Masana Nashimoto

Table of Contents

Abstract

This research undertakes a detailed exploration of song popularity on Spotify, utilizing a dataset extracted from the platform via the "Spotipy" Python library. The dataset, accumulated for research from 2000 to 2023, features approximately 1 million tracks, each characterized by 19 distinct attributes, and represents 61,445 unique artists across 82 genres.

The primary research inquiries focused on delineating the influential factors dictating song popularity on Spotify and assessing the feasibility of developing a predictive model capable of accurately forecasting song popularity. Exploratory data analysis, utilizing heat maps and matrix correlation, served as the initial step to discern patterns and associations within the musical features, laying down the groundwork for the analytical framework.

Subsequently, machine learning models—Random Forest Classifier and Random Forest Regression—were implemented to construct predictive models, facilitating the exploration of the relationship between track features and their popularity. The study uncovered crucial insights regarding the multifarious factors influencing a song's popularity and demonstrated the capability of the employed models to effectively predict popularity metrics, thereby, confirming the potential utility of machine learning/deep learning techniques in this domain.

Offering a granular understanding of music consumption patterns and preferences on Spotify is invaluable to artists, producers, and record labels. It also establishes a foundation for the utilization of sophisticated predictive models in the music industry, paving the way for future research aimed at further refining and enhancing the predictability of online music consumption.

Project Plan

Company Details:

Company: Spotify (Spotify Technology S.A.)
Founded: 2006 in Stockholm, Sweden
Founder: Daniel Ek, Martin Lorentzon
Headquarters: Stockholm, Sweden
Categories - Audio streaming, media services provider

Address:

Regeringsgatan 19 SE-111 53 Stockholm Sweden

Company Communication:

Phone Number: Not provided.
Website: https://open.spotify.com

Business Description:

Spotify Technology S.A. is a global audio streaming platform. Founded in Sweden in 2006, Spotify's core business revolves around offering users access to an extensive music library, personalized playlists, and a seamless streaming experience across various devices. With a robust free tier supported by advertising and a premium subscription model, the company has diversified its revenue streams. Spotify's strategic foray into podcasting, with exclusive content and acquisitions. Despite facing stiff competition, Spotify's commitment to personalization, user friendliness, and global reach has allowed it to maintain a strong presence and continue to innovate in an ever-evolving market.

Financials:

Ticker Symbol - SPOT
Latest Financial Data - 2022
Revenue - $12.355 Billion
No. of Employees - 5,584

Key Executives:

Daniel Ek - Founder, Chief Executive Officer & Chairman
Paul Vogel - Chief Financial Officer
Katarina Berg - Chief Human Resources Officer
Gustav Söderström - Co-President, Chief Product & Technology Officer
Dustee Jenkins - Chief Public Affairs Officer
Alex Norström - Co-President, Chief Business Officer
Martin Lorentzon - Co-Founder and Director

Major Competitors:

Apple Music
Amazon Music
YouTube Music
Deezer
Tidal
Google Play Music
SoundCloud

Business/Analysis Opportunity:

Utilizing the Spotify dataset includes about 1 million tracks with 19 features between 200 and 2023. The number of observations is sufficient to analyze the research questions and develop predictive models. These models will reveal any questions or trends that can be found in the charts.

Research Questions:

1: What makes a song popular on an online platform especially Spotify?

Purpose: The purpose of this research question is to investigate the factors that contribute to a song's popularity on the online platform Spotify. Understanding the elements that drive a song's success on Spotify can provide valuable insights for artists, record labels, and the music industry. By identifying these factors, we can potentially improve the chances of a song gaining traction and becoming a hit. This research can help in making data-driven decisions regarding song production, promotion, and distribution.

Statistical Technique/Model Approach: To address the research question of what makes a song popular on an online platform like Spotify, a data-driven approach will be employed using statistical techniques. The heatmap will offer a visually intuitive representation of how various song metrics relate to their respective popularity scores, highlighting any discernible patterns or trends in the dataset. This initial visual exploration will set the stage for a more quantitative evaluation using the correlation matrix. This matrix is designed to measure the strength and direction of linear relationships between song attributes and popularity scores. By identifying which attributes show strong positive or negative correlations with popularity, we can pinpoint potential key factors that influence a song's success on Spotify.

Brief Methodology: The methodology for investigating what leads to a song's popularity on Spotify involves data collection from, metadata, audio features, and popularity metrics. Feature engineering will transform this data into a feature matrix, with each row representing a song and columns containing relevant features. Exploratory data analysis will reveal initial patterns and relationships between features and popularity. Feature importance analysis will elucidate the critical factors influencing song popularity. Following this, a correlation matrix will be constructed to quantitatively assess the relationships between the various song attributes and their popularity. This matrix will highlight which attributes most closely align with song success, ultimately providing insights and recommendations for artists and the music industry.

2. Can I build a predictive model that accurately forecasts the popularity of a song on Spotify?

Purpose: The purpose of this research question is to explore the feasibility of building a predictive model that accurately forecasts the popularity of songs on the Spotify platform.

Accurately predicting song popularity can be valuable for artists, record labels, and the music industry, as it can inform decision-making related to production, promotion, and marketing strategies. By identifying the factors and features that contribute most significantly to a song's popularity, this research aims to develop a data-driven model that enhances the understanding of what resonates with Spotify's audience.

Statistical Technique/Model Approach: To address the research question, a predictive modeling approach will be employed, leveraging machine learning techniques. Regression models, such as linear regression or random forests will be considered to forecast song popularity, typically represented as the number of streams or other engagement metrics. Model evaluation will employ metrics like root mean square error to assess predictive accuracy.

Brief Methodology: I can use regression, like linear regression or other regression methods, to build a predictive model that accurately forecasts the popularity of songs on Spotify. To do this, I plan to gather a dataset containing historical song attributes and popularity metrics, preprocess the data, select relevant features, and split it into training, validation, and test sets. The next step will be to train the regression model on the training data, optimize hyperparameters, and assess its performance using evaluation metrics such as root mean squared error. Regression will allow me to estimate continuous popularity metrics, making it a effective methodology for this prediction task.

Hypothesis:

1: Musical Characteristics Impact Song Popularity Hypothesis

Songs with specific musical characteristics, such as higher danceability, tempo, or energy levels, are more likely to be popular on Spotify. Rationale: Users may be drawn to songs that are more energetic or suitable for dancing, leading to increased popularity. Analyzing these musical features can provide insights into what types of songs tend to perform well on the platform.

2: Artist Popularity Correlates with Song Success Hypothesis

Songs by popular artists, measured by factors like follower count or previous hit songs, are more likely to become popular on Spotify. Rationale: Established artists often have a built-in fan base, which can lead to higher initial listenership for their new releases. Investigating the relationship between artist popularity and song success can shed light on the importance of artist reputation in driving popularity.

3: Time of Release Influences Song Popularity Hypothesis

The timing of a song's release, such as the day of the week or time of day, affects its potential for popularity on Spotify. Rationale: Users' music listening habits may vary depending on the day of the week or time of day. Identifying patterns in song release timing and subsequent popularity can help artists and record labels optimize their release strategies.

About Data:

The Spotify song charts extracted from kaggle.

" This is a complete dataset of all the "Top 200" and "Viral 50" charts published globally by Spotify. Spotify publishes a new chart every 2-3 days. This is its entire collection since January 1, 2017."

(https://www.kaggle.com/datasets/dhruvildave/spotify-charts)

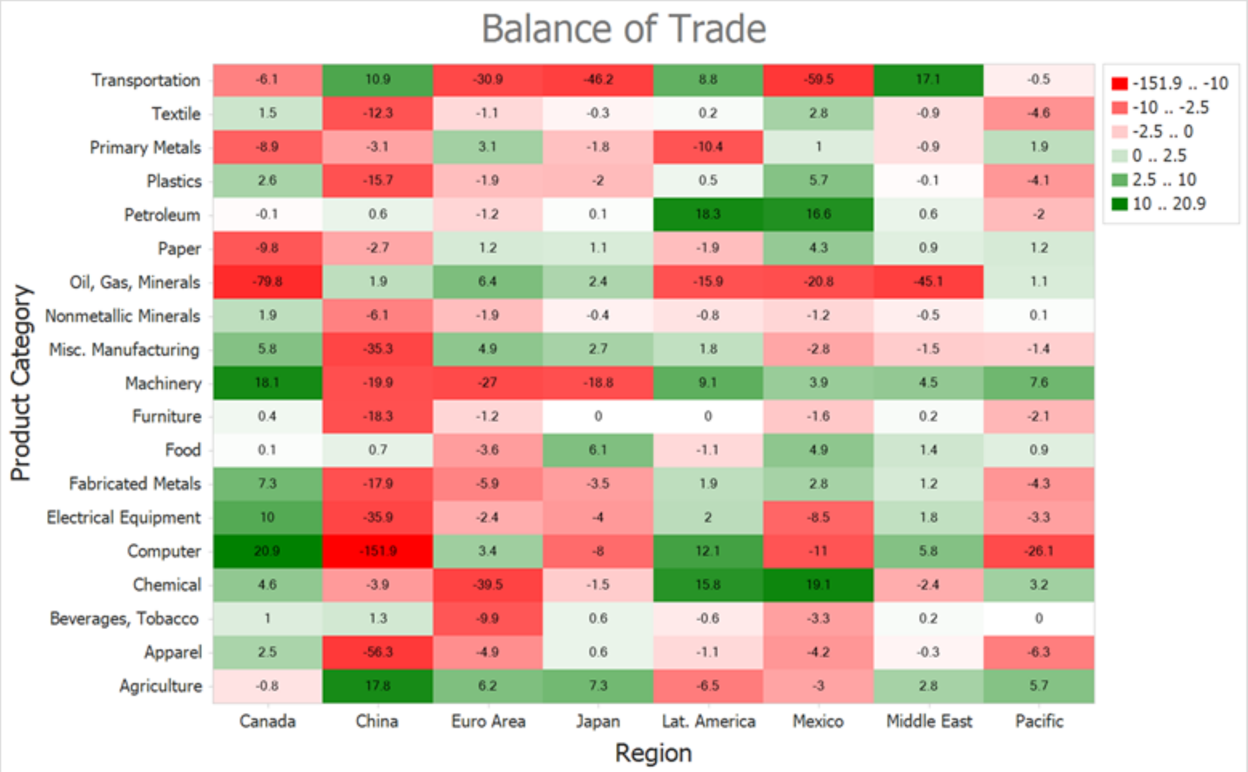
Computational Methods and Outputs:

Exploratory data analysis (EDA) will reveal what is necessary for research question number one. EDA helps me understand the distribution of key variables related to song popularity, such as the number of streams, chart rankings, and user engagement metrics. By examining histograms, density plots, and summary statistics, I can gain insights into the central tendency and variability of these metrics.

Question two builds on top of question one. Regression is a highly useful and appropriate approach for building a predictive model that accurately forecasts the popularity of a song on Spotify. Predicting song popularity, typically measured by metrics like the number of streams, chart rankings, or user engagement, involves estimating a continuous numeric target variable. Regression is specifically designed for such prediction tasks where the outcome is a continuous value.

Output Summaries:

Heatmap, like the ones below, will be implemented after the result of EDA for RQ1. Then I will be able to create the predictive model using the RQ1 EDA. The output of a predictive model for forecasting the popularity of a song on Spotify typically consists of numerical predictions or scores that estimate the song's expected level of popularity. These predictions can represent metrics like the number of streams, chart ranking, or user engagement. The format may vary, ranging from a single numeric value to probability scores, ranking scores, or time-series forecasts, depending on the specific prediction model design.



Product Category	Canada	China	Euro Area	Japan	Lat. America	Mexico	Middle East	Pacific
Transportation	-6.1	10.9	-30.9	-46.2	8.8	-59.5	17.1	-0.5
Textile	1.5	-12.3	-1.1	-0.3	0.2	2.8	-0.9	-4.6
Primary Metals	-8.9	-3.1	3.1	-1.8	-10.4	1	-0.9	1.9
Plastics	2.6	-15.7	-1.9	-2	0.5	5.7	-0.1	-4.1
Petroleum	-0.1	0.6	-1.2	0.1	18.3	16.6	0.6	-2
Paper	-9.8	-2.7	1.2	1.1	-1.9	4.3	0.9	1.2
Oil, Gas, Minerals	-79.8	1.9	6.4	2.4	-15.9	-20.8	-45.1	1.1
Nonmetallic Minerals	1.9	-6.1	-1.9	-0.4	-0.8	-1.2	-0.5	0.1
Misc. Manufacturing	5.8	-35.3	4.9	2.7	1.8	-2.8	-1.5	-1.4
Machinery	18.1	-19.9	-27	-18.8	9.1	3.9	4.5	7.6
Furniture	0.4	-18.3	-1.2	0	0	-1.6	0.2	-2.1
Food	0.1	0.7	-3.6	6.1	-1.1	4.9	1.4	0.9
Fabricated Metals	7.3	-17.9	-5.9	-3.5	1.9	2.8	1.2	-4.3
Electrical Equipment	10	-35.9	-2.4	-4	2	-8.5	1.8	-3.3
Computer	20.9	-151.9	3.4	-8	12.1	-11	5.8	-26.1
Chemical	4.6	-3.9	-39.5	-1.5	15.8	19.1	-2.4	3.2
Beverages, Tobacco	1	1.3	-9.9	0.6	-0.6	-3.3	0.2	0
Apparel	2.5	-56.3	-4.9	0.6	-1.1	-4.2	-0.3	-6.3
Agriculture	-0.8	17.8	6.2	7.3	-6.5	-3	2.8	5.7

Campaign Implementation:
RQ1: What makes a song popular on an online platform, especially Spotify?

By investigating this question, I can uncover the factors that contribute to a song's popularity, such as artist attributes, genre, release timing, and audio features. This knowledge empowers record labels, artists, and music promoters to make informed decisions about song production, marketing strategies, and playlist placements. Understanding the driving factors of song popularity allows businesses to optimize their content and promotional efforts, resulting in more successful music campaigns and improved audience engagement.

RQ2: Can I build a predictive model that accurately forecasts the popularity of a song on Spotify?

Developing a predictive model for song popularity offers businesses a powerful tool for decision-making. Record labels can identify potential hits and allocate resources effectively, while music marketers can refine their advertising and playlist placement strategies. Artists can tailor their music to align with audience preferences, increasing the likelihood of commercial success. Accurate predictions help businesses make data-driven choices, ensuring their investments are focused on songs with the greatest potential for success.

Literature Review

In the digital age of music consumption, the rise of online streaming platforms has revolutionized the way we discover and engage with music. Among these platforms, Spotify stands as a dominant force with millions of tracks and a user base spanning the globe. In this era of abundant musical choices, the factors that determine a song's popularity on Spotify have become a subject of great intrigue and investigation. This literature review attempts to uncover the underlying mechanisms that contribute to the success of songs on this influential platform. It seeks to address three pivotal questions: What are the key attributes that render a song popular on an online platform, especially Spotify? Can the combination of data science and predictive modeling provide accurate forecasts of a song's success on Spotify? Moreover, can the insights gleaned from such analyses be harnessed to craft an effective recommendation system? As I delve into these questions, I aim to not only illuminate the intricacies of the digital music landscape but also explore the potential for data-driven approaches to enhance our music listening experiences.

Rutger Nijkamp's research article provides valuable insights that directly relate to my literature review questions. Firstly, the study aligns with the inquiry into "What makes a song popular on an online platform, especially Spotify?" Nijkamp's investigation explores the determinants of song popularity on Spotify, concentrating on audio features sourced from the platform's database, such as key and tempo. By analyzing the number of streams, the research aims to uncover the underlying factors contributing to a song's popularity, thus shedding light on the dynamics of why certain songs gain more traction than others in the digital landscape. Secondly, the research resonates with the question of whether it is feasible to "build a predictive model that accurately forecasts the popularity of a song on Spotify." Beyond investigating the factors influencing popularity, Nijkamp's study delves into the creation of a predictive model. Utilizing regression analysis, the research endeavors to construct a model capable of forecasting a song's popularity based on its audio attributes. This aspect of the research directly addresses the feasibility and complexities associated with developing predictive models for song popularity, offering insights into the potential and challenges of such modeling in the context of online music platforms.

Lastly, while Nijkamp's abstract primarily focuses on predicting song popularity, it indirectly opens avenues for considering the creation of a recommendation system. Although not explicitly mentioned, the dataset and methodology used in the research can potentially be extended to build a recommendation system. After establishing a predictive model for song popularity, it becomes possible to adapt this model for recommending songs to users on Spotify, aligning with the broader question of whether a recommendation system can be developed using the given dataset.

Incorporating Rutger Nijkamp's research into my literature review allows me to demonstrate how the study addresses critical aspects of my research questions. It illustrates the research's contribution to my understanding of song popularity on Spotify, the potential for predictive modeling in this context, and the groundwork laid for potentially building a recommendation system using similar data. This integration will help position my own research within the context of existing studies, highlighting the significance of data science and predictive modeling in the context of online music platforms like Spotify.

Exploratory Data Analysis

For this project, I collected a Spotify dataset that was extracted from the Spotify platform. The dataset was obtained from Kaggle. The dataset is described on Kaggle to be, "This dataset was extracted from the Spotify platform using the Python library "Spotipy", which allows users to access music data provided via APIs. The dataset collected includes about 1 Million tracks with 19 features between 2000 and 2023. Also, there is a total of 61,445 unique artists and 82 genres in the data. This clean data has been prepared and utilized for research purposes. Its significance lies in its potential to unravel patterns and predict song popularity prior to its release. This dataset could be used to create various predictive models with machine-learning/deep-learning techniques."

The original data consist of 1,159,764 observations with 20 attributes. Out of all 20 attributes, I reduced a few of them to make the dataset look cleaner. Below are attributes I decided to keep:

Popularity: Track popularity (0 to 100)

Year: Year released (2000 to 2023)

Danceability: Track suitability for dancing (0.0 to 1.0)

Energy: The perceptual measure of intensity and activity (0.0 to 1.0)

Key: The key, the track is in (-1 to -11)

Loudness: Overall loudness of track in decibels (-60 to 0 dB)

Acousticness: Confidence measure from 0 to 1 of whether the track is acoustic

Instrumentalness: Whether tracks contain vocals (0.0 to 1.0)

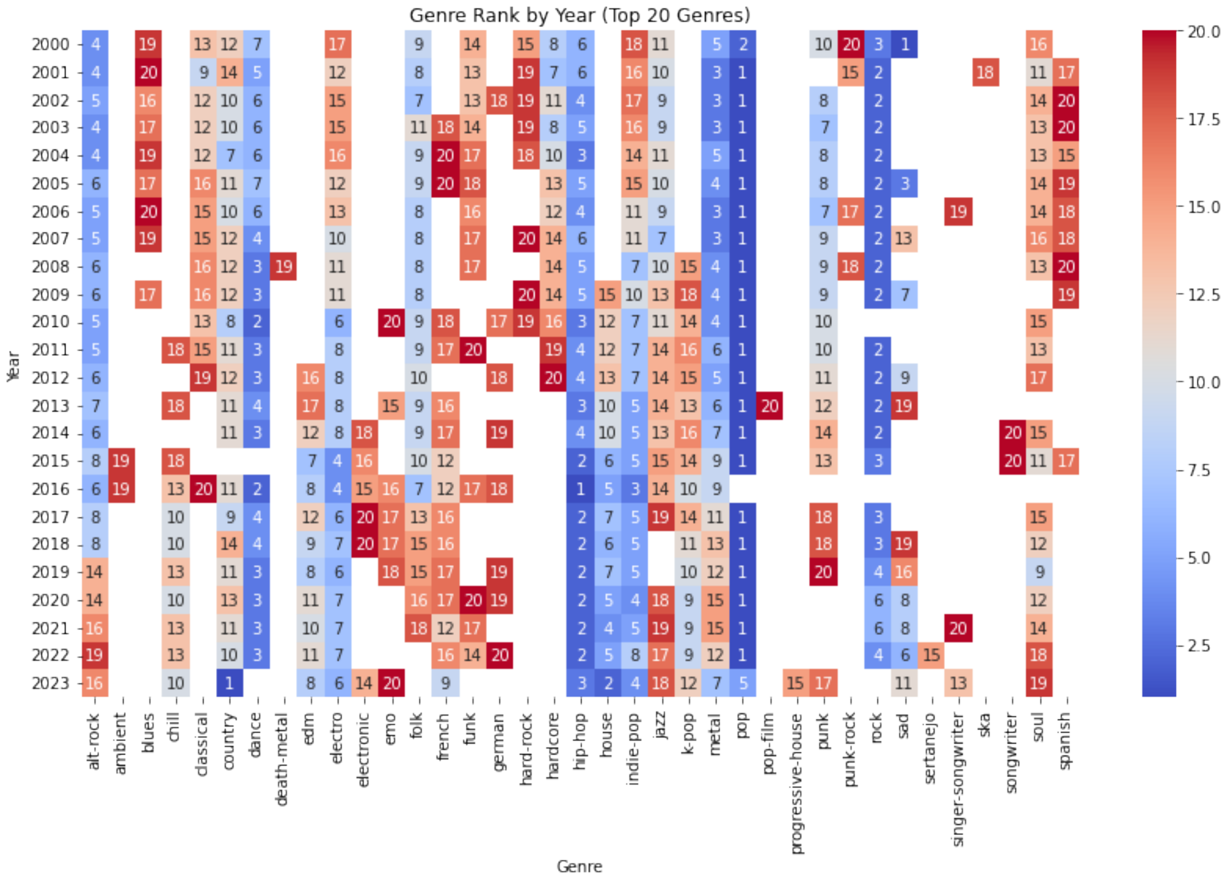
Liveness: Presence of audience in the recording (0.0 - 1.0)

Valence: Musical positiveness (0.0 to 1.0)

Tempo: Tempo of the track in beats per minute (BPM)

Duration_ms: Duration of track in milliseconds

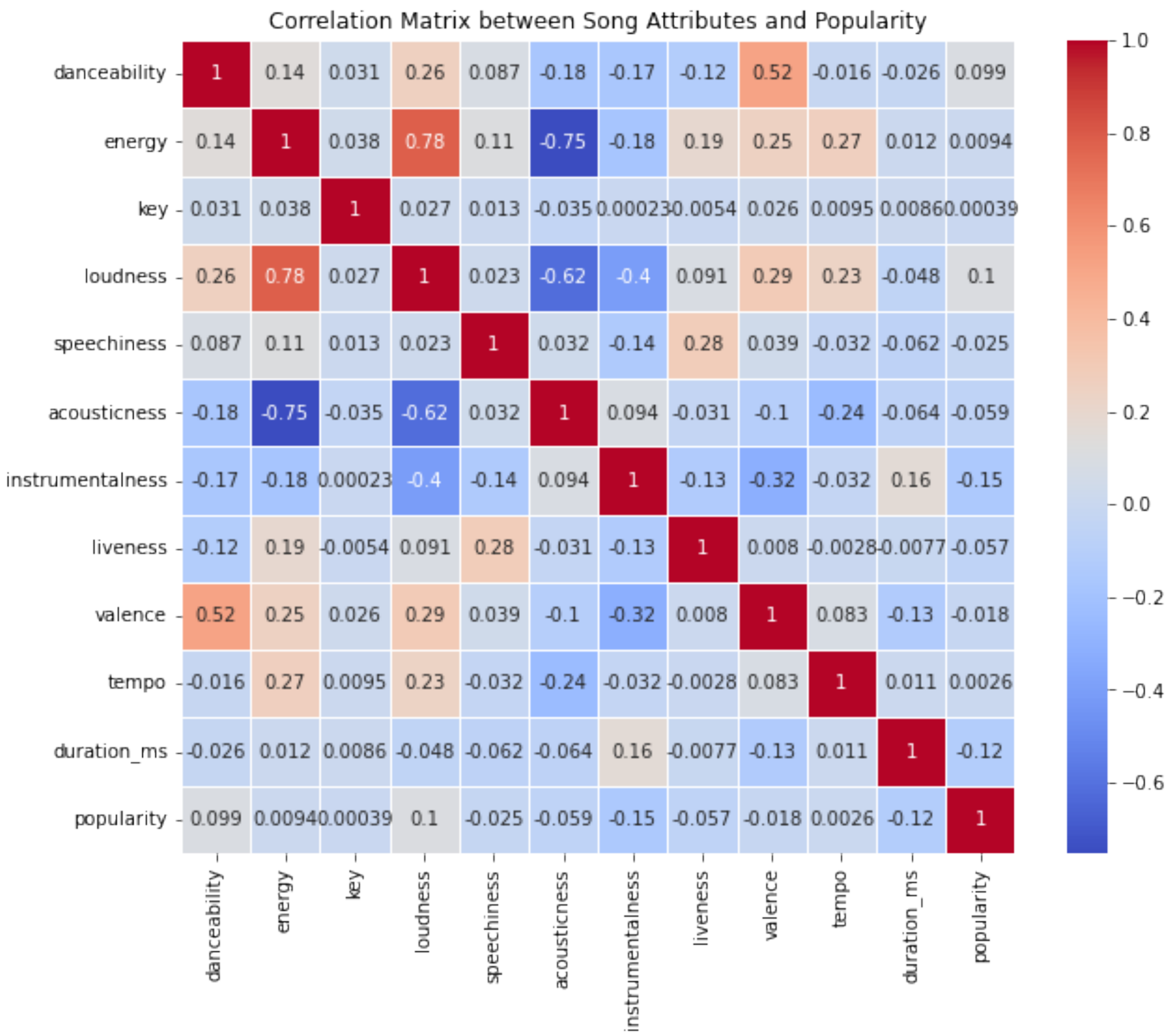
The RQ1 is: **What makes a song popular on an online platform, especially Spotify?**



The creation of a heatmap showcasing the top 20 genres by year is very helpful in addressing my overarching research question of what makes a song popular on Spotify. This visual representation reveals the intriguing dynamics of music genres on the platform over a five-year period, offering a glimpse into the ebb and flow of listener preferences. The heatmap illuminates dance music's impressive consistency, maintaining a position in the top 3 genres since 2008. This showcases the enduring appeal of dance music on Spotify, indicating its role as a genre that continually resonates with listeners. Another interesting finding is the surge of country music, which claimed the number 1 spot in 2023, despite not even qualifying within the top 10 genres in the previous four years. This sudden shift highlights the potential for genre surprises and suggests that country music has made a significant impact within a short timeframe. Hip-hop on the other hand has maintained its presence within the top 3 genres for the past five years, this emphasized its lasting popularity among Spotify users. This consistency positions hip-hop as a genre with a robust and dedicated audience.

Genres like house and indie-pop have demonstrated reliability, consistently hovering around the top 5 positions over the past five years. Their popularity suggests a consistent appeal that withstands yearly fluctuations. Pop music's dominance is very evident on the heatmap, having claimed the top spot for four consecutive years. However, its drop from the top 5 in the current year raises questions about whether listener preferences are evolving, signaling a potential shift in the consumption of music. The heatmap also highlights the decline of rock music, which had consistently ranked in the top 4 or 6 genres for the past four years but no longer holds a spot in the top 20 this year. The multiple shifts in popularity shown on the heatmap indicate that the music tastes of Spotify users are constantly in flux and also may show a possible shift away from traditionally popular music of the past like rock.

Next, I prepared the correlation matrix between song attributes and popularity as shown below.


Correlation Matrix between Song Attributes and Popularity

The correlation values between the 'Popularity' score and various song attributes suggest the strength and direction of the linear relationships between these attributes and song popularity. Here's what I can infer from the correlation values(Please see appendix for the scatter plot of each attributes vs popularity):

Danceability (0.099): The positive correlation (0.099) between danceability and popularity indicates a weak positive relationship. This suggests that songs with higher danceability tend to be slightly more popular on average, but the relationship is not very strong.

Energy (0.0094): The correlation between energy and popularity is very close to zero (0.0094), indicating that there is almost no linear relationship between energy and song popularity. Energy levels in songs do not seem to strongly influence their popularity.

Key (0.0039): The correlation between the key of the song and popularity are also close to zero (0.0039), suggesting that the key in which a song is composed has little to no impact on its popularity.

Loudness (0.1): The positive correlation (0.1) between loudness and popularity indicates a weak positive relationship. Louder songs tend to be slightly more popular on average, but the relationship is not very strong.

Speechiness (-0.025): The negative correlation (-0.025) between speechiness and popularity suggests a weak negative relationship. This means that songs with more speech-like elements (e.g., spoken word or rap) may be slightly less popular on average, but the effect is not substantial.

Acousticness (-0.059): The negative correlation (-0.059) between acousticness and popularity implies a weak negative relationship. Songs with higher acousticness (more acoustic instruments) tend to be slightly less popular on average, but the relationship is not strong.

Instrumentalness (-0.15): The negative correlation (-0.15) between instrumentalness and popularity indicates a moderate negative relationship. This suggests that songs with higher instrumentalness (i.e., fewer vocals) are less popular on average. The relationship is relatively stronger compared to other attributes.

Liveness (-0.057): The negative correlation (-0.057) between liveness and popularity suggests a weak negative relationship. Songs recorded with a live audience presence tend to be slightly less popular on average, but the effect is not very strong.

Valence (-0.018): The negative correlation (-0.018) between valence (musical positiveness) and popularity implies a weak negative relationship. Songs with higher valence (more positive-sounding) may be slightly less popular on average, but the relationship is not significant.

Tempo (0.0026): The correlation between tempo and popularity is very close to zero (0.0026), indicating that there is almost no linear relationship between the tempo of a song and its popularity.

Duration_ms (-0.12): The negative correlation (-0.12) between song duration and popularity suggests a moderate negative relationship. Shorter songs tend to be slightly more popular on average, and this relationship is relatively stronger compared to some other attributes.

The correlation matrix analysis of song attributes and their associations with popularity on Spotify provides valuable insights into the factors influencing a song's appeal to listeners. Notably, danceability and loudness exhibit weak positive correlations with popularity, suggesting that songs that are more danceable and louder tend to be slightly more popular, although these relationships are not particularly strong. Conversely, instrumentalness demonstrates a moderate negative correlation, indicating that songs with fewer vocals are generally less popular. Additionally, shorter songs tend to enjoy greater popularity. However, acousticness and speechiness both show weak negative correlations, suggesting that songs with higher acoustic elements or more speech-like features may be slightly less popular, though the effect is not substantial. Several attributes, including energy, key, valence, tempo, and liveness, have correlations close to zero, implying that they have minimal linear relationships with song popularity. It's important to note that these correlations provide a foundational understanding, but the multifaceted nature of song popularity encompasses a broader spectrum of influences, including genre, artist, marketing, and user preferences.

Methodology

RQ2: Can I build a predictive model that accurately forecasts the popularity of a song on Spotify?

Regression models are particularly useful when I want to understand the relationship between a song's popularity and one or more song attributes. Linear regression, for example, can help me uncover linear relationships between these attributes and song popularity. It can provide insights into how each attribute contributes to a song's popularity and whether it has a positive or negative impact.

Random forests, on the other hand, are an ensemble learning method that can handle non-linear relationships and interactions between attributes. They can capture complex patterns in the data and provide more accurate predictions. By using random forests, I can go beyond the linear assumptions of linear regression and create a more robust model for forecasting song popularity. My dataset, which includes information on over 1 million tracks and 14 features (not including artist and track name) spanning from 2000 to 2023, provides a rich source of data for building and training these predictive models. The features I've selected, such as danceability, energy, loudness, and others, are relevant indicators of a song's potential popularity, making them valuable inputs for my models.

In my EDA, including the creation of heatmaps and correlation matrices, has already helped me gain insights into how these attributes relate to song popularity. This preliminary analysis allows me to identify which attributes might be strong predictors of popularity and which ones might have a weaker influence.

Data Visualization

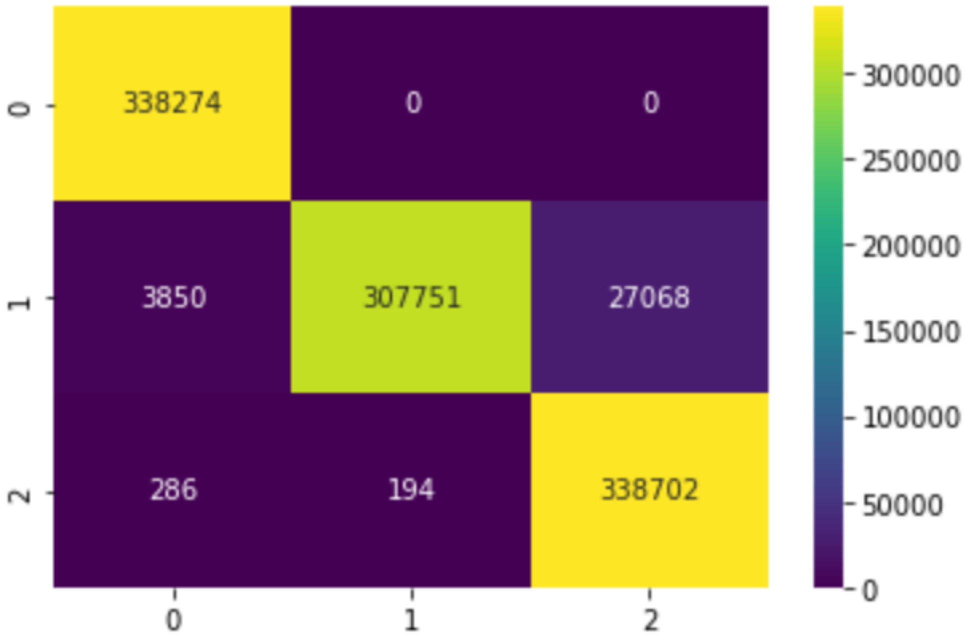
Random Forest Classifier:

Confusion Matrix:

338274	0	0
338	325542	12789
234	192	338756

Accuracy: 98.66620740558494

	Precision	Recall	Fl-Score	support
High	1.00	1.00	1.00	338274
Low	1.00	0.96	0.98	338669
Moderate	0.96	1.00	0.98	339182
Accuracy			0.99	1016125
Macro Avg	0.99	0.99	0.99	1016125
Weighted Avg	0.99	0.99	0.99	1016125



In developing a predictive model that accurately gauges the popularity of songs on Spotify, I chose to employ a random forest classifier. The random forest method offers the potential for enhanced accuracy and stability in predictions, and it is also known for its ability to combine multiple decision trees. In reflecting upon the results of this classifier, its precision and accuracy are evident.

Regarding the confusion matrix, my model's predictions, when set against the true classifications, reveal a level of accuracy that exceeded my expectations. Of the 338,274 songs that are categorized as "High" in popularity, my model was very accurate and did not misclassify a single one. Analyzing the "Low" popularity bracket, out of 338,669 songs, 325,542 were pinpointed accurately. While 12,789 were mislabeled as "Moderate" and 338 as "High", this minor margin of error showcases the model's overall reliability. For songs falling under the "Moderate" popularity classification, the model continued its trend of accuracy, correctly identifying 338,756 out of 339,182.

A deeper analysis into the overall accuracy metric positions it at an impressive 98.67%. This is more than just a number but also a reflection of the model's potential utility for industry stakeholders.

Examining the classification report further underscores the model's capabilities. For "High" popularity songs, both precision and recall stood at a perfect 1.00, emphasizing the model's confidence and reliability in identifying chart-toppers. The "Low" popularity category saw an equally commendable precision of 1.00, with a slightly diminished recall of 0.96. This suggests that while the model is adept at correctly classifying songs of low popularity, there's a slight tendency to misclassify a few as "Moderate". In the "Moderate" category, the precision of 0.96 combined with a recall of 1.00 points towards the model's excellence in identifying moderate hits, with a slight propensity to label a few songs from other categories as "Moderate". Reflecting on my initial research question regarding the feasibility of crafting an effective predictive model for song popularity on Spotify, the results from my random forest classifier offer a strong affirmative. With its understanding of the varied popularity brackets and high level of accuracy, I believe the current model set forth will be an effective tool in the realm of music analytics.

Random Forest Regression:

Mean Squared Error	76.63935130884003
Mean Absolute Error	6.48
Root Mean Squared Error	8.75
R-Squared Score	0.7

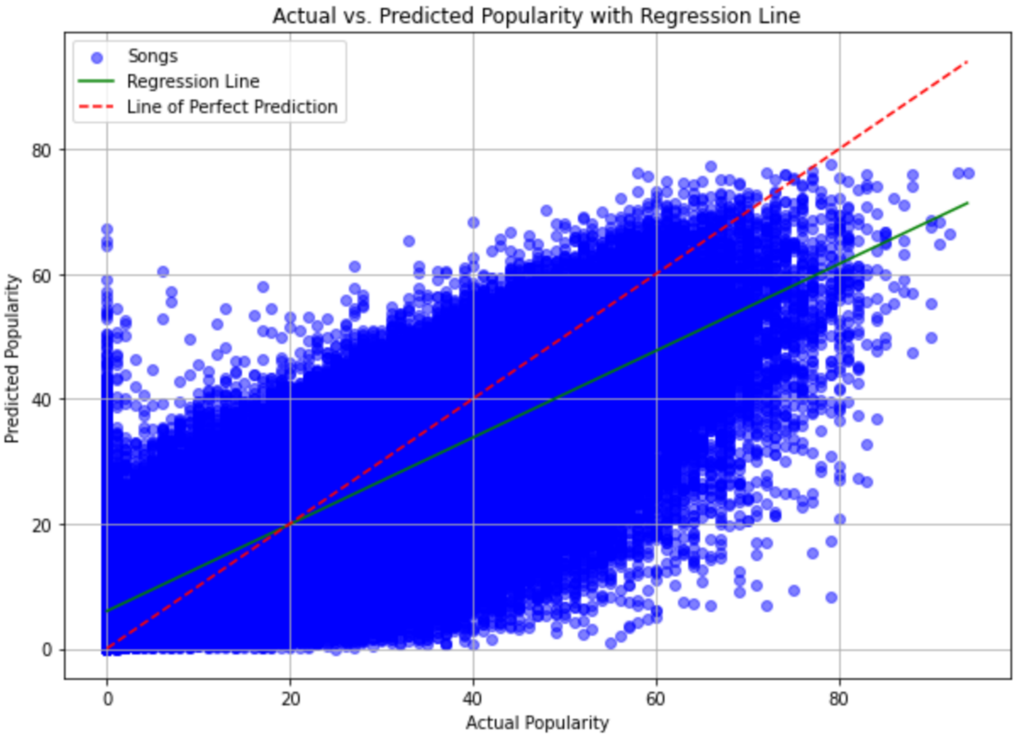
I also utilized the Random Forect Regression technique in order to understand and predict song popularity on Spotify. The outcomes were helpful and offered a fresh perspective on the intricacies of musical tastes and preferences.

At the heart of this analysis lies the Mean Squared Error (MSE), which stood at 76.64. This metric reflects the average squared differences between the anticipated and actual values of song popularity. While the number might initially seem abstract, it underscores the accuracy of the model in its predictive capacity. Naturally, the aspiration is to see this number dwindle, indicating heightened precision. However, as an initial attempt at musical analytics, this provides a constructive starting point.

A more tangible metric, the Mean Absolute Error (MAE), came in at 6.48. This illustrates that the model's predictions, on average, sway by about 6.48 units from the true popularity scores. Given the broader spectrum of Spotify's popularity measure, which spans from 0 to 100, this deviation, although not insignificant, is certainly within an acceptable margin. The Root Mean Squared Error (RMSE), another important metric, was measured at 8.75. It delineates the average magnitude by which our predictions falter from the real values. This suggests that the predictions are typically within an approximate range of ±8.75 units from the actual popularity score.

Another helpful statistic is the R-squared (R2) Score. The R2 score measured at 0.70, it indicates that a substantial 70% of the variability intrinsic to a song's popularity is captured and explained by the model. This number furhter helps to show the model's capability because it successfully encapsulates the majority of factors that sway a song's reception on Spotify. The R2 number also points towards potential unknowns, implying that about 30% of influential factors remain elusive.

This analytical journey into the realm of Spotify's song popularity, armed with the Random Forest Regression model, has been helpful and promising. While the initial results pave a solid foundation, there is potential refinements and enhancements. Future attempts could entail a deeper dive into feature engineering, exploring additional datasets, or even venturing into more intricate predictive models.



Here, I utilized a visual representation to depict the relationship between the actual and predicted values of song popularity using random forest regression. The scatter plot, representing this relationship, was accompanied by a regression line that provided deeper insights into the prediction accuracy and tendencies of the model.

One thing that stands out to me about this visual representation is the slope of the regression line, which is slightly more than three-fourths. This slope signifies the rate of change in the predicted values for every unit change in the actual values. In essence, for every rise of one unit in the predicted popularity, the actual value increases by a little more than three-fourths of a unit.

This observation suggests a couple of essential points. First, the model does capture a strong linear relationship between the actual and predicted values, given the prominent inclination of the regression line. However, the fact that the slope is not precisely one indicates that the model may slightly underestimate the popularity for songs that are highly popular, and overestimate for those that are less popular. This pattern might arise due to various reasons. There could be certain influential factors or nuances in the data that the model has not fully grasped. Also, the nature of song popularity itself might not always follow a perfect linear trend when considering all the features in the dataset.

In conclusion, the visual representation of actual vs. predicted popularity and the insights drawn from the regression line's slope have been very helpful. They not only highlight the strengths and areas of improvement for the model but also help illustrate what future enhancements are needed to determine song popularity on Spotify.

Analysis

Model Description:

To predict song popularity based on a range of musical features derived from a Spotify dataset, I utilized the Random Forest Classifier and Random Forest Regression models—two advanced machine learning algorithms known for their robustness and versatility. These models were chosen due to their ability to handle high dimensional spaces and their adaptability to different types of prediction problems, offering insights into the intricate patterns underlying song popularity.

The Random Forest Classifier was applied to categorize songs into distinct popularity levels or brackets. This ensemble learning model, which operates by constructing multiple decision trees during training and outputting, helped in unraveling the multifaceted relationships between various song features and their collective impact on the popularity levels. By leveraging the classifier, I was able to assess how different musical components interplay to push a song into varying degrees of popularity, this helped to show the qualitative aspects of song popularity.

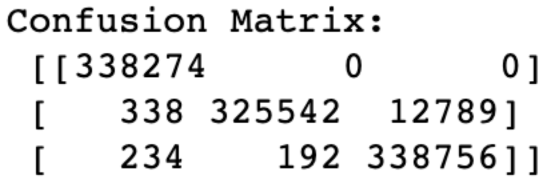
In contrast, the Random Forest Regression model was employed to predict the continuous popularity scores of the songs. It combines the predictions from multiple decision trees to generate a final prediction that is more generalized and less susceptible to the noise in the training data. This model illuminated the quantitative nuances of song popularity by rendering a numerical measure of popularity based on the input features. It allowed for the estimation of a song's popularity score, lending a more precise insight into the probable success of a song on Spotify.

These models were trained and validated using the dataset, and hyperparameter tuning was performed to optimize their predictive performance. The results from these Random Forest models were helpful in understanding the inherent dynamics of song popularity, revealing both the categorical distinctions and numerical variations in popularity that are influenced by the various attributes of a song. The findings also enabled the assessment of a song's potential popularity as well as offered valuable insights into the integral features and elements that predominantly contribute to a song's success on the streaming platform.

Model Results:

Random Forest Classifier:

The Random Forest Classifier was utilized to explore the feasibility of building a predictive model focusing on the Spotify platform, aiming to accurately forecast the popularity of songs. The model was executed with a substantial dataset encompassing diverse features inherent to the songs listed on Spotify.



```
Confusion Matrix:
 [[338274      0      0]
 [   338 325542  12789]
 [   234    192 338756]]
```

The confusion matrix showcases the performance of the model by distinguishing between High, Low, and Moderate popularity songs. The diagonal elements represent the number of correct classifications for each class. The off-diagonal elements provide insights into the misclassifications.

High popularity songs:

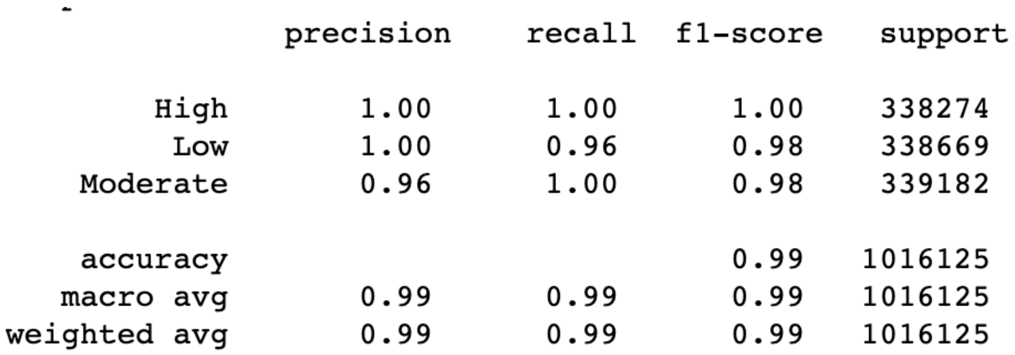
The model predicted 338,274 songs accurately as high popularity, with no misclassifications.

Low popularity Songs:

325,542 songs were correctly classified as low popularity, but there were 12,789 instances misclassified as Moderate and 338 as High.

Moderate Popularity Songs:

The model accurately classified 338,756 songs as Moderate popularity, with minor misclassifications as High and Low.



```
              precision    recall  f1-score   support

        High       1.00      1.00      1.00    338274
         Low       1.00      0.96      0.98    338669
    Moderate       0.96      1.00      0.98    339182

    accuracy                           0.99   1016125
   macro avg       0.99      0.99      0.99   1016125
weighted avg       0.99      0.99      0.99   1016125
```

The model achieved an accuracy of approximately 98.67%, indicating a highly reliable and precise classification of song popularity on Spotify. The classification report provides a detailed overview of the performance of the model in terms of Precision, Recall, and F1-score for each class:

High popularity:

Precision: 1.00, Recall: 1.00, F1-Score: 1.00

Low popularity:

Precision: 1.00, Recall: 0.96, F1-ScoreL 0.98

Moderate Popularity:

Precision: 0.96, Recall: 1.00, F1-Score: 0.98

The precision, recall, and F1-score values near 1.00 for all three classes indicate that the model has a balanced and proficient classification capability, with minimal false positives and false negatives.

The analyzed Random Forest Classifier manifests a high level of precision in predicting the popularity of songs on Spotify with an accuracy of 98.67%. This outcome substantiates the affirmative response to the research question, emphasizing the model's capability to forecast the popularity of a song accurately and reliably on Spotify, making it an asset in the realm of music analytics.

Random Forest Regression:

In response to the research question, a Random Forest Regression model was developed and deployed to predict the popularity of songs on Spotify, focusing on determining the quantitative aspect of song popularity. The following analysis offers insights into the model's performance based on various regression evaluation metrics.

Mean Squared Error: 76.64

The mean squared error is a measure of the average of the squares of the errors, representing the difference between the actual observed outcomes and the predictions made by the model. A mean squared error of 76.64 indicates that, on average, the model's predictions are approximately 76.64 units squared away from the actual values, signifying a moderate level of error.

Mean Absolute Error: 6.48

The mean absolute error is a measure of the average absolute difference between the observed actual outcomes and predictions made by the model. A mean absolute error of 6.48 denotes that, on average, the predictions made by the model are 6.48 units away from the actual values, indicating a relatively low level of error.

Root Mean Squared Error: 8.75

The root mean squared error is the square root of the average of squared differences between the actual observed values and the predictions made by the model. A root mean squared error value of 8.75 signifies a moderate level of prediction error, implying that the model's predictions are, on average, approximately 8.75 units away from the observed values.

R-squared Score: 0.7

The R2 score represents the proportion of the variance in the dependent variable that is predictable from the independent variables. An R2 score of 0.70 means that 70% of the variability in song popularity can be explained by the model, denoting a good level of model fit.

The Random Forest Regression model, with an R-squared score of 0.70, a moderate mean absolute error and root mean squared error, and a relatively low mean absolute error, has showcased a substantial capability in forecasting the popularity of songs on Spotify. While there is some degree of error in predictions, the model is reliable in capturing 70% of the variability in song popularity, emphasizing its proficiency in unraveling the complexities of song popularity on the streaming platform. This analysis underscores the affirmative resolution of the research question, demonstrating that it is possible to construct a predictive model with substantial accuracy in forecasting the popularity of a song on Spotify.

Ethical Recommendation

With the growing influence of machine learning and data analytics, various industries, including the music industry, are exploring innovative tools to forecast trends. One pursuit in particular is the creation of predictive models to estimate the potential popularity of songs on platforms like Spotify. As with any technological advance, this pursuit contains possible ethical challenges.

To begin, data privacy is a prominent concern. The vast array of data that trains these models needs to be handled with the utmost respect for privacy. Especially in the music industry, where the line between personal and professional often blurs, ensuring that data—such as artists' personal details or those of lesser-known indie artists—is anonymized is crucial. Measures like this ensure that individual rights aren't compromised for the sake of data-driven forecasts.

The next potential ethical problem revolves around the model's inherent biases and its representation capacities. Historical data is a double-edged sword; it provides a source for training models but can also be a repository of current entrenched biases. Certain genres or regions that have been historically underrepresented in global charts might inadvertently influence predictions, leading to a continuous cycle of underrepresentation. However, the diversity of music demand from around the world helps ensure that predictive models encompass a wide range of genres, languages, and cultures. This ensures fairness in the representation and gives every song, regardless of its origin, a fair shot at being recognized.

Moreover, transparency is not just about the clear articulation of methodologies but also about trust. When artists understand the criteria and mechanisms behind their song's predicted popularity, they are more likely to trust and engage with the platform. However, it's essential to strike a balance. While these models can be great tools, they shouldn't overshadow human judgment. Over-reliance on algorithms can stifle diversity and creativity, leading to a homogenized music scene. The ethical use of technology also requires a forward-looking approach. Predictive models should avoid skewing listener preferences or promoting particular tracks unduly. Such manipulation can limit musical diversity and deprive listeners of the vast tapestry of global music.

In conclusion, as predictive modeling in music platforms like Spotify gains traction, it's vital to navigate potential ethical problems diligently. Ensuring data privacy, mitigating biases, fostering transparency, and promoting fair representation are the pillars on which these models should stand. As technology and music intertwine further, a strong willingness to prevent ethical problems will help to ensure a smooth future for the music industry.

Challenges

Originally, one of my research questions was to see if I could create a recommendation system with this dataset. In the vastness of data analytics, the pull to use recommendation systems is persuasive for many companies. Platforms like Netflix, Amazon, Indeed, and Spotify have harnessed the power of these systems to significantly enhance user experience. When I started my research into the Spotify dataset, the idea of designing a recommendation system came up naturally. With attributes ranging from danceability to loudness, it appeared to me that the dataset held enough variables to easily explore and answer research questions. However, as I dove deeper, the challenges soon became apparent.

The biggest issue was the lack of user-specific data. A recommendation system hinges on understanding user behaviors, preferences, and interactions. To truly grasp and recommend what a user might like next, it's essential to know what they've liked before. The Spotify dataset had plenty of song characteristics but did not offer this essential user-oriented insight, which would have made the process easier and allowed for even more accurate predictions.

Furthermore, I realized that recommendation systems had many more intricacies and complexities than I originally expected when I began my research. Recommendation systems are often underpinned by sophisticated algorithms, including but not limited to collaborative filtering or matrix factorization. While the dataset allowed for a comprehensive exploration of song attributes and their inter-relationships, applying this knowledge to the vast and complex field of recommendation algorithms was a daunting task.

The dataset's uniformity posed another challenge. While it offered an extensive look into song attributes, it lacked diversity in terms of user interactions and preferences across these songs. Without this diversity, discerning patterns for effective song recommendations was a more difficult task. Additionally, it was a challenge to attempt recommending songs for new users with no prior listening history. Given that the dataset did not offer a panoramic view of global listening trends, addressing this problem was hard to solve.

Lastly, scalability was a looming concern. Beyond the right algorithms and data, a recommendation system should be equipped to handle vast numbers of users and recommendations. Ensuring timely, relevant, and scalable song suggestions with the limitations of the dataset is also a big challenge.

In conclusion, while the Spotify dataset was a rich resource for understanding song attributes and their relation to popularity, it presented several challenges when pivoting towards creating a recommendation system. These challenges emphasized the complex nature of recommendation systems and the depth of data and expertise they require.

Recommendations

Diversifying data sources is pivotal in building a predictive model for song popularity, as a sole reliance on Spotify's dataset may offer a skewed perspective. Integrating data from various platforms such as YouTube, Twitter, and traditional radio airplay metrics can lead to a more holistic understanding of a song’s resonance with audiences. Feature engineering is also a cornerstone in this process. It may also be helpful to create additional metrics like 'time since release' and 'collaborative artist impact' to enhance the model's predictive power. Addressing data anomalies like missing values or imbalanced datasets is equally essential to avoid inadvertent bias in the model.

Beyond regression models and random forests, exploring deep learning models like Recurrent Neural Network or Long Short-Term Memory networks could be beneficial due to their proficiency in understanding sequences and time series data. These models could potentially capture the nuances of music trends more effectively due to the intricate nature of musical attributes. Additionally, incorporating temporal considerations in data splitting can simulate a more authentic prediction scenario, allowing for training on historical data and testing on contemporary tracks. This approach, coupled with hyperparameter tuning, may refine the model's accuracy, helping in better anticipation of its real-world performance.

Furthermore, the incorporation of human expertise could be helpful. Collaboration with music industry veterans can add depth and refinement to the predictive model by bridging the gap between data-driven predictions and the intangible elements inherent in the music world. Their insights may assist the model in becoming more accurate by shedding light on the subtle aspects of the music industry that are not easily quantified.

Developing a predictive model for song popularity involves an intricate interplay of data, technology, and human intuition. It requires both analytical expertise and a genuine appreciation for the art form, especially as we progress in this digital music age. Balancing analytical precision with insights from industry experts and diverse data sources can lead to more comprehensive and nuanced understandings of song popularity.

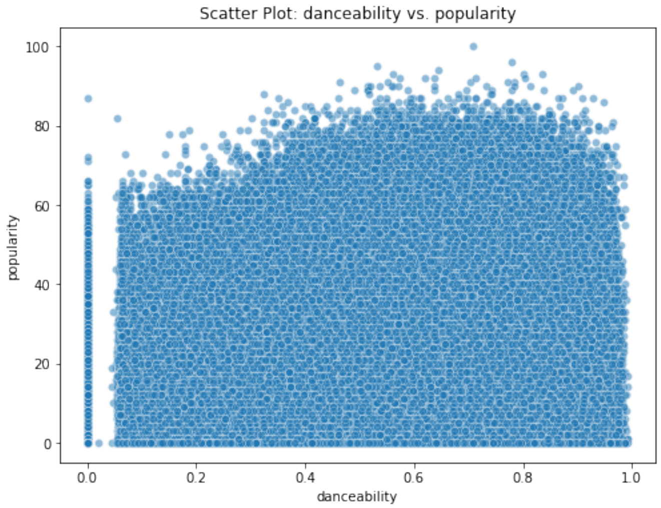
References

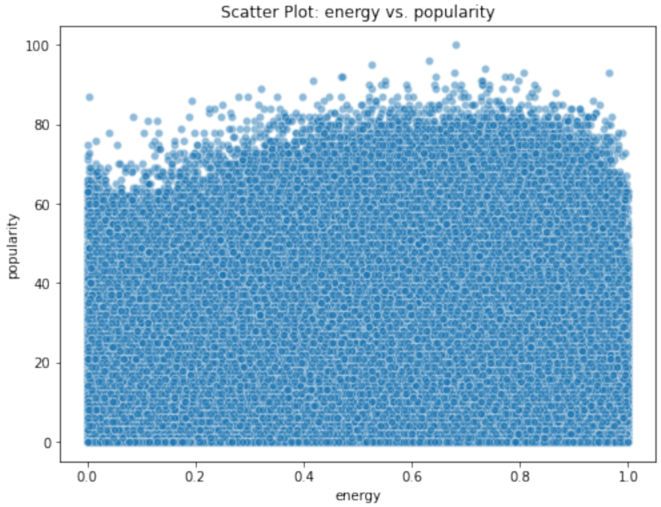
Nijkamp, Rutger (2018) Prediction of product success: explaining song popularity by audio features from Spotify data.

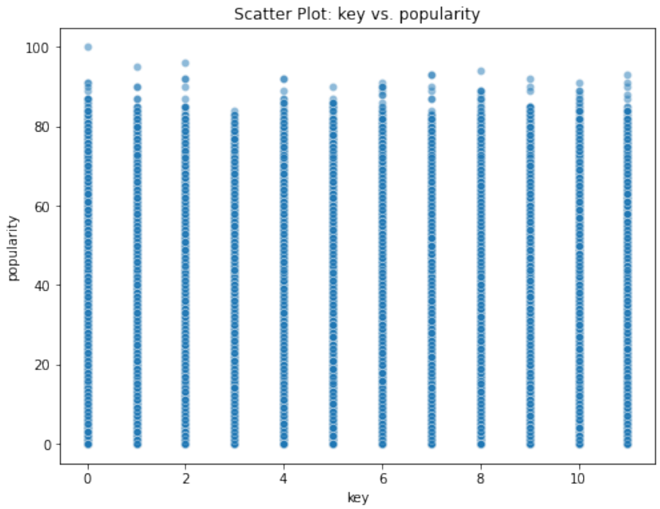
Rubinstein, B. I. P., & Wood, D. (2013). Data Valuation for Data Markets. EC '13 Proceedings of the Fourteenth ACM Conference on Electronic Commerce, 1-16.

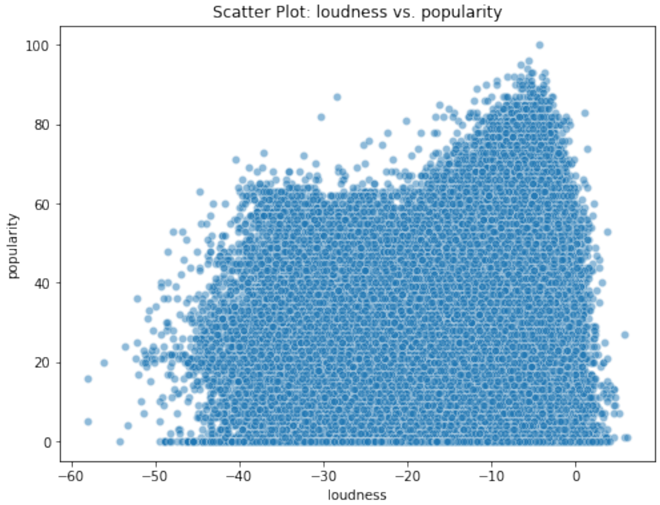
Burrell, J. (2016). How the machine ‘thinks’: Understanding opacity in machine learning algorithms. Big Data & Society, 3(1), 1-12.

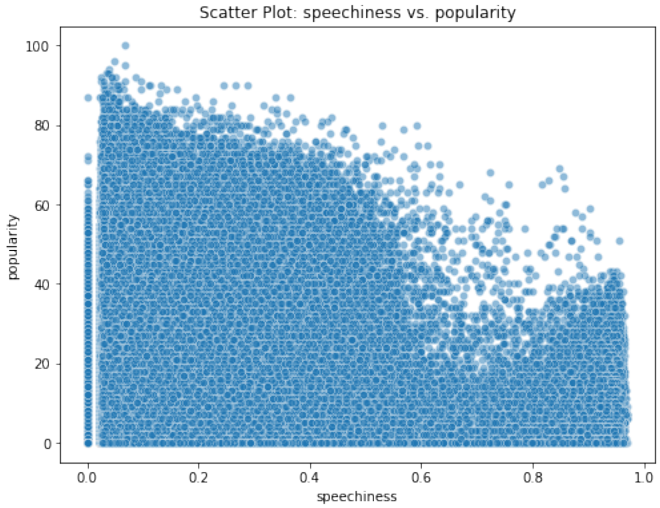
Appendix

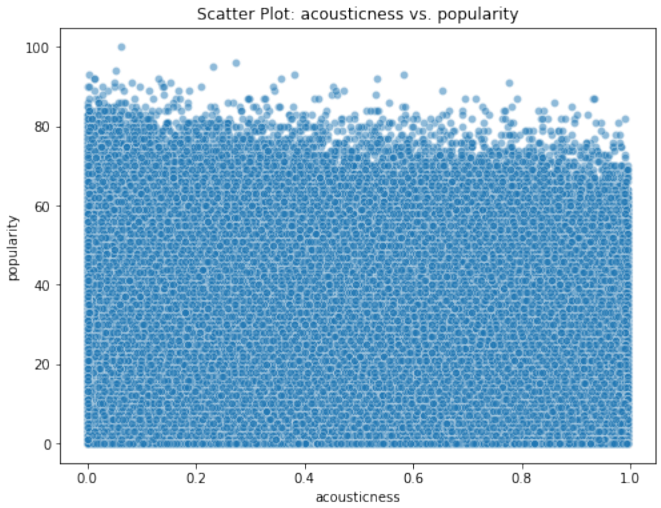

Scatter Plot: danceability vs. popularity
popularity
danceability
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

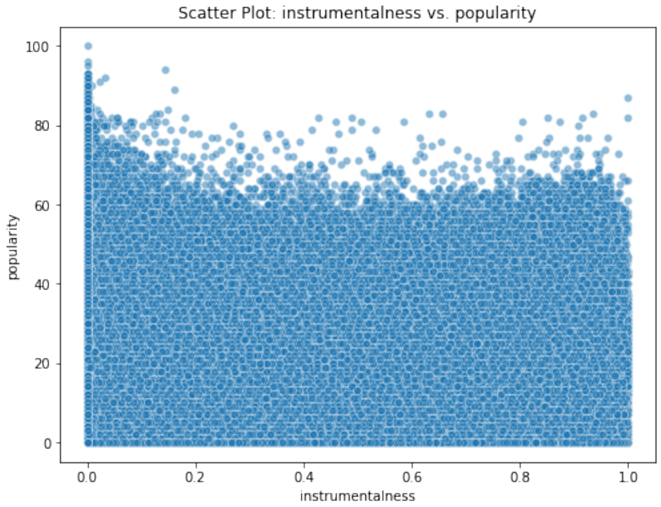

Scatter Plot: energy vs. popularity
popularity
energy
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

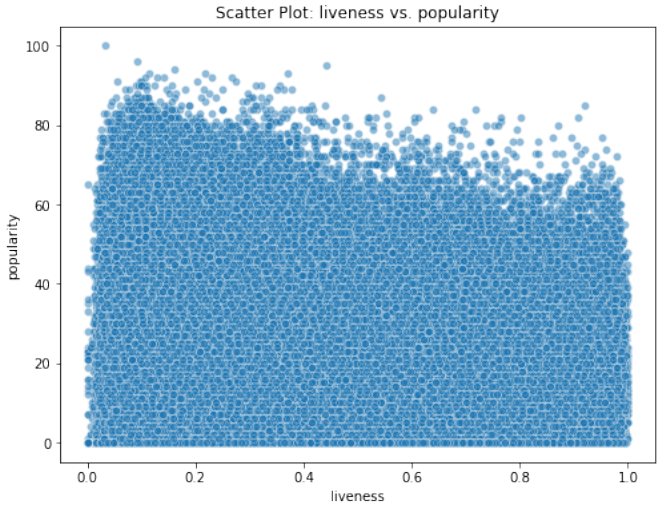

Scatter Plot: key vs. popularity
popularity
key
0
20
40
60
80
100
0
2
4
6
8
10

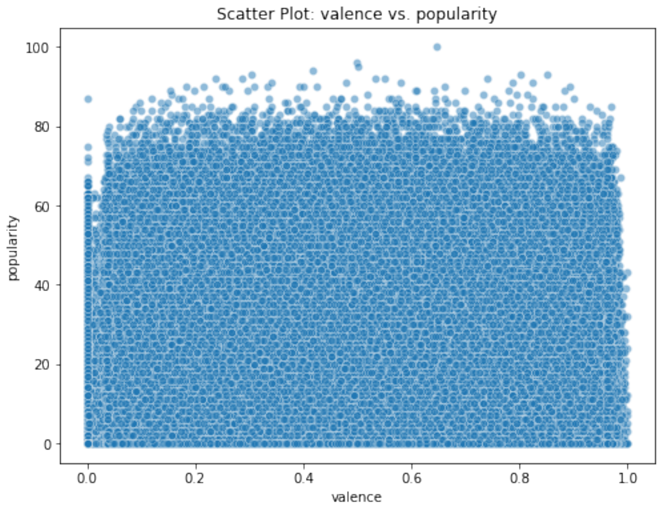

Scatter Plot: loudness vs. popularity
popularity
loudness
0
20
40
60
80
100
−60
−50
−40
−30
−20
−10
0

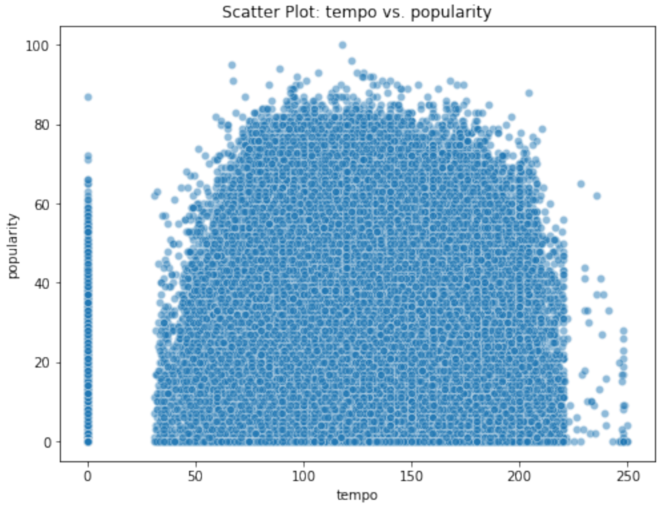

Scatter Plot: speechiness vs. popularity
popularity
speechiness
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

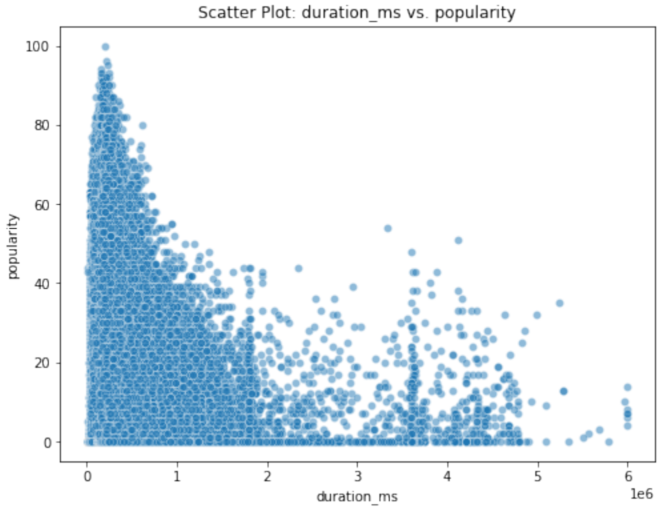

Scatter Plot: acousticness vs. popularity
popularity
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
acousticness


Scatter Plot: instrumentalness vs. popularity
popularity
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
instrumentalness


Scatter Plot: liveness vs. popularity
popularity
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
liveness


Scatter Plot: valence vs. popularity
popularity
valence
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0


Scatter Plot: tempo vs. popularity
popularity
tempo
0
20
40
60
80
100
0
50
100
150
200
250


Scatter Plot: duration_ms vs. popularity
popularity
duration_ms
0
20
40
60
80
100
0
1
2
3
4
5
6
1e6

Source Code

In [160]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('spotify_data.csv')
columns_to_drop = ['Unnamed: 0', 'track_id','mode','time_signature']
df.drop(columns=columns_to_drop, inplace=True)
df
```

Out[160]:

	artist_name	track_name	popularity	year	genre	danceability	energy	key	loudness	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms
0	Jason Mraz	I Won't Give Up	68	2012	acoustic	0.483	0.303	4	-10.058	0.0429	0.6940	0.000000	0.1150	0.1390	133.406	240166
1	Jason Mraz	93 Million Miles	50	2012	acoustic	0.572	0.454	3	-10.286	0.0258	0.4770	0.000014	0.0974	0.5150	140.182	216387
2	Joshua Hyslop	Do Not Let Me Go	57	2012	acoustic	0.409	0.234	3	-13.711	0.0323	0.3380	0.000050	0.0895	0.1450	139.832	158960
3	Boyce Avenue	Fast Car	58	2012	acoustic	0.392	0.251	10	-9.845	0.0363	0.8070	0.000000	0.0797	0.5080	204.961	304293
4	Andrew Belle	Sky's Still Blue	54	2012	acoustic	0.430	0.791	6	-5.419	0.0302	0.0726	0.019300	0.1100	0.2170	171.864	244320
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1159759	Nicola Conte	Black Spirits	4	2011	trip-hop	0.373	0.742	10	-6.453	0.0736	0.3250	0.000141	0.1590	0.5220	107.951	344013
1159760	Nicola Conte	Quiet Dawn	3	20	trip-hop	0.516	0.675	7	-7.588	0.0326	0.7880	0.000129	0.1300	0.2640	119.897	285067

	artist_name	track_name	popularity	year	genre	danceability	energy	key	loudness	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms
				11												
1159761	Amon Tobin	Morning Ms Candis	2	2011	trip-hop	0.491	0.440	5	-8.512	0.0274	0.4770	0.003130	0.0936	0.0351	100.076	214253
1159762	Peace Orchestra	Happy Christmas (War Is Over)	0	2011	trip-hop	0.480	0.405	0	-13.343	0.0276	0.4310	0.000063	0.1250	0.2020	133.885	239133
1159763	Mo' Horizons	Hit the Road Jack (Pé Na Éstrada)	3	2011	trip-hop	0.782	0.861	1	-7.292	0.1250	0.2200	0.000008	0.0581	0.8570	89.987	212227

1159764 rows × 16 columns

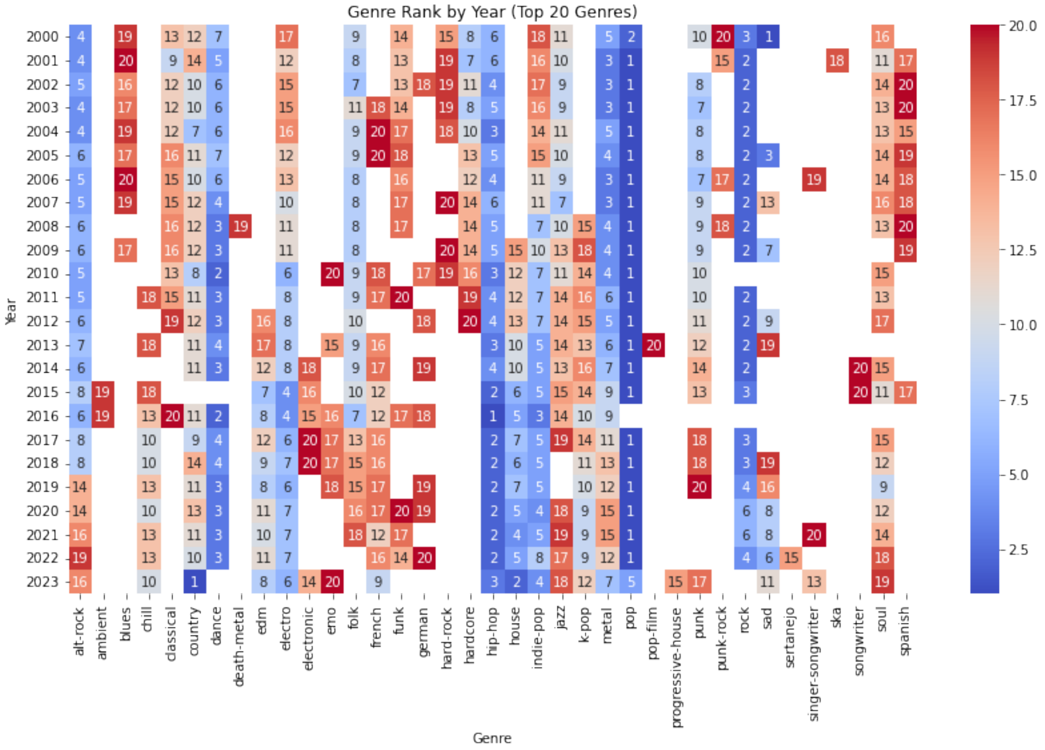
In [161]:

```
genre_year_avg_popularity = df.groupby(['year',
'genre'])['popularity'].mean().reset_index()

# Rank the genres within each year based on their average popularity
genre_year_avg_popularity['rank'] =
genre_year_avg_popularity.groupby('year')['popularity'].rank(ascending=False)

# Filter the top 20 genres for each year
top_20_genres_by_year =
genre_year_avg_popularity[genre_year_avg_popularity['rank'] <= 20]

# Sort the DataFrame by 'year' and then 'rank'
top_20_genres_by_year.sort_values(by=['year', 'rank'], ascending=[True,
True], inplace=True)

# The 'top_20_genres_by_year' DataFrame contains the top 20 popular genres
for each year, sorted by year and rank
print(top_20_genres_by_year)
      year     genre  popularity  rank
66    2000       sad   61.000000   1.0
56    2000       pop   44.899054   2.0
63    2000      rock   40.954023   3.0
2     2000  alt-rock   33.763393   4.0
49    2000     metal   33.328000   5.0
```

```
...    ...       ...        ...   ...
1838  2023  alt-rock  30.723705  16.0
1897  2023      punk  30.467033  17.0
1883  2023      jazz  29.201983  18.0
1910  2023      soul  28.746835  19.0
1862  2023       emo  28.540049  20.0

[480 rows x 4 columns]
```

In [162]:

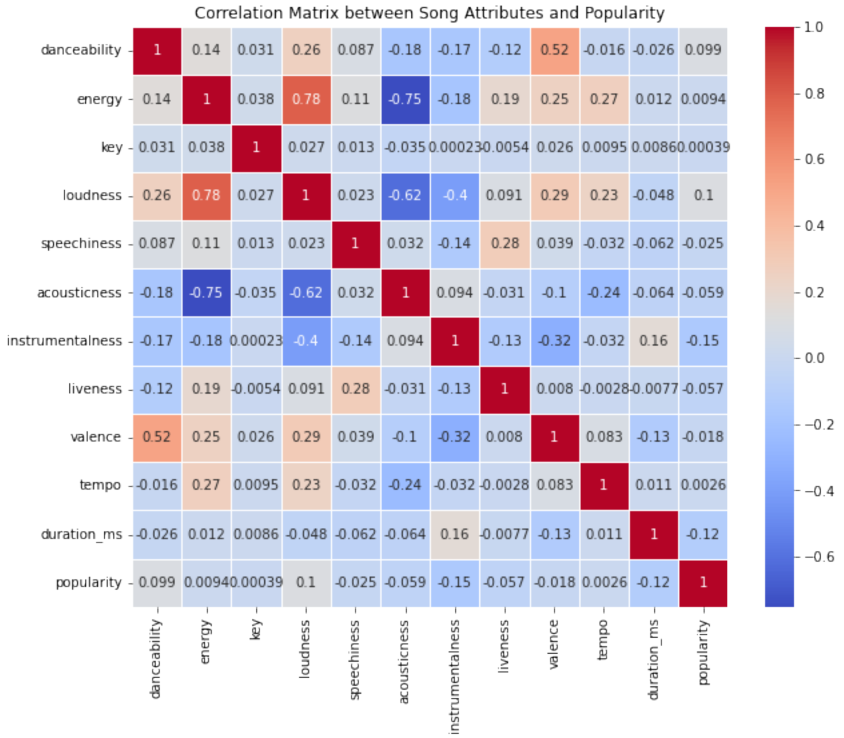
```
# Pivot the filtered DataFrame to create a genre-year matrix with rank as
values
genre_rank_matrix = top_20_genres_by_year.pivot(index='year',
columns='genre', values='rank')

# Create a heatmap to visualize the rank of genres by year
plt.figure(figsize=(12, 8))
sns.heatmap(genre_rank_matrix, cmap='coolwarm', annot=True, fmt='g',
cbar=True)
plt.title('Genre Rank by Year (Top 20 Genres)')
plt.xlabel('Genre')
plt.ylabel('Year')

# Show the plot
plt.tight_layout()
plt.show()
```



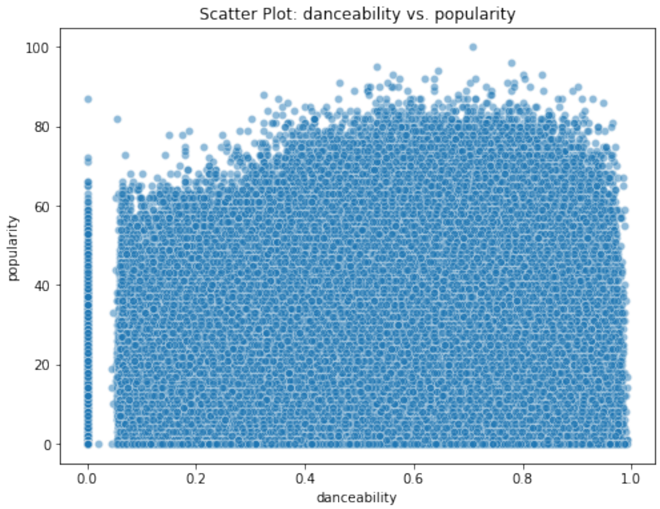
In [163]:

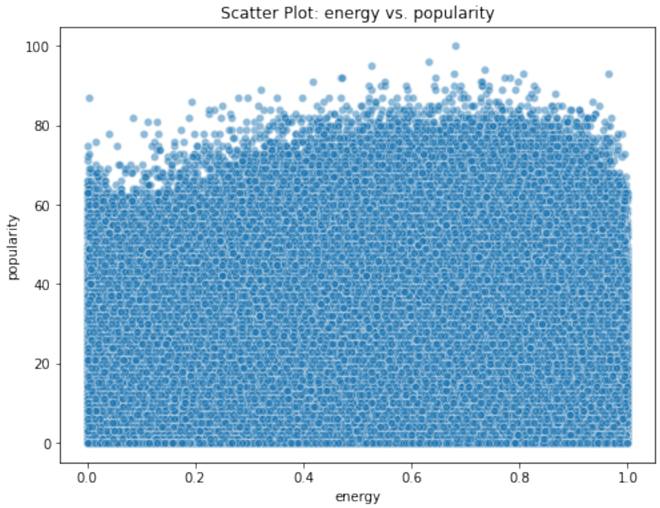
```
# Select the relevant attributes and 'Popularity'
selected_attributes = ['danceability', 'energy', 'key',
'loudness','speechiness','acousticness','instrumentalness','liveness','valenc
e','tempo','duration_ms']
data = df[selected_attributes + ['popularity']]
```

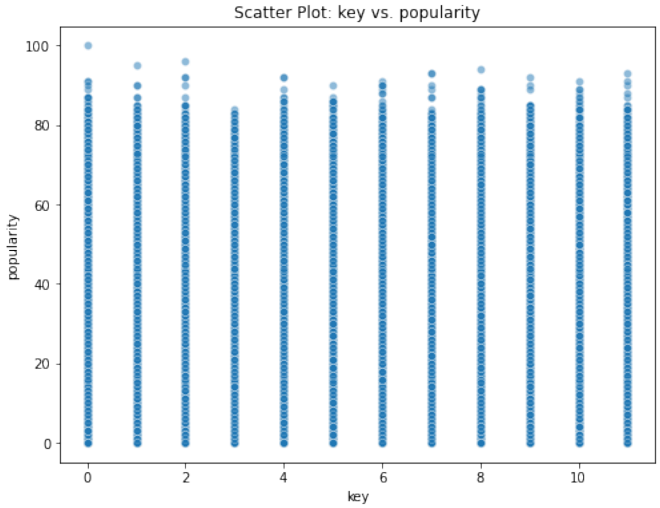
```
# Calculate the correlation matrix
correlation_matrix = data.corr()

# Create a heatmap to visualize the correlations
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix between Song Attributes and Popularity')
plt.show()

# Scatter plots to visualize each attribute vs. Popularity
for attribute in selected_attributes:
    plt.figure(figsize=(8, 6))
    sns.scatterplot(data=df, x=attribute, y='popularity', alpha=0.5)
    plt.title(f'Scatter Plot: {attribute} vs. popularity')
    plt.xlabel(attribute)
    plt.ylabel('popularity')
    plt.show()
```

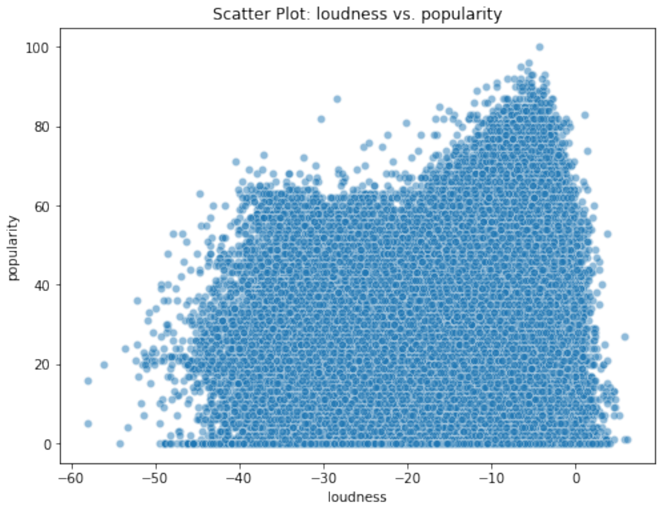


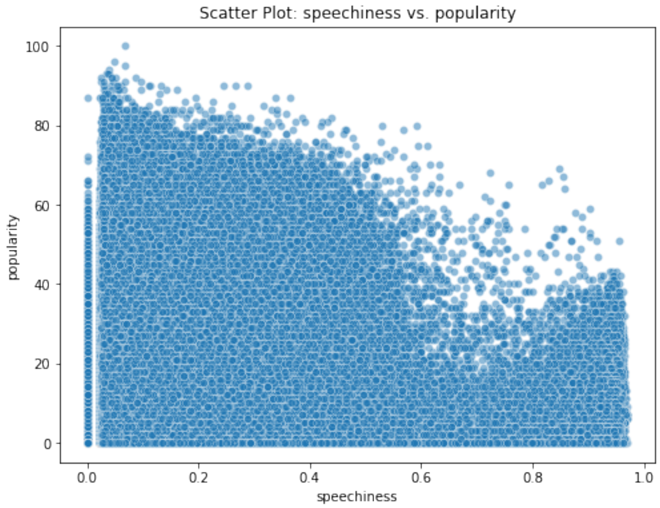
	danceability	energy	key	loudness	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms	popularity
danceability	1	0.14	0.031	0.26	0.087	-0.18	-0.17	-0.12	0.52	-0.016	-0.026	0.099
energy	0.14	1	0.038	0.78	0.11	-0.75	-0.18	0.19	0.25	0.27	0.012	0.0094
key	0.031	0.038	1	0.027	0.013	-0.035	0.00023	-0.0054	0.026	0.0095	0.0086	0.00039
loudness	0.26	0.78	0.027	1	0.023	-0.62	-0.4	0.091	0.29	0.23	-0.048	0.1
speechiness	0.087	0.11	0.013	0.023	1	0.032	-0.14	0.28	0.039	-0.032	-0.062	-0.025
acousticness	-0.18	-0.75	-0.035	-0.62	0.032	1	0.094	-0.031	-0.1	-0.24	-0.064	-0.059
instrumentalness	-0.17	-0.18	0.00023	-0.4	-0.14	0.094	1	-0.13	-0.32	-0.032	0.16	-0.15
liveness	-0.12	0.19	-0.0054	0.091	0.28	-0.031	-0.13	1	0.008	-0.0028	-0.0077	-0.057
valence	0.52	0.25	0.026	0.29	0.039	-0.1	-0.32	0.008	1	0.083	-0.13	-0.018
tempo	-0.016	0.27	0.0095	0.23	-0.032	-0.24	-0.032	-0.0028	0.083	1	0.011	0.0026
duration_ms	-0.026	0.012	0.0086	-0.048	-0.062	-0.064	0.16	-0.0077	-0.13	0.011	1	-0.12
popularity	0.099	0.0094	0.00039	0.1	-0.025	-0.059	-0.15	-0.057	-0.018	0.0026	-0.12	1

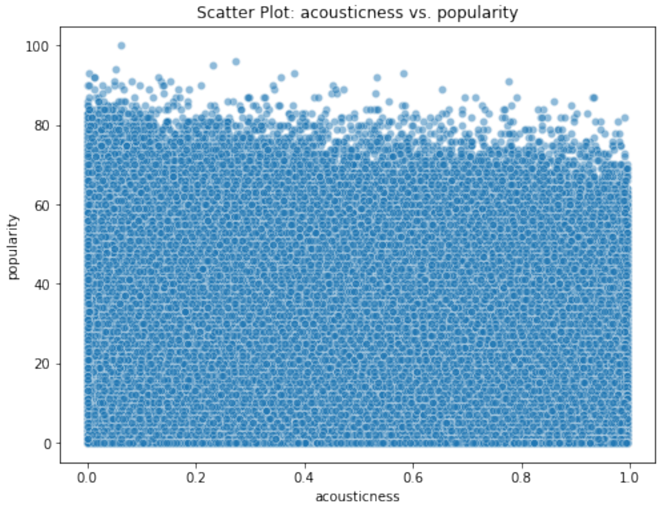

Scatter Plot: danceability vs. popularity
popularity
0
20
40
60
80
100
danceability
0.0
0.2
0.4
0.6
0.8
1.0

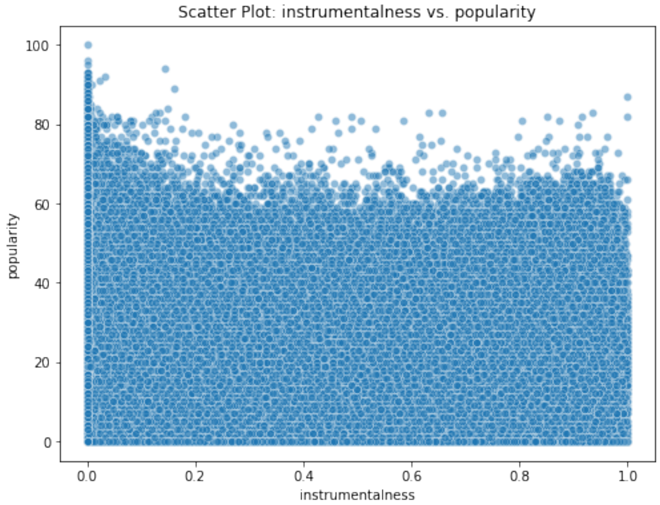

Scatter Plot: energy vs. popularity
popularity
0
20
40
60
80
100
energy
0.0
0.2
0.4
0.6
0.8
1.0

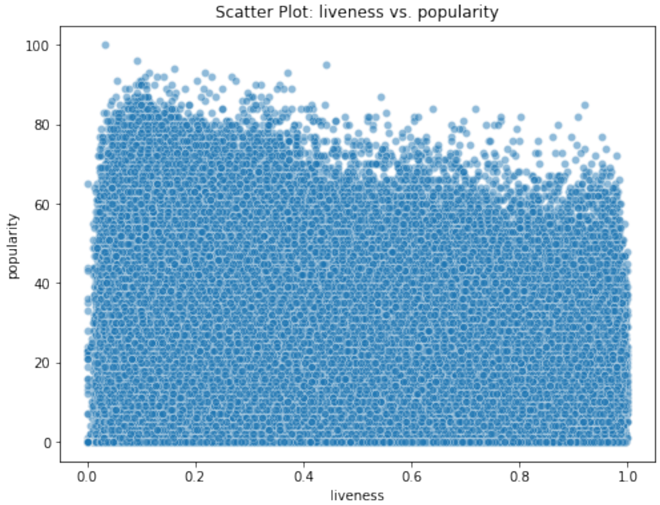

Scatter Plot: key vs. popularity
popularity
0
20
40
60
80
100
key
0
2
4
6
8
10

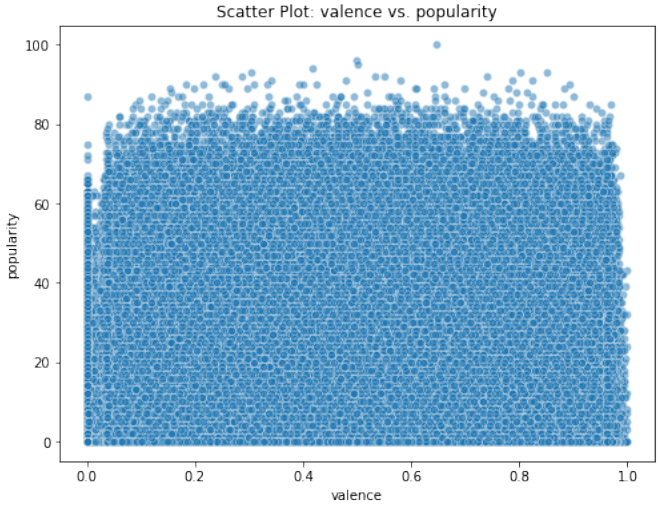

Scatter Plot: loudness vs. popularity

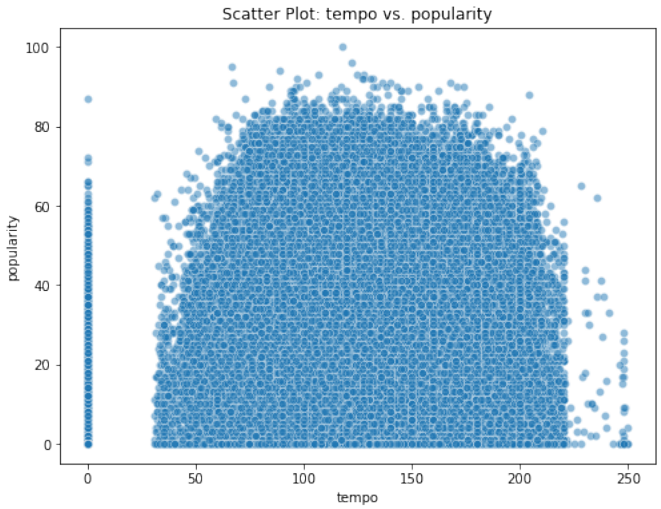

Scatter Plot: speechiness vs. popularity

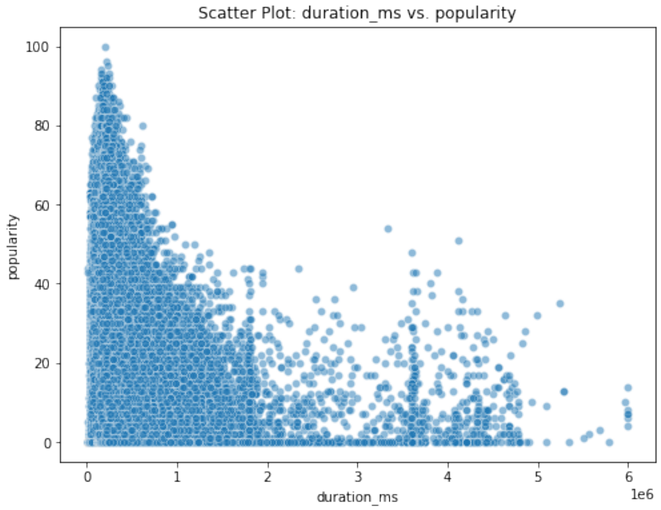

Scatter Plot: acousticness vs. popularity


Scatter Plot: instrumentalness vs. popularity
popularity
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
instrumentalness


Scatter Plot: liveness vs. popularity
popularity
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
liveness


Scatter Plot: valence vs. popularity
popularity
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
valence


Scatter Plot: tempo vs. popularity
popularity
tempo
0
20
40
60
80
100
0
50
100
150
200
250


Scatter Plot: duration_ms vs. popularity
popularity
duration_ms
0
20
40
60
80
100
0
1
2
3
4
5
6
1e6

In [32]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, classification_report,confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
```

In [33]:

```
df=pd.read_csv('spotify_data.csv')
```

In [39]:

```
# Convert the 'duration_ms' column from milliseconds to minutes
df['duration_minutes'] = df['duration_ms'] / 60000

# Display the DataFrame with the 'duration_minutes' column
print(df[['duration_ms', 'duration_minutes']])
         duration_ms  duration_minutes
0             240166          4.002767
1             216387          3.606450
2             158960          2.649333
3             304293          5.071550
4             244320          4.072000
...              ...               ...
1159759       344013          5.733550
1159760       285067          4.751117
1159761       214253          3.570883
1159762       239133          3.985550
1159763       212227          3.537117

[1159764 rows x 2 columns]
```

In [40]:

```
# unnamed is an un-necessary column so drop it
df.drop(["Unnamed: 0",'track_name','track_id','duration_ms'], axis=1 ,
inplace= True )
```

In [41]:

```
df.head()
```

Out[41]:

	artist_name	popularity	year	genre	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature	duration_minutes
0	Jason Mraz	68	2012	acoustic	0.483	0.303	4	-10.058	1	0.0429	0.6940	0.000000	0.1150	0.139	133.406	3	4.002767
1	Jason Mraz	50	2012	acoustic	0.572	0.454	3	-10.286	1	0.0258	0.4770	0.000014	0.0974	0.515	140.182	4	3.606450

	artist_name	popularity	year	genre	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature	duration_minutes
2	Joshua Hyslop	57	2012	acoustic	0.409	0.234	3	-13.711	1	0.0323	0.3380	0.000050	0.0895	0.145	139.832	4	2.649333
3	Boyce Avenue	58	2012	acoustic	0.392	0.251	10	-9.845	1	0.0363	0.8070	0.000000	0.0797	0.508	204.961	4	5.071550
4	Andrew Belle	54	2012	acoustic	0.430	0.791	6	-5.419	0	0.0302	0.0726	0.019300	0.1100	0.217	171.864	4	4.072000

In [43]:

```
# remove Nulls
df=df.dropna()
```

Data Visualization

In [51]:

```
# Get the value counts for the "artist_name" column
artist_counts = df['artist_name'].value_counts()

# Filter the counts to include only values with count less than 20
artist_counts_less_than_20 = artist_counts[artist_counts < 50]

# Display the value counts for artist names with count less than 20
print(artist_counts_less_than_20)
The Carburetors         49
Agni                    49
Kavita Krishnamurthy    49
Lit                     49
Kash Trivedi            49
                        ..
Gaia Consort             1
Dzej Ramadanovski        1
Goca Trzan               1
Robbie O'Connell         1
Sleep Therapist          1
Name: artist_name, Length: 58294, dtype: int64
```

In [52]:

```
# Convert artist with less than 50 appearance to other category

# Get the value counts for the "artist_name" column
artist_counts = df['artist_name'].value_counts()

# Identify artists with counts less than 50
artists_less_than_50 = artist_counts[artist_counts < 50].index.tolist()

# Label the artists as "Others" in the DataFrame
```

```
df.loc[df['artist_name'].isin(artists_less_than_50), 'artist_name'] =
'Others'

# Get the updated value counts
updated_artist_counts = df['artist_name'].value_counts()

# Display the updated value counts
print(updated_artist_counts)
Others                    549622
Traditional                 4058
Grateful Dead               2320
Johann Sebastian Bach       2125
Giacomo Meyerbeer           1345
                           ...
Area 11                       50
Taio Cruz                     50
Sons Of Maria                 50
Savoy                         50
CalledOut Music               50
Name: artist_name, Length: 5866, dtype: int64
```

In [54]:

```
# Select only the numerical columns for correlation analysis
numerical_columns = df.select_dtypes(include=['float64'])

# Calculate the correlation matrix
correlation_matrix = numerical_columns.corr()

# Select only the numerical columns for correlation analysis (excluding
'popularity')
numerical_columns =
df.select_dtypes(include=['float','int']).drop(columns=['popularity'])

# Calculate the correlation matrix
correlation_matrix = numerical_columns.corr()

# Create a mask for highly correlated features (excluding the diagonal)
high_corr_mask = (correlation_matrix.abs() >= 0.5) & (correlation_matrix < 1)

# Find pairs of highly correlated features
high_corr_pairs = [(col1, col2) for col1 in high_corr_mask.columns for col2
in high_corr_mask.columns if high_corr_mask.loc[col1, col2]]

# Remove one of each highly correlated pair
columns_to_drop = set()
for col1, col2 in high_corr_pairs:
    if col1 not in columns_to_drop and col2 not in columns_to_drop:
        # Decide which feature to drop (e.g., based on domain knowledge or
other criteria)
        # Here, we drop the one with the longer column name
        if len(col1) > len(col2):
            columns_to_drop.add(col1)
        else:
            columns_to_drop.add(col2)

# Drop the highly correlated features from the DataFrame
df_filtered = df.drop(columns=list(columns_to_drop))
```

```
# Display the DataFrame with highly correlated features removed
df_filtered.head()
```

Out[54]:

	artist_name	popularity	year	genre	energy	key	mode	speechiness	instrumentalness	liveness	valence	tempo	time_signature	duration_minutes
0	Jason Mraz	68	2012	acoustic	0.303	4	1	0.0429	0.000000	0.1150	0.139	133.406	3	4.002767
1	Jason Mraz	50	2012	acoustic	0.454	3	1	0.0258	0.000014	0.0974	0.515	140.182	4	3.606450
2	Joshua Hyslop	57	2012	acoustic	0.234	3	1	0.0323	0.000050	0.0895	0.145	139.832	4	2.649333
3	Boyce Avenue	58	2012	acoustic	0.251	10	1	0.0363	0.000000	0.0797	0.508	204.961	4	5.071550
4	Andrew Belle	54	2012	acoustic	0.791	6	0	0.0302	0.019300	0.1100	0.217	171.864	4	4.072000

Label Encoding

In [55]:

```
from sklearn.preprocessing import LabelEncoder


# convert artist name to one hot encoding
categorical_columns = ['artist_name']

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Apply label encoding to each categorical column
for column in categorical_columns:
    df_filtered[column] = label_encoder.fit_transform(df_filtered[column])

# Display the DataFrame with encoded categorical columns
df_filtered
```

Out[55]:

	artist_name	popularity	year	genre	energy	key	mode	speechiness	instrumentalness	liveness	valence	tempo	time_signature	duration_minutes
0	2441	68	2012	acoustic	0.303	4	1	0.0429	0.000000	0.1150	0.1390	133.406	3	4.002767
1	2441	50	2012	acoustic	0.454	3	1	0.0258	0.000014	0.0974	0.5150	140.182	4	3.606450
2	2624	57	2012	acoustic	0.234	3	1	0.0323	0.000050	0.0895	0.1450	139.832	4	2.649333

	artist_name	popularity	year	genre	energy	key	mode	speechiness	instrumentalness	liveness	valence	tempo	time_signature	duration_minutes
3	731	58	2012	acoustic	0.251	10	1	0.0363	0.000000	0.0797	0.5080	204.961	4	5.071550
4	272	54	2012	acoustic	0.791	6	0	0.0302	0.019300	0.1100	0.2170	171.864	4	4.072000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1159759	3694	4	2011	trip-hop	0.742	10	0	0.0736	0.000141	0.1590	0.5220	107.951	3	5.733550
1159760	3694	3	2011	trip-hop	0.675	7	0	0.0326	0.000129	0.1300	0.2640	119.897	4	4.751117
1159761	245	2	2011	trip-hop	0.440	5	1	0.0274	0.003130	0.0936	0.0351	100.076	4	3.570883
1159762	3857	0	2011	trip-hop	0.405	0	1	0.0276	0.000063	0.1250	0.2020	133.885	3	3.985550
1159763	3502	3	2011	trip-hop	0.861	1	0	0.1250	0.000008	0.0581	0.8570	89.987	4	3.537117

1159764 rows × 14 columns

In [56]:

```
# grnre for one hot encoding
categorical_columns = ['genre']

# Apply one-hot encoding
df_encoded = pd.get_dummies(df_filtered, columns=categorical_columns)

# Display the first few rows of the encoded DataFrame
df_encoded.head()
```

Out[56]:

	artist_name	popularity	year	energy	key	mode	speechiness	instrumentalness	liveness	valence	...	genre_ska	genre_sleep	genre_songwriter	genre_soul	genre_spanish	genre_swedish	genre_tango	genre_techno	genre_trance	genre_trip-hop
(	2441	68	2012	0.303	4	1	0.0429	0.000000	0.1150	0.139	...	0	0	0	0	0	0	0	0	0	0

	artist_name	popularity	year	energy	key	mode	speechiness	instrumentalness	liveness	valence	...	genre_ska	genre_sleep	genre_songwriter	genre_soul	genre_spanish	genre_swedish	genre_tango	genre_techno	genre_trance	genre_trip-hop
1	2441	50	2012	0.454	3	1	0.0258	0.000014	0.0974	0.515	...	0	0	0	0	0	0	0	0	0	0
2	2624	57	2012	0.234	3	1	0.0323	0.000050	0.0895	0.145	...	0	0	0	0	0	0	0	0	0	0
3	731	58	2012	0.251	10	1	0.0363	0.000000	0.0797	0.508	...	0	0	0	0	0	0	0	0	0	0
4	272	54	2012	0.791	6	0	0.0302	0.019300	0.1100	0.217	...	0	0	0	0	0	0	0	0	0	0

5 rows × 95 columns

For Regression

In [57]:

```
#Drop the target column for regression
X_reg=df_encoded.drop(columns=['popularity'])
y_reg=df_encoded['popularity']
```

In [58]:

```
# Create a custom function to map popularity to categories
def map_popularity_to_category(popularity):
    if popularity > 55:
        return 'High'
    elif 40 <= popularity <= 55:
        return 'Moderate'
    else:
        return 'Low'

# Apply the mapping function to create the 'popularity_category' column
df_encoded['popularity_category'] =
df_encoded['popularity'].apply(map_popularity_to_category)

# Display the DataFrame with the new 'popularity_category' column
print(df_encoded[['popularity', 'popularity_category']].head())
```

```
   popularity popularity_category
0          68                High
1          50            Moderate
2          57                High
3          58                High
```

```
4          54            Moderate
```

In [59]:

```
df_encoded[['popularity', 'popularity_category']]
```

Out[59]:

	popularity	popularity_category
0	68	High
1	50	Moderate
2	57	High
3	58	High
4	54	Moderate
...	...	...
1159759	4	Low
1159760	3	Low
1159761	2	Low
1159762	0	Low
1159763	3	Low

1159764 rows × 2 columns

For Classification

In [60]:

```
#Drop the target column for regression
X_cls=df_encoded.drop(columns=['popularity','popularity_category'])
y_cls=df_encoded['popularity_category']
```

Sampling

In [61]:

```
# Balancing the dataset
from imblearn.over_sampling import RandomOverSampler
oversample = RandomOverSampler()
X_cls, y_cls = oversample.fit_resample(X_cls, y_cls)
```

Apply Classification Models

In [62]:

```
# Spliting the  the train and test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_cls, y_cls,
test_size=0.33, random_state=42)
```

In [63]:

```
X_train
```

Out[63]:

	artist_name	year	energy	key	mode	speechiness	instrumentalness	liveness	valence	tempo	...	genre_ska	genre_sleep	genre_songwriter	genre_soul	genre_spanish	genre_swedish	genre_tango	genre_techno	genre_trance	genre_trip-hop
2237006	2465	2020	0.465	4	0	0.0276	0.000000	0.1080	0.2910	92.929	...	0	0	0	0	0	0	0	0	0	0
1306144	3857	2022	0.570	0	0	0.1410	0.000000	0.0913	0.7600	136.048	...	0	0	0	0	0	0	0	0	0	0
1654440	3857	2022	0.763	0	1	0.2610	0.000003	0.3140	0.5250	129.994	...	0	0	0	0	0	0	0	0	0	0
2009121	3857	2011	0.241	10	1	0.0349	0.004130	0.1090	0.2030	122.115	...	0	0	0	0	0	0	0	0	0	0
2709395	3857	2021	0.724	0	0	0.0656	0.000069	0.1220	0.4880	120.226	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1692743	3857	2015	0.301	0	0	0.0441	0.421000	0.0335	0.5540	140.106	...	0	0	0	0	0	0	0	0	0	0
2356330	709	2017	0.844	9	0	0.0566	0.000019	0.1190	0.6060	153.014	...	0	0	0	0	0	0	0	0	0	0
2229084	3857	2002	0.989	5	0	0.2470	0.028500	0.5700	0.0616	187.859	...	0	0	0	0	0	0	0	0	0	0
2768307	1775	2018	0.482	8	1	0.0247	0.000001	0.1080	0.2430	92.567	...	0	0	0	0	0	0	0	0	0	0
2219110	3857	2019	0.279	1	0	0.0382	0.000000	0.1060	0.4780	90.002	...	0	0	0	0	0	0	0	0	0	0

2063039 rows × 94 columns

In [64]:

```
y_train.value_counts()
```

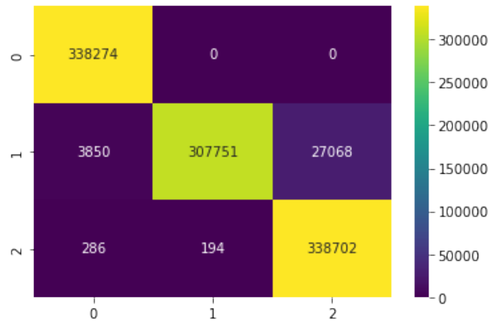
Out[64]:

```
High        688114
Low         687719
Moderate    687206
Name: popularity_category, dtype: int64
```

RANDOM FOREST CLASSIFIER

In [68]:

```
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
y_pred_rfc = rfc.predict(X_test)
print("Confusion Matrix: \n",confusion_matrix(y_test, y_pred_rfc))
print ("\n Accuracy : \n", accuracy_score(y_test, y_pred_rfc)*100)
print("\n Report : \n",classification_report(y_test, y_pred_rfc))
print('\n The  RFc Heatmap of Confusion matrix is \n')
sns.heatmap(confusion_matrix(y_test, y_pred_dtc),annot=True,
fmt='g',cmap='viridis')
plt.show()
Confusion Matrix:
 [[338274      0      0]
 [   338 325542  12789]
 [   234    192 338756]]

 Accuracy :
 98.66620740558494

 Report :
               precision    recall  f1-score   support

        High       1.00      1.00      1.00    338274
         Low       1.00      0.96      0.98    338669
    Moderate       0.96      1.00      0.98    339182

    accuracy                           0.99   1016125
   macro avg       0.99      0.99      0.99   1016125
weighted avg       0.99      0.99      0.99   1016125


 The  RFc Heatmap of Confusion matrix is
```



Regression Modeling

In [69]:

```
#Drop the target column for regression
X_reg=df_encoded.drop(columns=['popularity','popularity_category'])
y_reg=df_encoded['popularity'].astype(float)
```

In [70]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_reg, y_reg,
test_size=0.33, random_state=42)
```

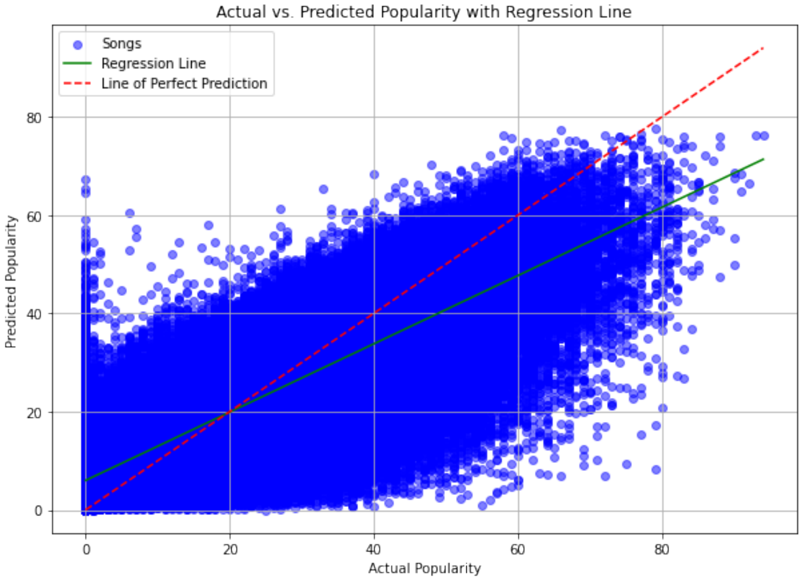
In [73]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error,
mean_squared_error, r2_score

# Initialize the Linear Regression model
from sklearn.ensemble import RandomForestRegressor


# Initialize the Random Forest Regressor model
regressor = RandomForestRegressor(random_state=42)

# Fit the model to the training data
regressor.fit(X_train, y_train)

# Make predictions on the test data
y_pred = regressor.predict(X_test)

# Evaluate the model (for regression, typically using Mean Squared Error)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
# Calculate predictions on the test data
y_pred = regressor.predict(X_test)

# Calculate additional regression metrics
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

# Print the metrics
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")
```

```
Mean Squared Error: 76.63935130884003
Mean Squared Error (MSE): 76.64
Mean Absolute Error (MAE): 6.48
Root Mean Squared Error (RMSE): 8.75
R-squared (R2) Score: 0.70
```

In [76]:

```
from numpy.polynomial.polynomial import Polynomial


# Visualizing Predicted vs. Actual Popularity
plt.figure(figsize=(10, 7))
plt.scatter(y_test, y_pred, color='blue', label='Songs', alpha=0.5)

# Fit a polynomial of degree 1 (i.e., a line) to the data
p = Polynomial.fit(y_test, y_pred, 1)

# Plot the regression line
x = np.linspace(min(y_test), max(y_test), 100)
y = p(x)
plt.plot(x, y, color='green', label='Regression Line')

# Plot the line of perfect prediction
plt.plot([min(y_test), max(y_test)],
         [min(y_test), max(y_test)],
         color='red', linestyle='--', label='Line of Perfect Prediction')  #
Diagonal line for perfect predictions

# Labeling the plot
plt.title("Actual vs. Predicted Popularity with Regression Line")
plt.xlabel("Actual Popularity")
plt.ylabel("Predicted Popularity")
plt.legend()
plt.grid(True)
plt.show()
```


Actual vs. Predicted Popularity with Regression Line
Songs
Regression Line
Line of Perfect Prediction
Predicted Popularity
Actual Popularity