SPOTIFY DATA ANALYSIS &

EXPLORATORY DATA ANALYSIS

Summary:

Spotify is the largest music streaming service available. The company started in 2006 in a time when piracy caused considerable losses to the music industry. In January 2015 they had 60 million users in total of which 15 million premium users (1) and these numbers seem to be increasing.

Spotify offers free streaming of music to its users, though one can purchase a premium membership for added benefits, such as no advertisements and being able to listen to music offline.

The large number of users and content of Spotify create a large database of users and songs that users listened to that could hold interesting patterns and information for related companies, such as Spotify themselves, record companies or radio stations. The dataset in question has been provided by <Undisclosed company>, so we first look for general applications of the data and then focus on possibilities that will also be useful to <Undisclosed company>, but will also be interesting from a scientific perspective

Introduction:

Spotify is a Swedish-based audio streaming and media service provider. It is now one of the most prominent digital music, podcast, and video streaming services, giving access to millions of songs from artists worldwide. Spotify is one of the largest music streaming service providers worldwide. As a freemium service, it has free features with advertisements and limited control and additional features such as offline listening and commercial-free listening, which are offered via paid subscriptions [1]. Subscribing to Spotify Premium is a great way to get rid of pesky ads, but there is more to the paid version of the service such as no advertisement anywhere, better audio quality, and downloading songs [2]. Users can search for music based on artist, genre, and popularity and create playlists. Not only does Spotify give us access to good songs on multiple platforms, but it has also exposed everyone to trending and upcoming artists from various genres that we had never experienced

Spotify's Objectives :

Unlocking human creativity: Spotify's mission is to unlock the potential of human creativity by giving artists the opportunity to live off their art and fans the opportunity to enjoy and be inspired by it. Becoming the world's number one audio

platform: Spotify's goal is to become the world's number one global audio platform.

Providing resources for artists and creators: Spotify aims to provide infrastructure and resources for artists and creators to grow and manage their businesses.

Achieving global equality and harmony: Spotify believes that allowing people from all walks of life access to music can bring about positive change in society.

Reaching one billion users by 2030: Spotify aims to have one billion users by 2030. Achieving $100 billion in annual revenue: Spotify aims to achieve an annual revenue of $100 billion in a decade.

Spotify's Purpose:

Spotify's purpose is to revolutionize audio and become the world's leading audio platform. Spotify's mission is to offer access to millions of songs, podcasts, and videos from creators around the world. Spotify's services include:

Music and podcasts: Spotify offers access to over 100 million tracks and more than 6 million podcasts.

Recommendations: Spotify can provide recommendations based on your taste.

Playlists: Spotify allows you to create playlists or use playlists made by music experts.

Audiobooks: Spotify offers access to over 350,000 audiobooks.

Data saver: Spotify allows you to save mobile data by turning on Data Saver in Settings.

Spotify is available on a variety of devices, including computers, phones, tablets, speakers, TVs, and cars. Spotify offers a basic free service, as well as a premium subscription service

DESCRIPTION OF DATASET

DESCRIPTION

Introduction and Objectives: The Spotify EDA project is designed to delve deep into the Spotify content dataset, aiming to gain comprehensive insights into the platform's content landscape. The primary objective is to understand, analyze, and derive meaningful insights from the dataset to inform strategic decisions, optimize content strategies, and enhance user experience on the platform.

Data Exploration and Cleaning: The project commences with a meticulous data exploration phase to understand the dataset's structure, variables, and general patterns. This involves identifying and handling missing values, duplicates, and inconsistencies to ensure data integrity and reliability.

Descriptive Statistics: Through descriptive statistics computation, the project aims to provide a quantitative perspective on the dataset. Key metrics such as mean, median, mode, range, and standard deviation will be calculated for relevant variables to gain insights into the content's distribution, characteristics, and variability.

Data Visualization: The project includes creating compelling visualizations to represent the distribution of content across different genres, release years, and potentially geographical locations. These visualizations facilitate intuitive understanding, pattern recognition, and insight generation from the dataset.

Time Series Analysis: With a temporal component in the dataset, the project conducts time series analysis to identify evolving trends, patterns, and fluctuations over time, shedding light on the dynamic nature of Spotify content offerings.

Content Analysis: The project delves into content- specific analyses, exploring attributes such as ratings, duration, and content variety. This analysis offers insights into the diversity, popularity, and characteristics of the content available on the platform.

Audience Engagement Analysis: The project assesses audience engagement through user reviews, sentiment analysis, and engagement metrics like views or watch time. This analysis provides valuable data on audience behavior, interaction, and sentiment towards Spotify content.

Conclusions and Recommendations: In conclusion, the project synthesizes the insights gained from the analysis to draw meaningful conclusions and provide actionable recommendations. These recommendations aim to enhance Spotify's content offerings, user experience, and engagement on the platform, based on the identified insights and trends.

Technologies Used:

• JuypterNotebook: JuypterNotebook was used for data preprocessing, analysis, and visualization. Libraries such as Python, Pandas, NumPy, Seaborn, Plotly and Matplotlib were utilized for these tasks.

Library Used:

Python

NumPy

Seaborn,

Plotly

Matplotlib

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import datetime
import warnings
warnings.filterwarnings('ignore')
```

Data Set File Name:

Data .Csv

Data By Artist . Csv

Data By Genres. Csv

Data By Year. Csv

Data w Genres. Csv

```
df_data = pd.read_csv('data.csv')
df_artist = pd.read_csv('data_by_artist.csv')
df_by_genres = pd.read_csv('data_by_genres.csv')
df_year = pd.read_csv('data_by_year.csv')
df_w_genres = pd.read_csv('data_w_genres.csv')
```

Components:

Functions:

Read() method returns the specified number of bytes from the file. Default is -1 which means the whole file.

Head function in Python displays the first five rows of the dataframe by default. It takes in a single parameter: the number of rows. We can use this parameter to display the number of rows of our choice.

In Python, the shape attribute is commonly used with NumPy arrays and Pandas DataFrames to get the dimensions of the data structure.

Sample() method returns a list with a specified number of randomly selected items from a sequence.

Info() function is commonly used with Pandas DataFrames. It provides a concise summary of the DataFrame, including:

- ***Index: The type and range of the index.***
- ***Columns: The names and data types of each column.***
- ***Non-null values: The number of non-null values in each column.***

- ***Memory usage: The amount of memory used by the DataFrame.***

Describe() method returns description of the data in the DataFrame. If the DataFrame contains numerical data, the description contains these information for each column: count – The number of not-empty values. mean – The average (mean) value. std – The standard deviation.

Transpose() function changes the row elements into column elements and the column elements into row elements. The output of this function is a modified array of the original one.

Isnull() method returns a DataFrame object where all the values are replaced with a Boolean value True for NULL values, and otherwise False.

Duplicated() method returns a Series with True and False values that describe which rows in the DataFrame are duplicated and not.

Nunique() function is used to count the number of unique (non-null) values in a Series or DataFrame column.

Sort values() function sorts values in a DataFrame along the selected axis and returns a DataFrame with sorted values or None.

GroupBy is used to separate identical data into groups to allow for further aggregation and analysis.

Replace() method replaces a specified phrase with another specified phrase. Note: All occurrences of the specified phrase will be replaced, if nothing else is specified.

Corr() finds the correlation between every column(variable) in the dataframe with one another, it returns a 2D-Datamatrix. The data values are represented as colors in the heatmap.

Drop() method removes the specified row or column. By specifying the column axis (axis='columns'), the drop() method removes the specified column. By specifying the row axis (axis='index'), the drop() method removes the specified row.

Data Visualization:

Heatmap depicts values for a main variable of interest across two axis variables as a grid of colored squares. The axis variables are divided into ranges like a bar chart or histogram, and each cell's color indicates the value of the main variable in the corresponding cell range.

Bar chart plots numeric values for levels of a categorical feature as bars. Levels are plotted on one chart axis, and values are plotted on the other axis. Each categorical value claims one bar, and the length of each bar corresponds to the bar's value.

Scatter plot identifies a possible relationship between changes observed in two different sets of variables. It provides a visual and statistical means to test the strength of a relationship between two variables.

Regplot is used to plot data and a linear regression model fit. There are a number of mutually exclusive options for estimating the regression model.

Line plot is a graphical display of data along a number line with Xs or dots recorded above the responses to indicate the number of occurrences a

response appears in the data set. The Xs or dots represent the frequency. A line plot will have an outlier.

Pair plot, also known as a scatterplot matrix, is a matrix of graphs that enables the visualization of the relationship between each pair of variables in a dataset. It combines both histogram and scatter plots, providing a unique overview of the dataset's distributions and correlations.

Distplot or distribution plot, depicts the variation in the data distribution. Seaborn Distplot represents the overall distribution of continuous data variables. The Seaborn module along with the Matplotlib module is used to depict the distplot with different variations in it.

Histogram method helps to visualize dataset distributions. We can draw either univariate or bivariate histograms. A histogram is a traditional visualization tool that counts the number of data that fall into discrete bins to illustrate the distribution of one or more variables.

Box plots are used to show distributions of numeric data values, especially when you want to compare them between multiple groups. They are built to provide high-level information at a glance, offering general information about a group of data's symmetry, skew, variance, and outliers.

Detail Information of Data Set with Coding Screenshot:

Shape Of Data Set:

Data – 170653 ,19

Artist- 28680,15

Genres-2973,14

Year-100,14

W_genres – 28680,16

Inspect and Cleaning:

Checking Null Values:

Data – 0 null values

Artist – 0 null values

Genres – 0 null values

Year – 0 null values

W_genres- 0 null values

Duplicated Values-

Data – 543 values

Artist- 0 values

Genres – 0 Values

Year- 0 values

W_genres – 0 values

Using Drop Command:

```
df_data.drop('id',axis=1,inplace=True)
```

```
[59]: df_artist.drop('mode',axis=1,inplace=True)
```

```
df_by_genres.drop('key',axis=1,inplace=True)
```

Using this command in the data,artist,genres csv file to remove the id, mode, key column in the csv file.

Str Replace Command:

Genres

```
df_by_genres['genres'] = df_by_genres['genres'].str.replace("'","")
df_by_genres['genres'] = df_by_genres['genres'].str.replace("[","")
df_by_genres['genres'] = df_by_genres['genres'].str.replace("]","")

df_by_genres.head(100)
```

Artist

```
[65]: df_artist["artists"]=df_artist["artists"].str.replace("[", "")
      df_artist["artists"]=df_artist["artists"].str.replace("]", "")
      df_artist["artists"]=df_artist["artists"].str.replace("'", "")

      df_artist.head(100)
```

W_genres

```
df_w_genres['artists'] = df_w_genres['artists'].str.replace("'","")
df_w_genres['artists'] = df_w_genres['artists'].str.replace("[","")
df_w_genres['artists'] = df_w_genres['artists'].str.replace("]","")

df_w_genres.head()
```

```
df_w_genres['genres'] = df_w_genres['genres'].str.replace("'","")
df_w_genres['genres'] = df_w_genres['genres'].str.replace("[","")
df_w_genres['genres'] = df_w_genres['genres'].str.replace("]","")

df_w_genres.head()
```

In the csv file to remove the alphabetical sign to more accuracy to determine the duplicate value of data.

Most Popular Song in a Year:-

18 The Most Popular Song of Each Year

[75]:
```
#What were the most popular songs of each year
df_artist['artists'] = df_artist['popularity']
popular_songs = df_artist.groupby('artists')['popularity'].idxmax()
most_popular_songs = df_artist.loc[popular_songs, ['energy', 'tempo',␣
 ↪'popularity']]
most_popular_songs
```

[75]:

	energy	tempo	popularity
6	0.512000	134.819000	0.000000
9295	0.427806	99.140482	0.009709
8394	0.209431	105.889382	0.009804
20951	0.435190	111.985567	0.011407
13635	0.472767	106.635673	0.012579
…	…	…	…
7463	0.774000	112.050000	88.000000
11764	0.525000	97.054000	89.000000
15070	0.821000	99.999000	90.000000
14354	0.686000	103.013000	92.000000
20966	0.491000	91.066000	93.000000

[4663 rows x 3 columns]

Top 10:

27 The Top Ten Most Popular Songs

[113]:
```
#What are the most popular songs
top_songs = df_by_genres.sort_values('genres', ascending=False).head(10)
top_songs
```

[113]:

	mode	genres	acousticness	danceability	duration_ms	\
2972	1	zydeco	0.421038	0.629409	171671.690476	
2971	0	zurich indie	0.993000	0.705667	198417.333333	
2970	1	zouk	0.263261	0.748889	306072.777778	
2969	0	zouglou	0.161000	0.863000	206320.000000	
2968	1	zolo	0.222625	0.547082	258099.064530	
2967	0	zimdancehall	0.016600	0.766000	174893.000000	
2966	1	zhongguo feng	0.434976	0.574426	251337.727723	
2965	1	zeuhl	0.285011	0.359500	413526.500000	
2964	0	zambian pop	0.473000	0.689000	42813.000000	
2963	1	yugoslav rock	0.280400	0.560972	234421.722222	

	energy	instrumentalness	liveness	loudness	speechiness	\
2972	0.609369	0.019248	0.255877	-9.854825	0.050491	
2971	0.172667	0.468633	0.179667	-11.453333	0.348667	
2970	0.622444	0.257227	0.089678	-10.289222	0.038778	

2969	0.909000	0.000000	0.108000	-5.985000	0.081300
2968	0.610240	0.143872	0.204206	-11.295878	0.061088
2967	0.881000	0.000002	0.347000	-5.158000	0.137000
2966	0.527515	0.004562	0.137687	-6.746653	0.034451
2965	0.671500	0.019811	0.154200	-6.864500	0.042800
2964	0.775000	0.000000	0.219000	-11.938000	0.219000
2963	0.724750	0.001868	0.107108	-6.268444	0.056878

	tempo	valence	popularity	duration_min
2972	126.366087	0.808544	30.261905	2.86
2971	91.278000	0.739000	0.000000	3.31
2970	101.965222	0.824111	46.666667	5.10
2969	119.038000	0.845000	58.000000	3.44
2968	125.494919	0.596155	33.778943	4.30
2967	112.027000	0.674000	67.000000	2.91
2966	165.377817	0.556663	52.460396	4.19
2965	131.825000	0.309500	29.500000	6.89
2964	130.223000	0.646000	0.000000	0.71
2963	129.652444	0.588556	40.750000	3.91

[ ]:
```
# The Top Ten Most Popular Songs
```

[135]:
```
#What are the most popular songs
top_songs = df_year.sort_values('year', ascending=False).head(10)
top_songs
```

[135]:

	mode	year	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	key	duration_min
99	1	2020	0.219931	0.692904	193728.397537	0.631232	0.016376	0.178535	-6.595067	0.141384	124.283129	0.501048	64.301970	1	3.23
98	1	2019	0.278299	0.644814	201024.788096	0.593224	0.077640	0.172616	-7.722192	0.121043	120.235644	0.458818	65.256542	1	3.35
97	1	2018	0.267633	0.663500	206001.007133	0.602435	0.054217	0.176326	-7.168785	0.127176	121.922308	0.447921	63.296243	1	3.43
96	1	2017	0.286099	0.612217	211115.696787	0.590421	0.097091	0.191713	-8.312630	0.110536	117.202740	0.416476	63.263554	1	3.52
95	1	2016	0.284171	0.600202	221396.510295	0.592855	0.093984	0.181170	-8.061056	0.104313	118.652630	0.431532	59.647190	0	3.69
94	1	2015	0.253952	0.593774	230029.046606	0.627064	0.106787	0.188856	-7.625639	0.096779	120.115411	0.432098	56.700608	7	3.83
93	1	2014	0.249313	0.589948	233728.314713	0.648795	0.076570	0.191822	-7.067440	0.084061	122.305263	0.463049	55.543142	0	3.90
92	1	2013	0.257488	0.571148	242267.661437	0.645597	0.098365	0.199631	-7.472039	0.093849	120.806829	0.454741	54.047065	1	4.04
91	1	2012	0.249953	0.570882	245807.457584	0.656571	0.085206	0.189733	-7.260550	0.081742	121.781736	0.462709	52.655013	7	4.10
90	1	2011	0.273183	0.552867	236998.787308	0.648301	0.103772	0.203309	-7.574986	0.087479	121.483997	0.472454	53.307387	2	3.95

[ ]:
```
# The Top Ten Most Popular Songs
```

[167]:
```
#What are the most popular songs
top_songs = df_w_genres.sort_values('popularity', ascending=False).head(10)
top_songs
```

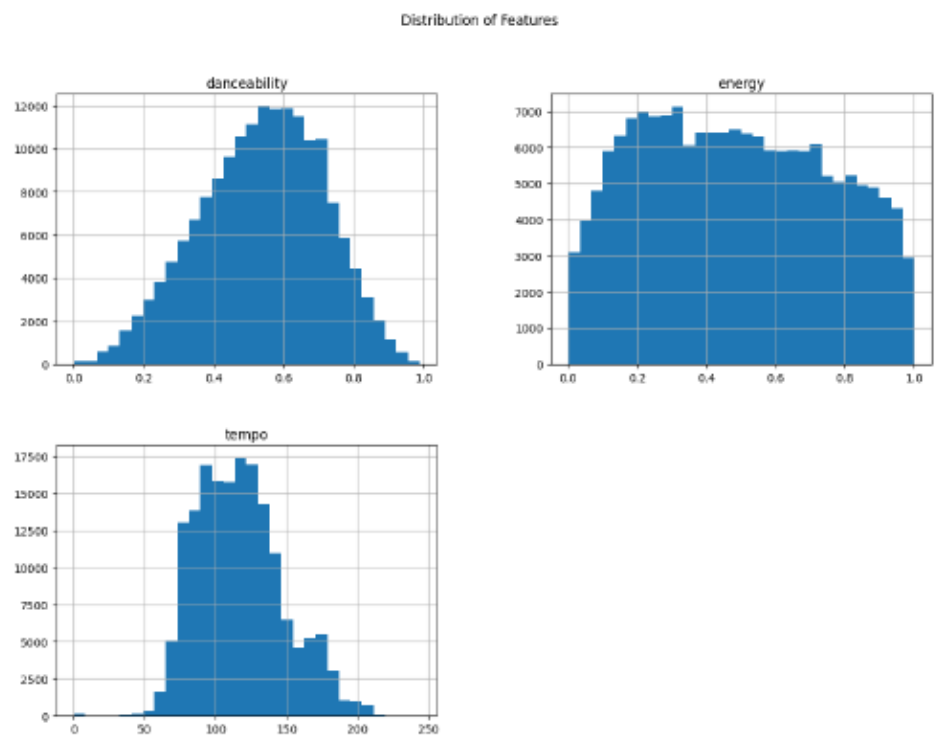
[167]:

	genres	artists	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	key	mode	count	duration_min
20966	bedroom pop	Ritt Momney	0.056300	0.399000	210463.0	0.491	0.000890	0.110000	-10.778000	0.0538	91.066000	0.151000	93.0	6	0	2	3.51
14354	latin pop, viral pop	Lele Pons	0.090700	0.905000	155825.0	0.686	0.000000	0.266000	-3.152000	0.0664	103.013000	0.963000	92.0	0	1	1	2.60
15070		Los Legendarios	0.310000	0.823000	213314.0	0.821	0.000004	0.143000	-3.402000	0.1660	99.999000	0.791000	90.0				
11764	cubaton, latin, pop venezolano, reggaeton, tra…	Jerry Di	0.819000	0.854000	197587.0	0.525	0.000000	0.146000	-4.426000	0.2140	97.054000	0.630000	89.0				
28263	modern indie pop	salem ilese	0.424000	0.738000	136839.0	0.621	0.000007	0.692000	-7.313000	0.0486	113.968000	0.715000	88.0				
23687	tropical house	Surf Mesa	0.068600	0.674000	176547.0	0.774	0.001880	0.393000	-7.567000	0.0892	112.050000	0.330000	88.0				
7463	social media pop	Emilee	0.068600	0.674000	176547.0	0.774	0.001880	0.393000	-7.567000	0.0892	112.050000	0.330000	88.0				
213	scandipop	A7S	0.166633	0.742667	168293.0	0.726	0.000000	0.154667	-5.921333	0.1980	121.296333	0.554667	87.0				
26318		Towy	0.111000	0.863000	260545.0	0.669	0.000000	0.056900	-5.952000	0.2820	164.062000	0.947000	86.0				
16453	south african house	Master KG	0.018500	0.880000	342613.0	0.483	0.000009	0.060700	-7.012000	0.0504	124.009000	0.827000	86.0				

Histogram:

Histogram

```
]: df_data[['danceability', 'energy', 'tempo']].hist(bins=30, figsize=(14, 10))
   plt.suptitle('Distribution of Features')
   plt.show()
```



Conclusion

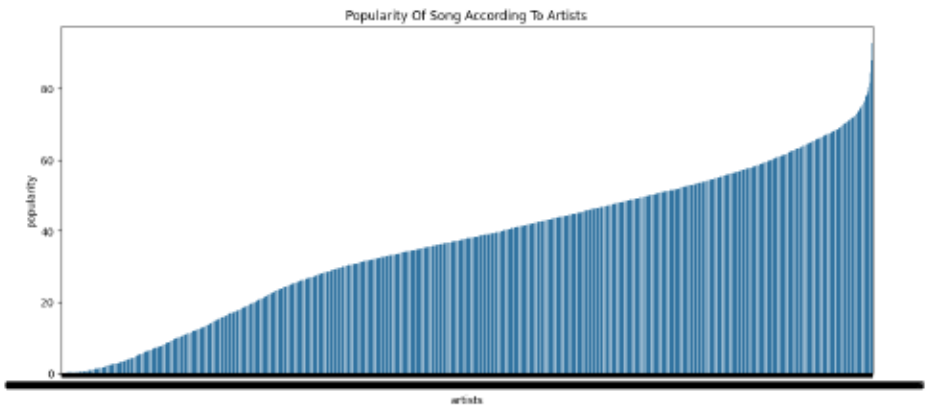
Each histogram reveals different characteristics of the music tracks in your dataset. Danceability and energy are confined to a 0–1 range, indicative of

their nature as proportions or percentages, while tempo, measured in BPM, varies more widely. Understanding these distributions can help in further analysis, like clustering songs into genres or optimizing recommendations based on user preferences.

Bar Plot:

```
: plt.figure(figsize=(14,6))
  sns.barplot(df_artist,x='artists',y='popularity')
  plt.title('Popularity Of Song According To Artists')
  plt.show()
```

23



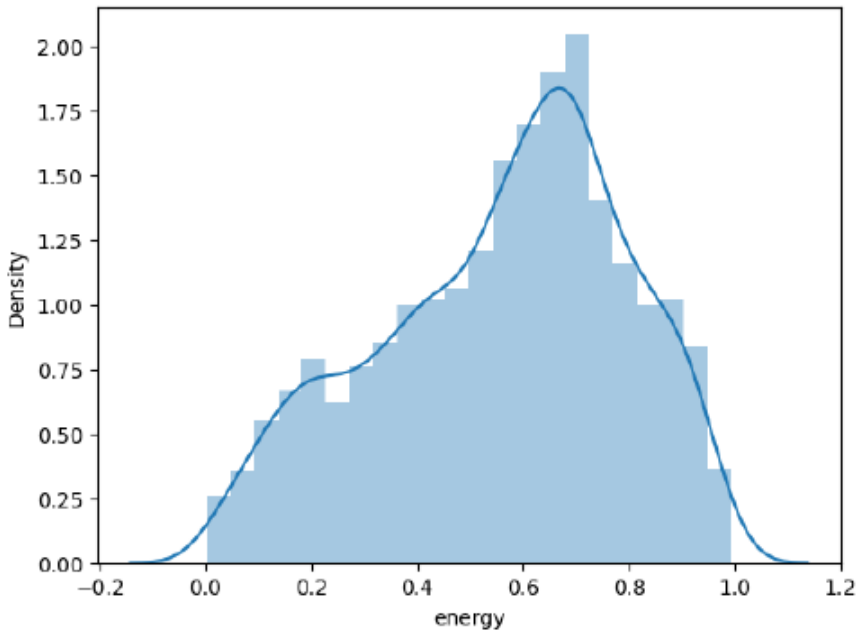
Conclusion:

The distribution shows a long-tail phenomenon, where a few artists have very high popularity, but most artists tend to cluster around lower or moderate popularity. This trend is typical in many industries, where a

small number of entities dominate in terms of public attention or success, while the majority are less known or successful.

Distplot:

```
[115]: sns.distplot(df_by_genres.energy)

[115]: <Axes: xlabel='energy', ylabel='Density'>
```



32

Conclusion:

- ***The distribution is relatively symmetric, suggesting that songs in the dataset typically have a moderate energy level.***
- ***This reflects a balance in the music collection: songs are neither overwhelmingly energetic nor very low energy. Instead, many songs fall into a mid-range of energy, making them potentially suitable for a variety of moods and settings.***

Final Thought:

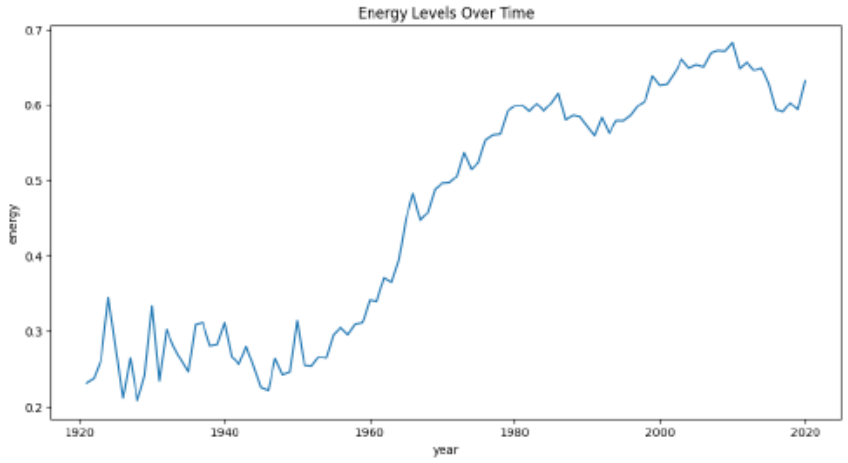
What can this distribution tell you about the types of songs in your dataset? Do you think this reflects the kind of music you'd expect (e.g., genres with balanced energy levels), or are you surprised by the lack of extreme energy values?

Line Plot:

```
[137]: plt.figure(figsize=(12, 6))
       sns.lineplot(data=df_year, x='year', y='energy')
       plt.title('Energy Levels Over Time')
```

38

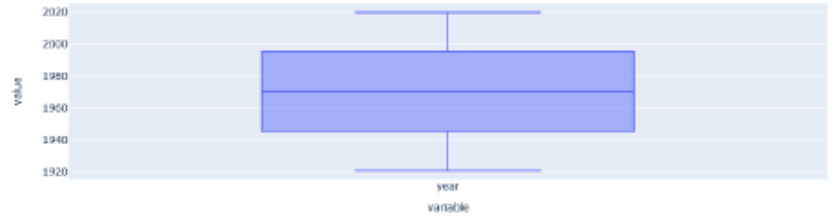
```
plt.show()
```



<u>It shows the how much increases and decreases the energy level of songs in the which year.</u>

<u>Box Plot:</u>

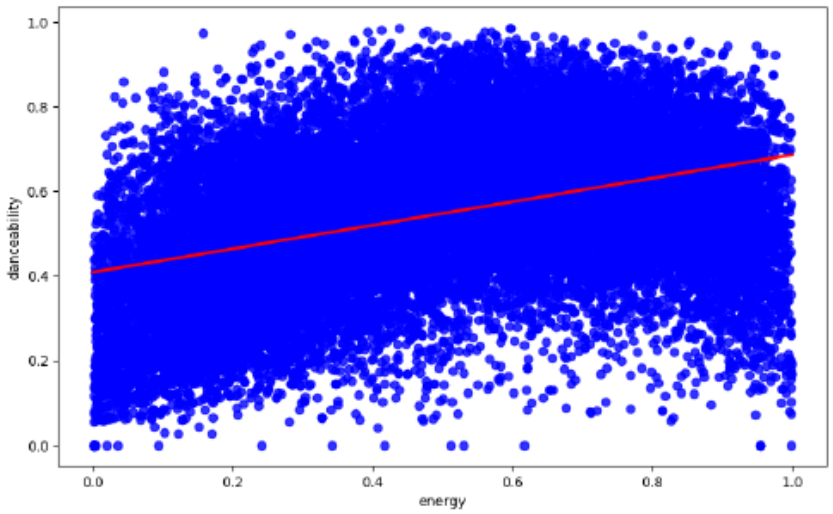
```
[139]: # Boxplot

       px.box(df_year.year)
```



<u>It shows the period of year according to the popularity of song.</u>

<u>Reg Plot:</u>

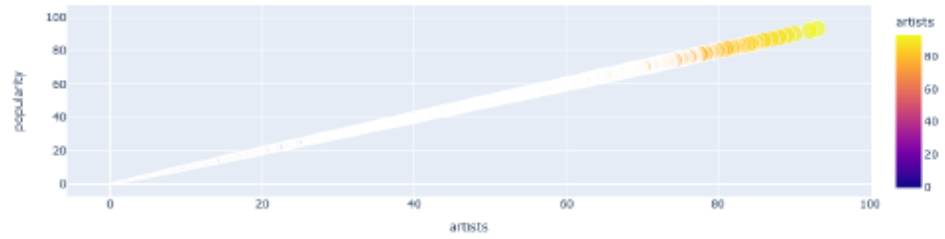
```
[169]: plt.figure(figsize = (10, 6))
       sns.regplot(data = df_w_genres, y ="danceability" , x = "energy", color = "b",␣
        ↪line_kws={"color":"red"})

[169]: <Axes: xlabel='energy', ylabel='danceability'>
```



<u>It shows the relation between danceability and energy as the genres of songs whose touch the red line in 0.4 axis in the graph.</u>

<u>Correlation Between Popularity and Artist:</u>

```
#Plot the corralation between popularity and energy
fig2 = px.scatter(pop_dance_corr, x="artists", y="popularity",␣
 ↪color="artists",size='popularity')
fig2.show()
```



It shows the relation between popularity and artist according to artist popular by their composed songs.

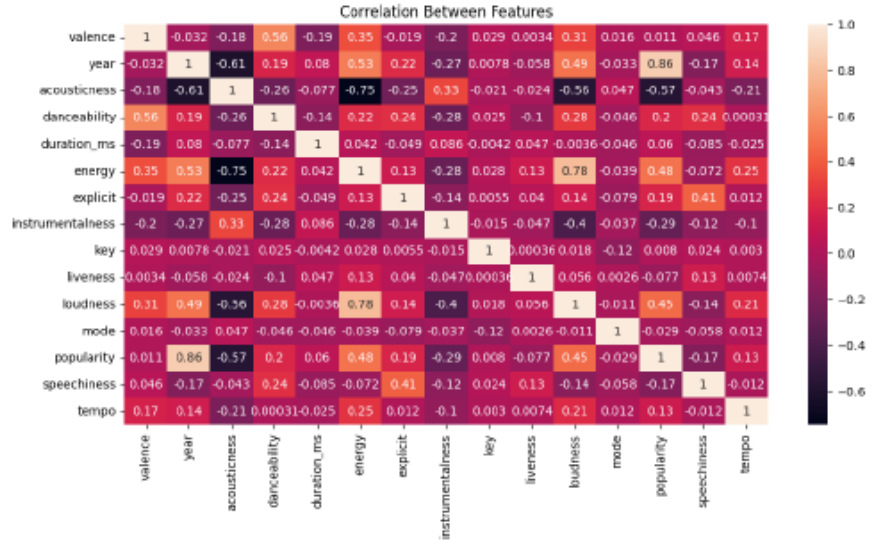
Heat Map:

12 Heatmap

```
[7]: # If we want to visualize the correltion we need to create a HeatMap...
plt.figure(figsize=(12, 6))
sns.heatmap(df_data.corr(numeric_only = True), annot=True,)
plt.title('Correlation Between Features')
```

11

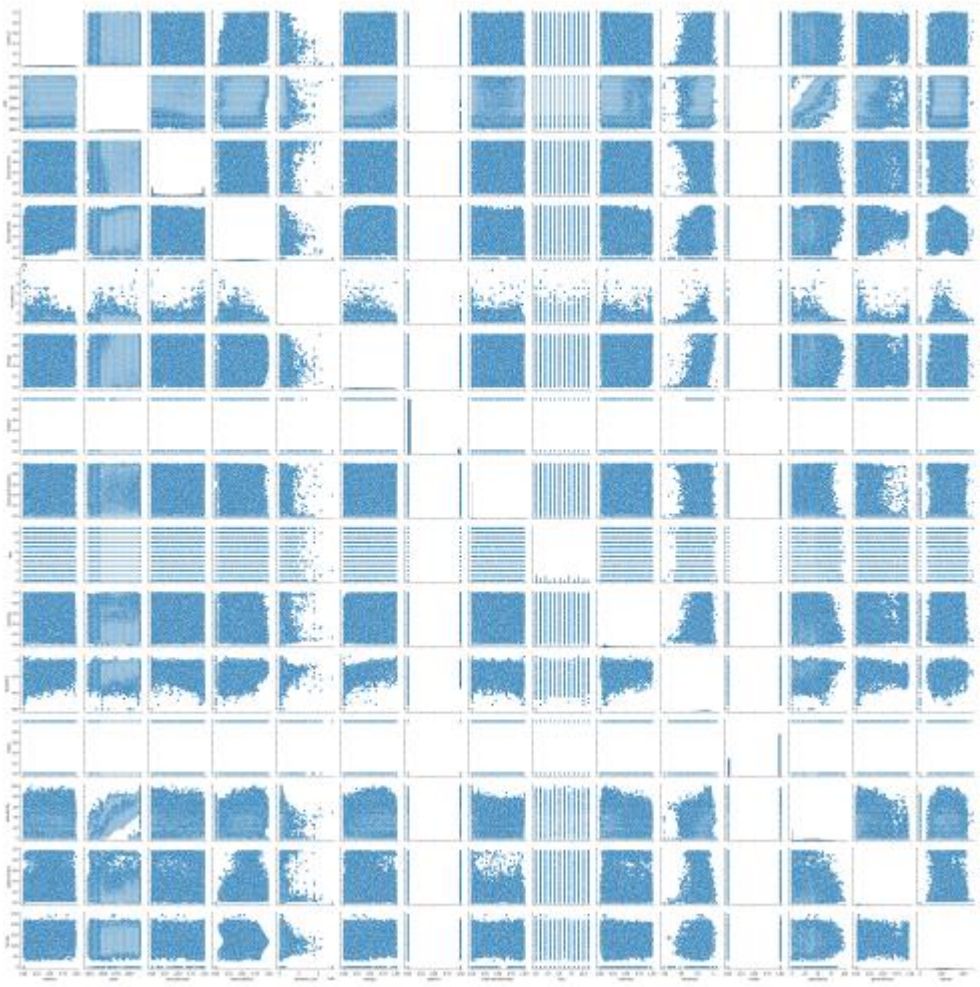
```
plt.show()
```



It shows the correlation between different columns of data and their values.

Pair Plot:

```
[39]: sns.pairplot(df_data)
      plt.show()
```



It shows all the diagram according to the column of data.

Final Report:

Conclusion Of Analysis:

The Spotify data analysis project demonstrates your expertise in data analysis, Python programming, and visualization techniques. By exploring the Spotify dataset, you uncover valuable insights into music trends, popularity, user behavior, and recommendations. The project showcases your ability to extract meaningful information from large datasets, apply statistical analysis and machine learning techniques, and present the results through interactive dashboards and visualizations.