3        0.6040         0.494       199413   0.338          0.510000    5
4        0.1800         0.678       392893   0.561          0.512000    5
```

```
   liveness  loudness  mode  speechiness    tempo  time_signature  valence  \
0    0.1650    -8.795     1       0.4310  150.062             4.0    0.286
1    0.1370   -10.401     1       0.0794  160.083             4.0    0.588
2    0.1590    -7.148     1       0.2890   75.044             4.0    0.173
3    0.0922   -15.236     1       0.0261   86.468             4.0    0.230
4    0.4390   -11.648     0       0.0694  174.004             4.0    0.904

   target      song_title            artist
0       1        Mask Off            Future
1       1         Redbone  Childish Gambino
2       1    Xanny Family            Future
3       1  Master Of None       Beach House
4       1  Parallel Lines       Junior Boys
```

Data Cleaning

[7]:
```
#Checking for missing values

df.isna().sum()
```

[7]:
```
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
target              0
song_title          0
artist              0
dtype: int64
```

[8]:
```
df.info() #Print a concise summary of a DataFrame.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2017 entries, 0 to 2016
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   acousticness      2017 non-null   float64
 1   danceability      2017 non-null   float64
```

```
 2   duration_ms       2017 non-null   int64
 3   energy            2017 non-null   float64
 4   instrumentalness  2017 non-null   float64
 5   key               2017 non-null   int64
 6   liveness          2017 non-null   float64
 7   loudness          2017 non-null   float64
 8   mode              2017 non-null   int64
 9   speechiness       2017 non-null   float64
 10  tempo             2017 non-null   float64
 11  time_signature    2017 non-null   float64
 12  valence           2017 non-null   float64
 13  target            2017 non-null   int64
 14  song_title        2017 non-null   object
 15  artist            2017 non-null   object
dtypes: float64(10), int64(4), object(2)
memory usage: 252.2+ KB
```

[9]:
```
df.shape  #Dimensionality of the DataFrame.
```

[9]:
```
(2017, 16)
```

[68]:
```
df.size
```

[68]:
```
32272
```

[10]:
```
df.columns
```

[10]:
```
Index(['acousticness', 'danceability', 'duration_ms', 'energy',
       'instrumentalness', 'key', 'liveness', 'loudness', 'mode',
       'speechiness', 'tempo', 'time_signature', 'valence', 'target',
       'song_title', 'artist'],
      dtype='object')
```

[12]:
```
df.describe() #Generating descriptive statistics
```

[12]:
```
       acousticness  danceability   duration_ms       energy  \
count   2017.000000   2017.000000  2.017000e+03  2017.000000
mean       0.187590      0.618422  2.463062e+05     0.681577
std        0.259989      0.161029  8.198181e+04     0.210273
min        0.000003      0.122000  1.604200e+04     0.014800
25%        0.009630      0.514000  2.000150e+05     0.563000
50%        0.063300      0.631000  2.292610e+05     0.715000
75%        0.265000      0.738000  2.703330e+05     0.846000
max        0.995000      0.984000  1.004627e+06     0.998000

       instrumentalness          key     liveness     loudness         mode  \
count       2017.000000  2017.000000  2017.000000  2017.000000  2017.000000
```

mean	0.133286	5.342588	0.190844	-7.085624	0.612295
std	0.273162	3.648240	0.155453	3.761684	0.487347
min	0.000000	0.000000	0.018800	-33.097000	0.000000
25%	0.000000	2.000000	0.092300	-8.394000	0.000000
50%	0.000076	6.000000	0.127000	-6.248000	1.000000
75%	0.054000	9.000000	0.247000	-4.746000	1.000000
max	0.976000	11.000000	0.969000	-0.307000	1.000000

	speechiness	tempo	time_signature	valence	target
count	2017.000000	2017.000000	2017.000000	2017.000000	2017.000000
mean	0.092664	121.603272	3.968270	0.496815	0.505702
std	0.089931	26.685604	0.255853	0.247195	0.500091
min	0.023100	47.859000	1.000000	0.034800	0.000000
25%	0.037500	100.189000	4.000000	0.295000	0.000000
50%	0.054900	121.427000	4.000000	0.492000	1.000000
75%	0.108000	137.849000	4.000000	0.691000	1.000000
max	0.816000	219.331000	5.000000	0.992000	1.000000

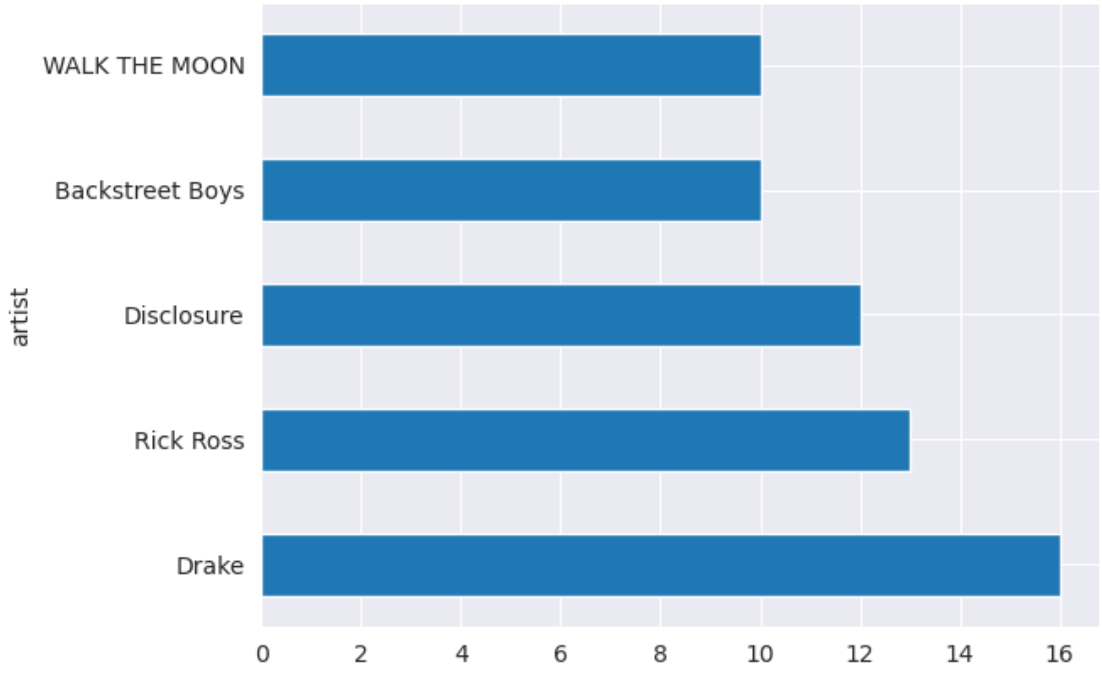
0.3 Data Analysis

0.3.1 Top 5 most popular artists

[13]:
```
top_five_artists = df.groupby("artist").count().sort_values(by = "song_title",␣
↪ascending = False)["song_title"][:5]
top_five_artists
```

[13]:
```
artist
Drake              16
Rick Ross          13
Disclosure         12
Backstreet Boys    10
WALK THE MOON      10
Name: song_title, dtype: int64
```

[14]:
```
top_five_artists.plot.barh()
plt.show()
```



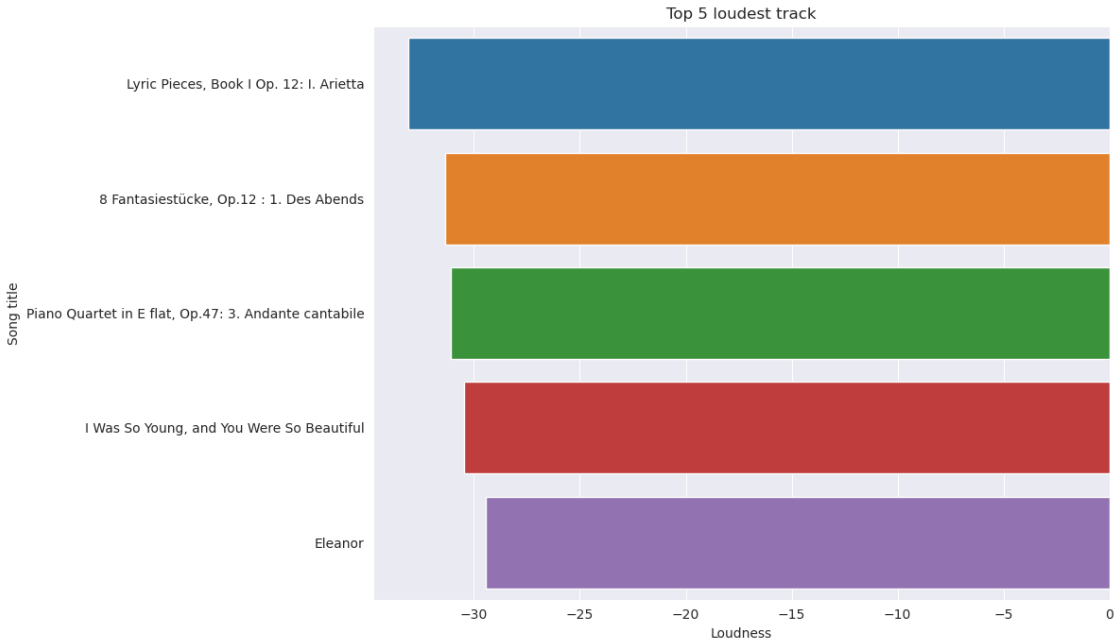
0.4 Top 5 loudest tracks

[15]:
```
top_five_loudest_tracks =  df[["loudness", "song_title"]].sort_values(by =␣
 ↪"loudness", ascending =  True)[:5]
top_five_loudest_tracks
```

[15]:

	loudness	song_title
1594	-33.097	Lyric Pieces, Book I Op. 12: I. Arietta
1596	-31.367	8 Fantasiestücke, Op.12 : 1. Des Abends
1598	-31.082	Piano Quartet in E flat, Op.47: 3. Andante can…
1531	-30.447	I Was So Young, and You Were So Beautiful
1549	-29.460	Eleanor

[16]:
```
plt.figure(figsize = (12,7))
sns.barplot(x ="loudness", y = "song_title", data = top_five_loudest_tracks )
plt.title("Top 5 loudest track")
plt.xlabel("Loudness")
plt.ylabel("Song title")
plt.tight_layout()
plt.savefig("top_five_loudest_tracks.png")
plt.show()

#More negative in values means the loudest
```



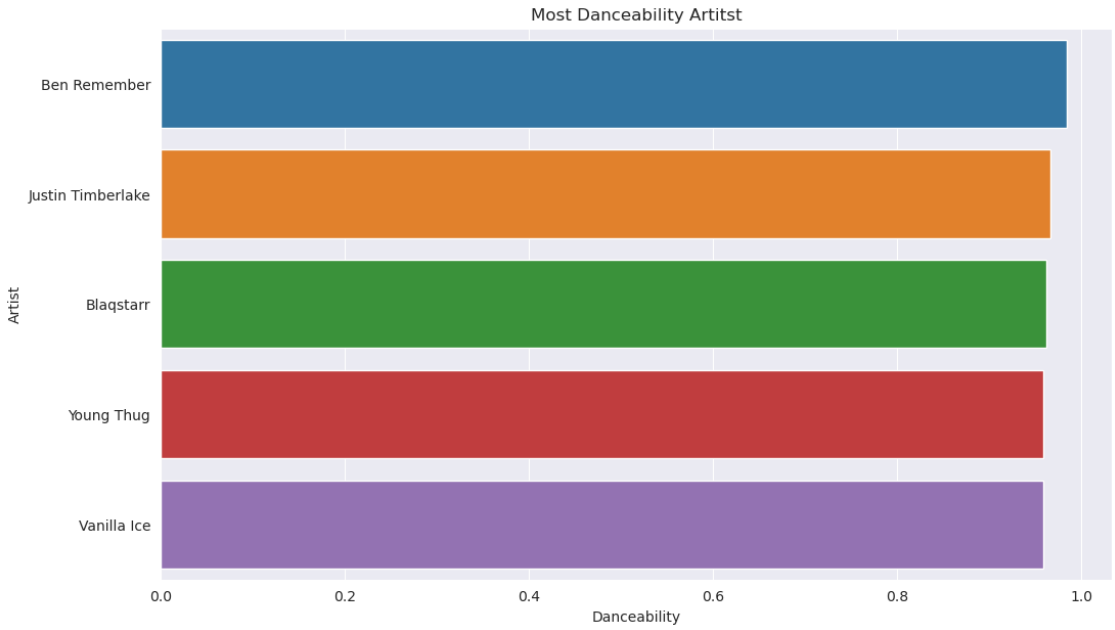
0.5 Artist with the most danceability song

[17]:
```
top_five_artists_danceable_songs = df[["danceability","song_title","artist"]].
 ↪sort_values(by = "danceability", ascending =  False)[:5]
top_five_artists_danceable_songs
```

[17]:

	danceability	song_title	artist
1433	0.984	Flashwind - Radio Edit	Ben Remember
1901	0.967	SexyBack	Justin Timberlake
604	0.962	Check Me Out Like	Blaqstarr
32	0.959	Best Friend	Young Thug
1957	0.959	Ice Ice Baby	Vanilla Ice

[18]:
```
plt.figure(figsize = (12,7))
sns.barplot(x ="danceability", y = "artist", data =␣
 ↪top_five_artists_danceable_songs)
plt.title("Most Danceability Artitst")
plt.xlabel("Danceability")
plt.ylabel("Artist")
plt.savefig("Most Danceability Artitst")
plt.xlabel("Danceability")
plt.show()


#Closer to 1 mean highest and vice versa
```



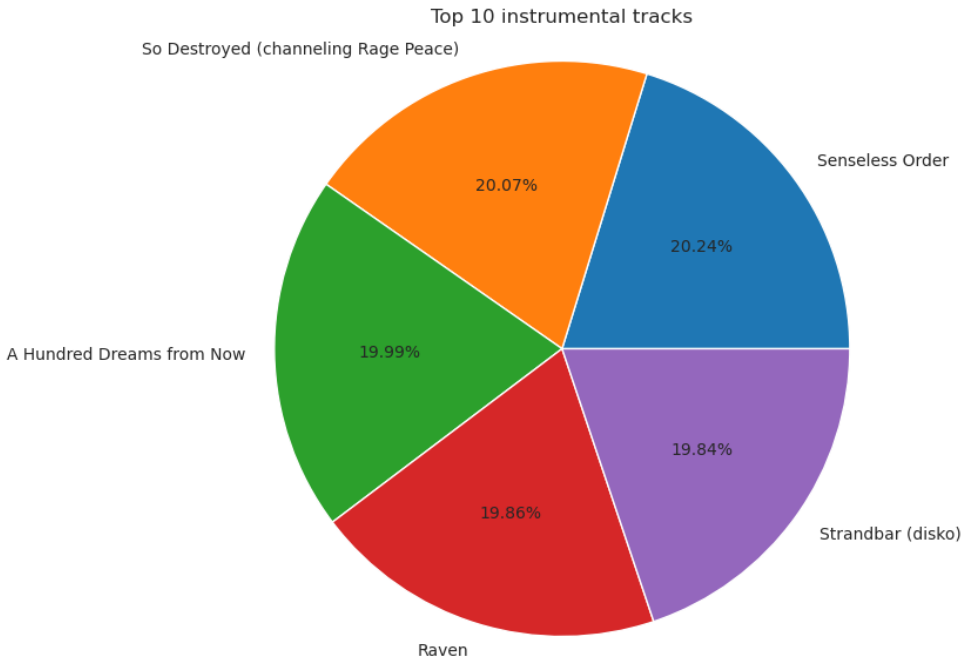
0.6 Top 10 instrumental tracks

[19]:
```
top_ten_instrumental_tracks = df[["instrumentalness", "song_title", "artist"]].
  ↪sort_values(by = "instrumentalness", ascending =  False)[:5]
top_ten_instrumental_tracks
```

[19]:
```
      instrumentalness                          song_title  \
1313             0.976                      Senseless Order
271              0.968  So Destroyed (channeling Rage Peace)
1575             0.964           A Hundred Dreams from Now
1619             0.958                               Raven
725              0.957                   Strandbar (disko)

                 artist
1313  Signs of the Swarm
271          Prince Rama
1575          Ray Bryant
1619       John Dahlbäck
725           Todd Terje
```

[20]:
```
plt.figure(figsize = (12,7))
plt.pie("instrumentalness", data = top_ten_instrumental_tracks, autopct = '%1.
  ↪2f%%', labels = top_ten_instrumental_tracks.song_title)
plt.title("Top 10 instrumental tracks")
plt.axis('equal')
plt.savefig("Top 10 instrumental tracks")
```

```
plt.show()
```



0.7 Multiple feature Plots

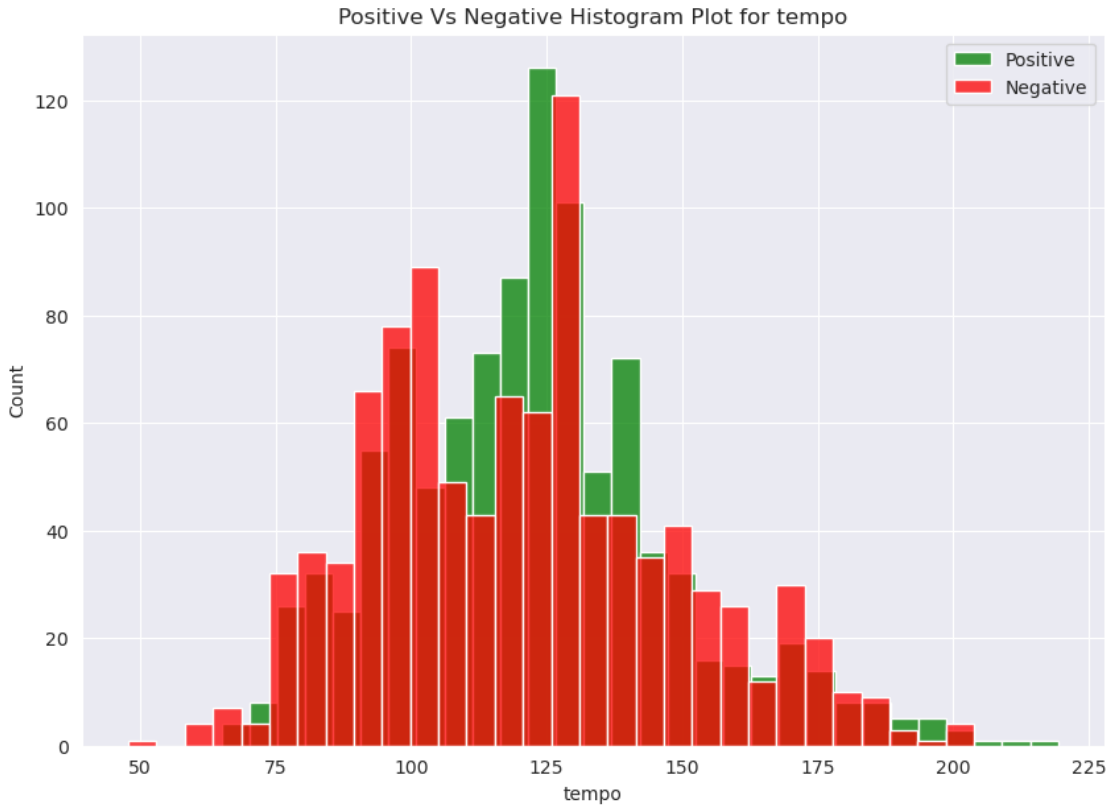
[21]:
```
df.columns
```

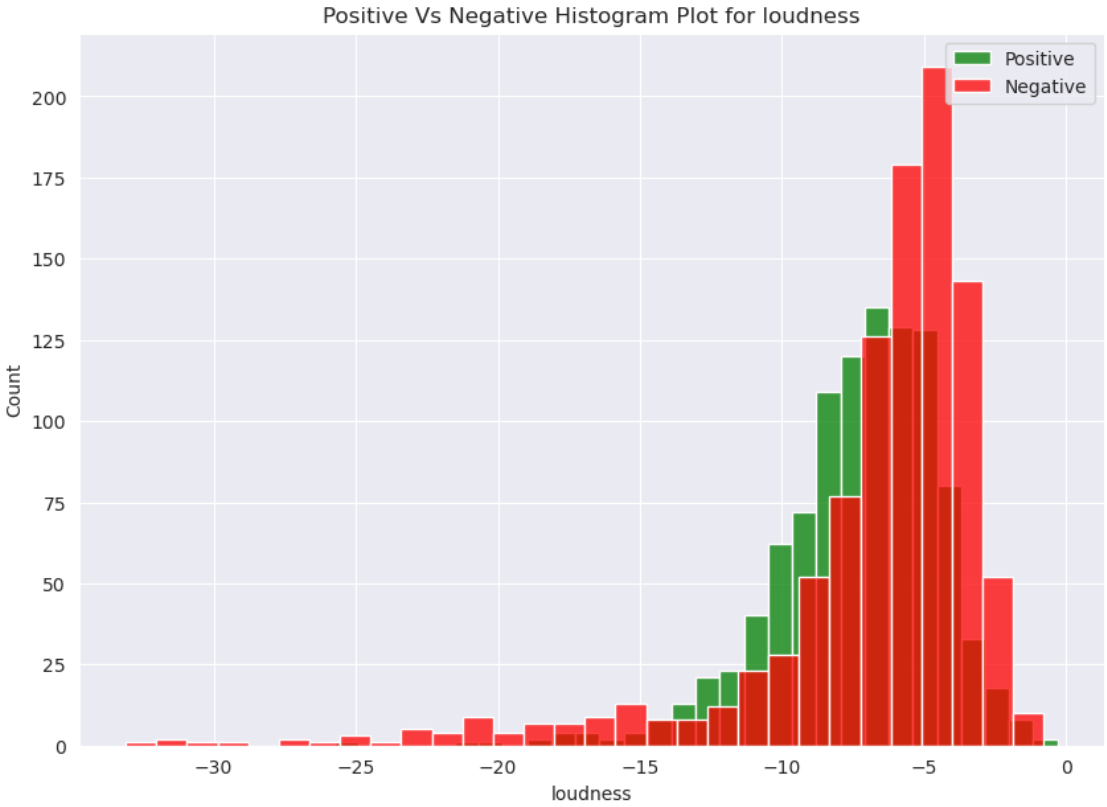
[21]:
```
Index(['acousticness', 'danceability', 'duration_ms', 'energy',
       'instrumentalness', 'key', 'liveness', 'loudness', 'mode',
       'speechiness', 'tempo', 'time_signature', 'valence', 'target',
       'song_title', 'artist'],
      dtype='object')
```

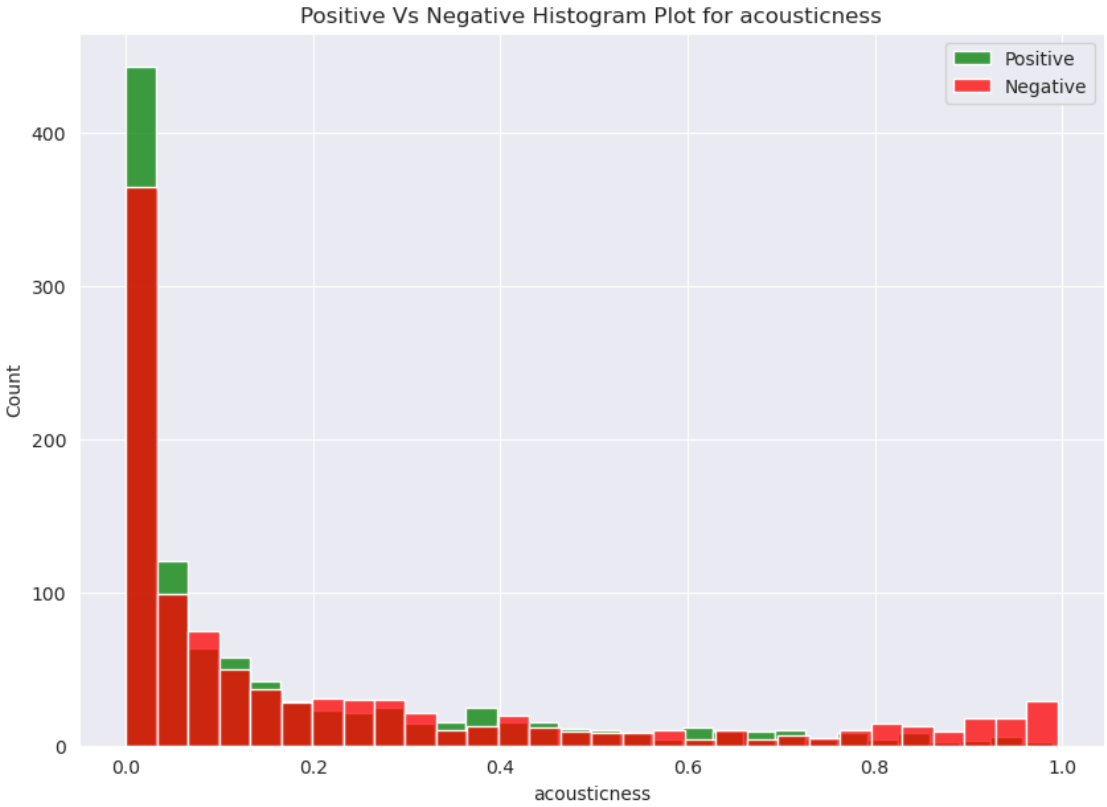
[22]:
```
Interested_features =␣
↪['tempo','loudness','acousticness','danceability','duration_ms','energy','instrumentalness'
                      'liveness','speechiness','valence']
```

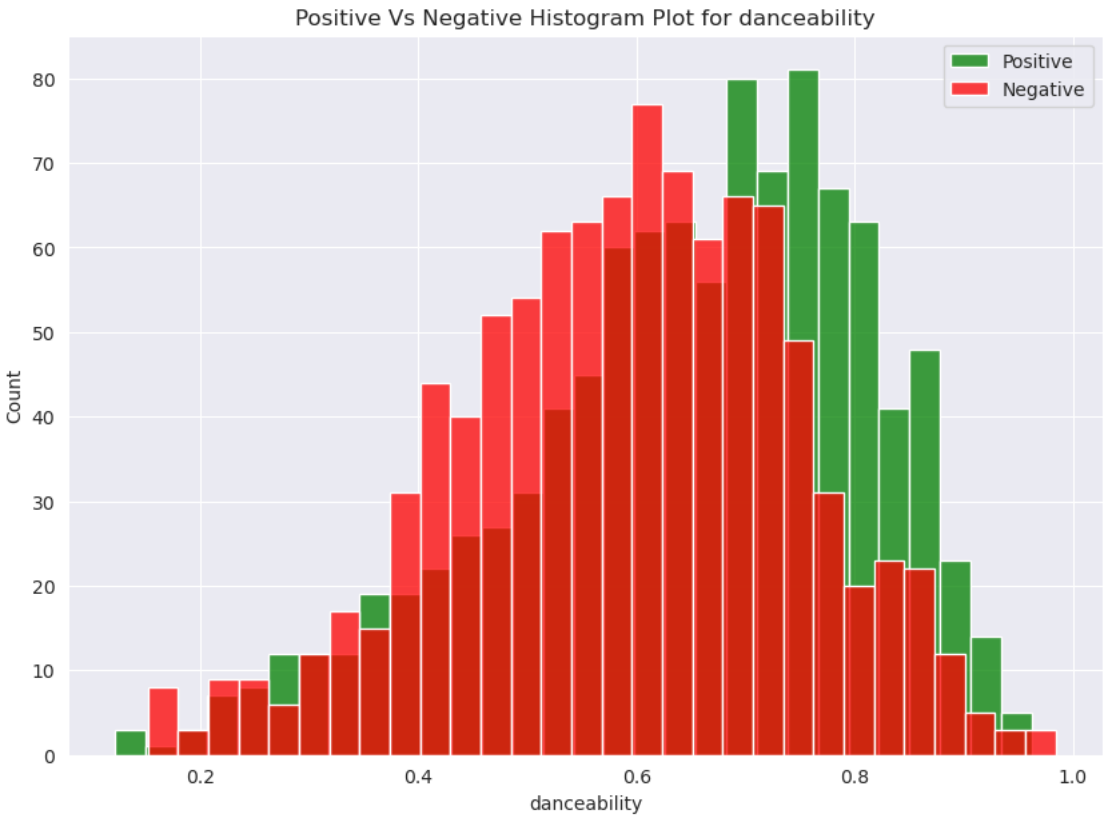
[23]:
```
#Using a for loop to plot visuals for all interested features

for feature_col in Interested_features:
    positive_data = df[df["target"]==1] [feature_col]
    negative_data = df[df["target"]==0] [feature_col]

    plt.figure(figsize = (10,7)) # Giving dimensions for the figures
```

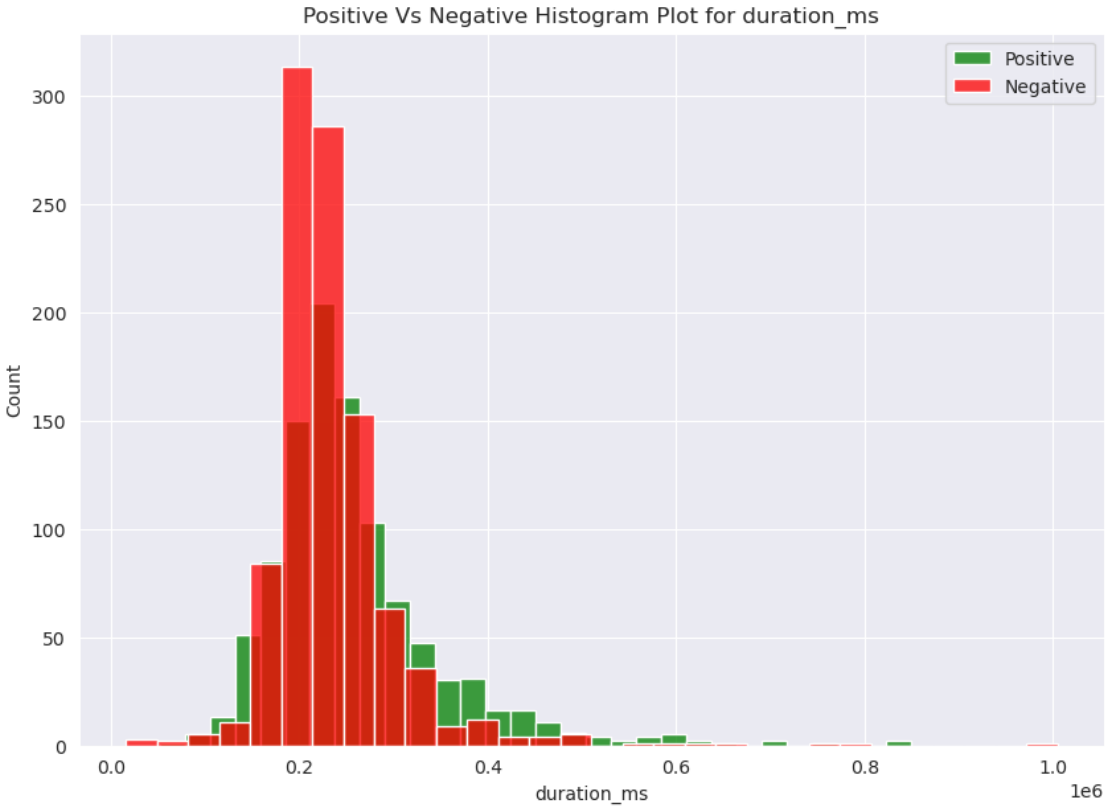
```
   sns.histplot(positive_data, bins = 30, label = "Positive", color = "green")␣
↪#Giving features
   sns.histplot(negative_data, bins = 30, label = "Negative", color = "red")

   #sns.distplot(positive_data, bins = 30, label = "Positive", color =␣
↪"green",kde=True) #Giving features
   #sns.distplot(negative_data, bins = 30, label = "Negative", color =␣
↪"red",kde=True)



   #Decorating the figures/Graphs
   plt.legend(loc = "upper right")
   plt.title(f"Positive Vs Negative Histogram Plot for {feature_col}")
   plt.show()
```

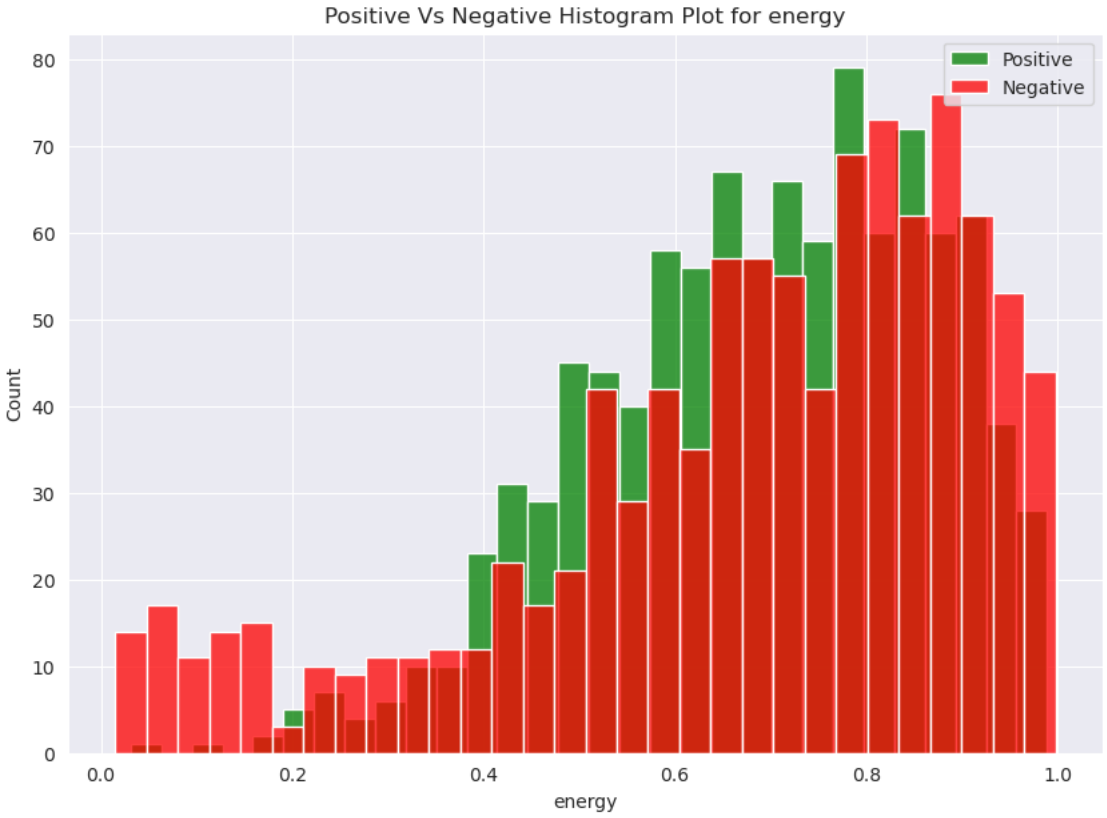


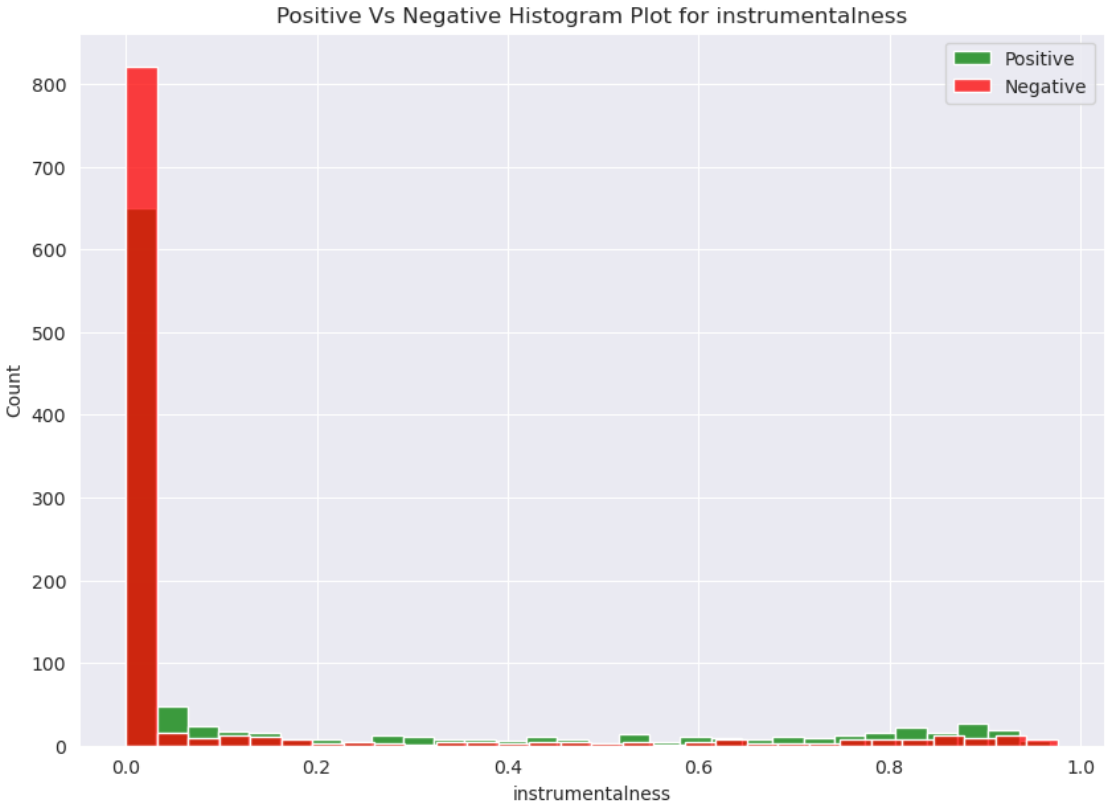

Positive Vs Negative Histogram Plot for loudness
Positive
Negative
Count
200
175
150
125
100
75
50
25
0
−30
−25
−20
−15
−10
−5
0
loudness

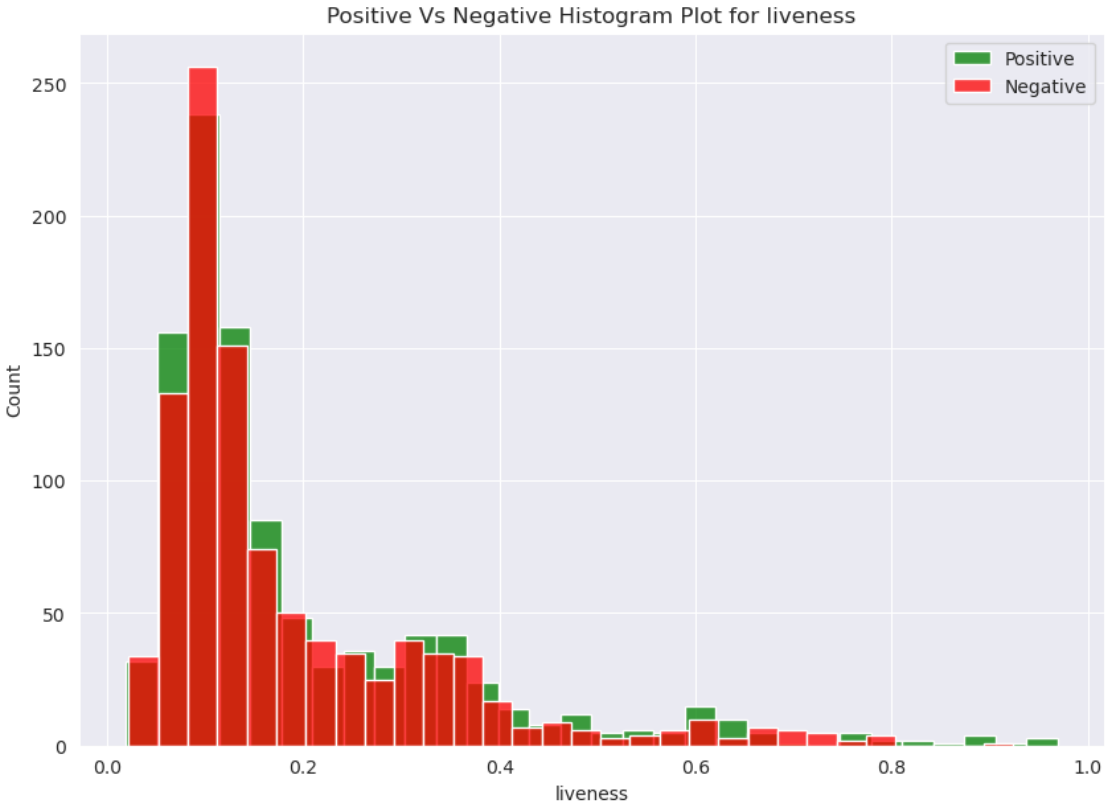

Positive Vs Negative Histogram Plot for acousticness
Positive
Negative
Count
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
acousticness

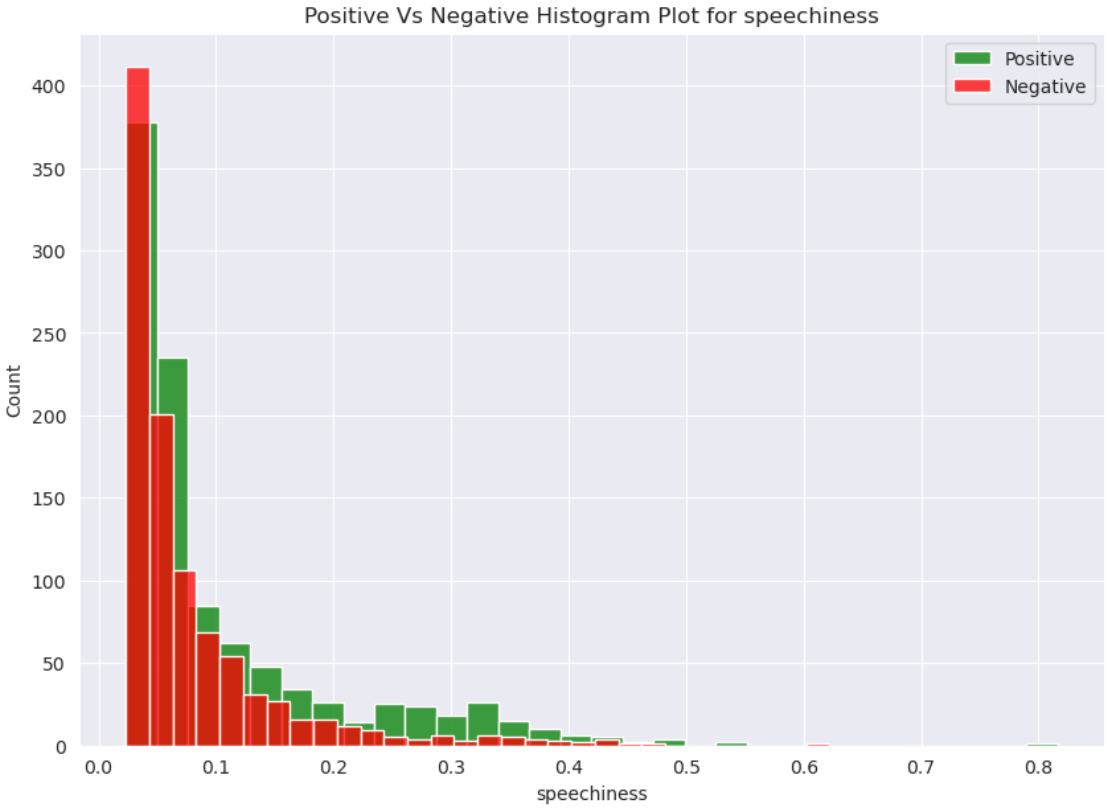

Positive Vs Negative Histogram Plot for danceability
Positive
Negative
Count
0
10
20
30
40
50
60
70
80
0.2
0.4
0.6
0.8
1.0
danceability

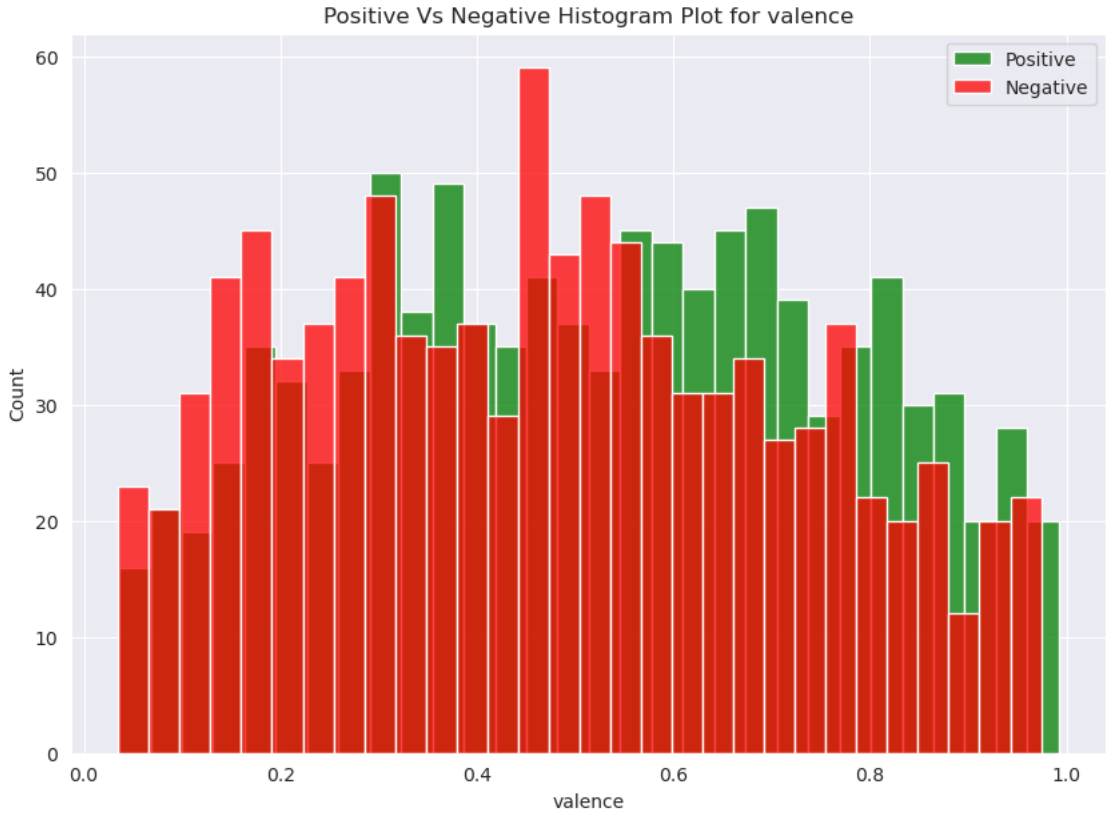

Positive Vs Negative Histogram Plot for duration_ms
Positive
Negative
Count
300
250
200
150
100
50
0
0.0
0.2
0.4
0.6
0.8
1.0
1e6
duration_ms


Positive Vs Negative Histogram Plot for energy
Positive
Negative
Count
energy
0
10
20
30
40
50
60
70
80
0.0
0.2
0.4
0.6
0.8
1.0


Positive Vs Negative Histogram Plot for instrumentalness
Positive
Negative
Count
800
700
600
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
instrumentalness


Positive Vs Negative Histogram Plot for liveness
Positive
Negative
Count
250
200
150
100
50
0
0.0
0.2
0.4
0.6
0.8
1.0
liveness


Positive Vs Negative Histogram Plot for speechiness
Positive
Negative
Count
400
350
300
250
200
150
100
50
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
speechiness



0.8 Most Trending Genre

[24]:
```
df.columns
```

[24]:
```
Index(['acousticness', 'danceability', 'duration_ms', 'energy',
       'instrumentalness', 'key', 'liveness', 'loudness', 'mode',
       'speechiness', 'tempo', 'time_signature', 'valence', 'target',
       'song_title', 'artist'],
      dtype='object')
```

[25]:
```
Most_trending_genre =  df[["artist", "song_title"]].sort_values(by = "artist",␣
 ↪ascending =  True)[:5]
Most_trending_genre
```

[25]:

	artist	song_title
388	!!!	Slyd
1739	*NSYNC	Pop - Radio Version
1740	*NSYNC	Tearin' up My Heart - Radio Edit
1738	*NSYNC	I Want You Back - Radio Edit
1737	*NSYNC	God Must Have Spent a Little More Time on You …

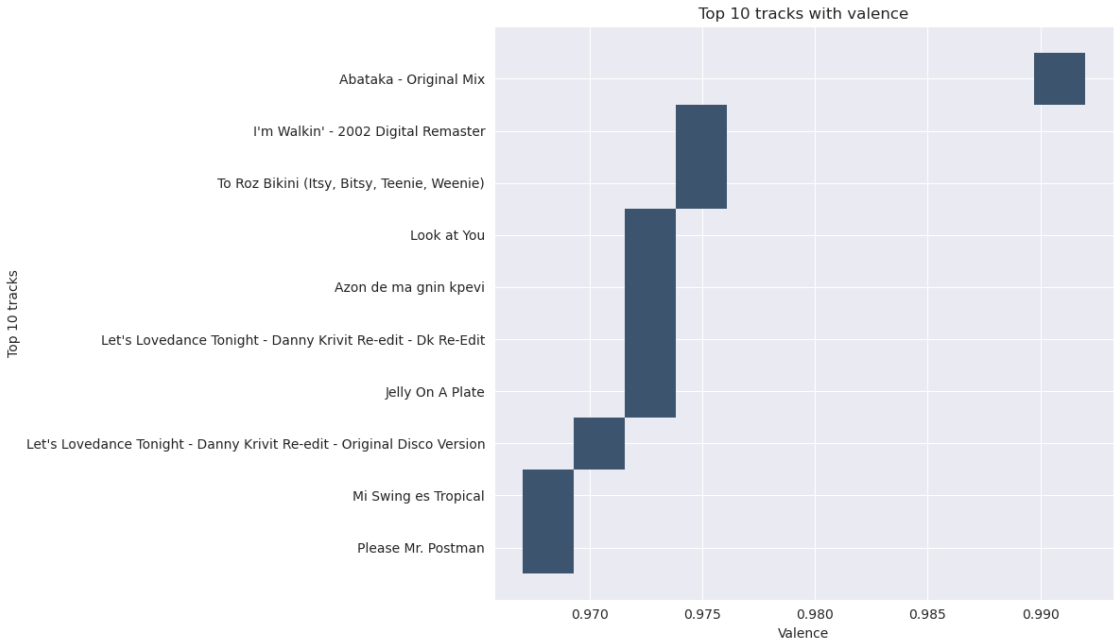
0.9 Top 10 tracks with most valence

[26]:
```
top_tracks_valence = df[["valence", "song_title"]].sort_values(by = "valence",␣
 ↪ascending =  False)[:10]

"""
Describing the musical positiveness conveyed by a track.
Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric),␣
 ↪while tracks with low valence sound more negative (e.g. sad, depressed,␣
 ↪angry).
Example value: 0.428
Range: 0 - 1

"""

top_tracks_valence
```

[26]:

	valence	song_title
460	0.992	Abataka - Original Mix
912	0.975	I'm Walkin' - 2002 Digital Remaster
1966	0.974	To Roz Bikini (Itsy, Bitsy, Teenie, Weenie)
207	0.973	Look at You
48	0.973	Azon de ma gnin kpevi
337	0.972	Let's Lovedance Tonight - Danny Krivit Re-edit…
1590	0.972	Jelly On A Plate
838	0.971	Let's Lovedance Tonight - Danny Krivit Re-edit…
497	0.968	Mi Swing es Tropical
112	0.967	Please Mr. Postman

[27]:
```
plt.figure(figsize = (12,7))
sns.histplot(y ="song_title", x = "valence", data = top_tracks_valence)
plt.title("Top 10 tracks with valence")
plt.ylabel("Top 10 tracks")
plt.xlabel("Valence")
plt.tight_layout()
plt.savefig("top_tracks_valence.png")
plt.show()
```



0.10 Top 10 energetic & Least energetic tracks

[28]:
```
df.columns
```

[28]:
```
Index(['acousticness', 'danceability', 'duration_ms', 'energy',
       'instrumentalness', 'key', 'liveness', 'loudness', 'mode',
       'speechiness', 'tempo', 'time_signature', 'valence', 'target',
       'song_title', 'artist'],
      dtype='object')
```

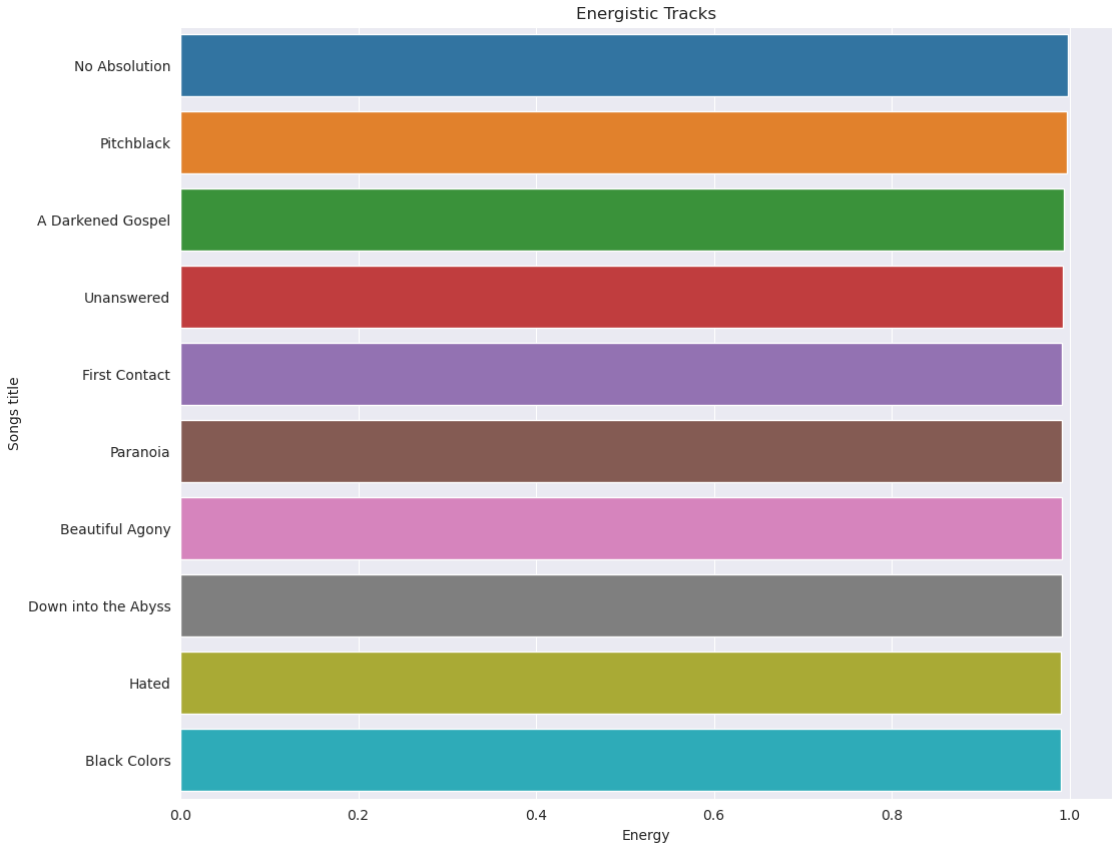
[29]:
```
top_energetic_tracks = df [["song_title", "energy"]].sort_values(by = "energy",␣
 ↪ascending =  False)[:10]

"""
Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of␣
 ↪intensity and activity.
Typically, energetic tracks feel fast, loud, and noisy. For example, death␣
 ↪metal has high energy, while a Bach prelude scores low on the scale.

"""

top_energetic_tracks
```

[29]:

	song_title	energy
1299	No Absolution	0.998

```
1322           Pitchblack   0.997
1297    A Darkened Gospel   0.994
1347           Unanswered   0.993
2015        First Contact   0.992
1680             Paranoia   0.992
1332      Beautiful Agony   0.992
1328  Down into the Abyss   0.991
1681                Hated   0.990
1296         Black Colors   0.990
```

[30]:
```
plt.figure(figsize = (12,10))
sns.barplot(x = "energy", y = "song_title", data = top_energetic_tracks)
plt.title('Energistic Tracks')
plt.xlabel('Energy')
plt.ylabel('Songs title')
plt.show()
```

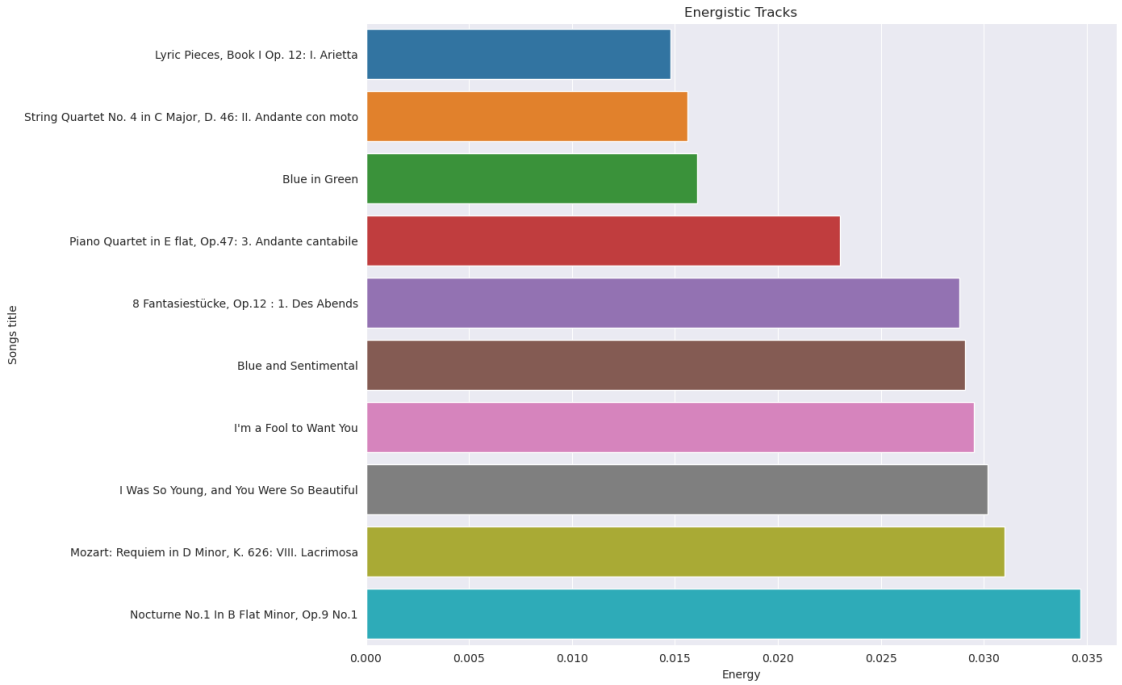


[31]:
```
top_least_energetic_tracks = df [["song_title", "energy"]].sort_values(by =␣
 ↪"energy", ascending =  True)[:10]
top_least_energetic_tracks
```

[31]:

	song_title	energy
1594	Lyric Pieces, Book I Op. 12: I. Arietta	0.0148
1595	String Quartet No. 4 in C Major, D. 46: II. An…	0.0156
1537	Blue in Green	0.0161
1598	Piano Quartet in E flat, Op.47: 3. Andante can…	0.0230
1596	8 Fantasiestücke, Op.12 : 1. Des Abends	0.0288
1545	Blue and Sentimental	0.0291
1530	I'm a Fool to Want You	0.0295
1531	I Was So Young, and You Were So Beautiful	0.0302
817	Mozart: Requiem in D Minor, K. 626: VIII. Lacr…	0.0310
1876	Nocturne No.1 In B Flat Minor, Op.9 No.1	0.0347

[32]:
```
plt.figure(figsize = (12,10))
sns.barplot(x = "energy", y = "song_title", data = top_least_energetic_tracks)
plt.title('Energistic Tracks')
plt.xlabel('Energy')
plt.ylabel('Songs title')
plt.show()
```



0.11 Danceability Vs Tempo insights

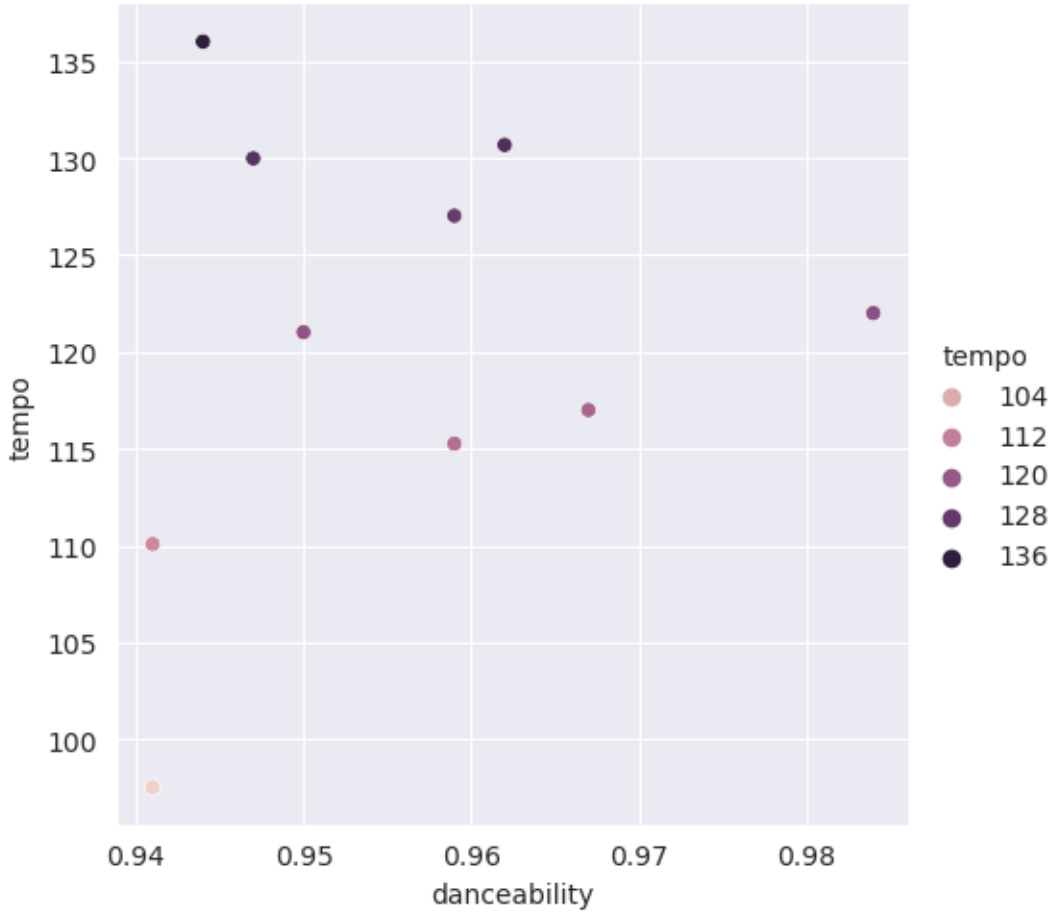
[58]:
```
Danceability_insights = df[["danceability", "tempo"]].sort_values(by =␣
 ↪"danceability", ascending = False)[:10]
Danceability_insights
```

[58]:

	danceability	tempo
1433	0.984	122.003
1901	0.967	117.000
604	0.962	130.684
32	0.959	127.029
1957	0.959	115.266
1136	0.950	121.019
1977	0.947	129.991
138	0.944	136.019
623	0.941	97.524
1591	0.941	110.082

[66]:
```
sns.relplot(x = df.danceability, y = df.tempo, data = Danceability_insights,␣
↪hue = "tempo")

# Danceability of the song is high when tempo is between mid range. Higher the␣
↪tempo lower the danceability
```

[66]: <seaborn.axisgrid.FacetGrid at 0x7f4fc1789210>



0.12 Thankyou for reading. Happy Coding! (~iamkd99)

[ ]: