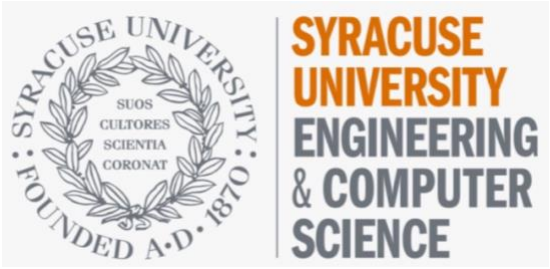


Follower Based Song Popularity Prediction

Report By:

1. Tejaswini Atulchandra Shinde (SUID: 201990091)
2. Sakshi Abhay Patil (SUID: 817700442)
3. Devika Rajesh Shendkar (SUID: 376388943)
4. Akhilesh Santosh Jadhav (SUID: 651126694)
5. Gitika Tirumishi Jada (SUID: 832570194)

Table of Contents

1.Introduction

The "Song Popularity Prediction Using Spotify Dataset" project is an ambitious initiative focused on predicting the popularity of songs on Spotify. Addressing a significant gap in the music industry, it aims to analyze a vast array of features like 'artist_popularity', 'artist_followers', and various song characteristics, including 'danceability', 'energy', and 'acousticness'. This approach is crucial for understanding what resonates with listeners, providing artists, producers, and record labels with actionable insights for effective strategizing.In terms of methodology, the project adopts an innovative approach, utilizing Decision Tree and KNN algorithms. By comparing the results of these algorithms, the team seeks to determine which one offers superior accuracy in predicting song popularity. This technical rigor is essential in ensuring the reliability and relevance of the findings.

The potential industry impact of this project is substantial. Accurately predicting song popularity could revolutionize marketing strategies, unveil emerging trends, and potentially influence the future of music production. By offering a more nuanced understanding of listener preferences and trends, the project stands to make a significant contribution to the evolution of the music industry. Moreover, the project will delve into how the artist's country of origin might influence song popularity, adding a cultural dimension to the analysis. This unique aspect could uncover global trends and regional preferences in music consumption. Additionally, by identifying key indicators of song popularity, the project could pave the way for AI-driven music creation, where insights from data guide the creative process. Ultimately, the project's comprehensive approach promises not only to enhance the music industry's understanding of audience preferences but also to inspire innovative practices in music production and distribution.

2. Problem Statement, Approach and Significance

a. Problem Statement

The goal of the project is to predict song popularity on Spotify using a comprehensive set of features including 'artist_popularity', 'artist_followers', 'danceability', 'energy', 'loudness', 'acousticness', 'instrumentalness', 'liveness', and the artist's country of origin. This aims to offer actionable insights for artists, producers, and record labels for effective release strategies.

b. Approach

The approach for the "Song Popularity Prediction Using Spotify Dataset" project involves a sophisticated data analytics process. Initially, comprehensive data collection is undertaken from Spotify, focusing on a range of metrics like artist popularity, followers, and specific song features. This data is then meticulously cleaned and preprocessed to ensure accuracy and relevance for the analysis.

In the next phase, the project employs machine learning algorithms, specifically Decision Trees and K-Nearest Neighbors (KNN), to model and predict song popularity. These algorithms are chosen for their effectiveness in handling large datasets with multiple variables. The team conducts rigorous model training and testing, along with hyperparameter tuning and cross-validation, to optimize predictive performance. This methodical approach aims to provide a robust and reliable model for predicting song popularity, contributing valuable insights to the music industry.

c.Significance

The significance of the "Song Popularity Prediction Using Spotify Dataset" project is multifaceted and profound. Primarily, it serves as a pivotal tool for artists, producers, and record labels in the music industry, providing them with critical insights to formulate effective marketing and production strategies. This project's predictive analytics capabilities enhance the efficiency of decision-making processes by identifying potential hit songs, thereby optimizing resource allocation and marketing efforts. Moreover, its consistent and accurate predictions offer a reliable foundation for strategic planning.

In terms of scalability, the model's adaptability to diverse datasets signifies its potential application across various music genres and markets. This universal applicability amplifies its utility in global music analytics. Additionally, the project contributes significantly to academic and industry research, offering a novel methodological approach to understanding consumer preferences in the entertainment sector. It stands as a testament to the power of data science in transforming traditional industry practices, paving the way for more data-driven, informed decision-making in the rapidly evolving landscape of music and entertainment.

3. Data

a.Data Gathering

Data collection for the "Song Popularity Prediction Using Spotify Dataset" project is a systematic and crucial procedure, focusing on gathering and analyzing relevant information from Spotify's database. This process is instrumental in acquiring detailed and accurate data, pivotal for making informed decisions, answering key research questions, and evaluating song popularity factors.

This endeavor involves several steps: defining the research objective, identifying essential data sources from Spotify, designing an effective data collection strategy, gathering and structuring the data for analysis, and interpreting the findings. Techniques like API data retrieval and algorithmic analysis are employed to ensure a comprehensive understanding of song popularity dynamics. The goal is to ensure the data is varied, representative, and rich in the features that influence a song's success on Spotify.

Steps -

i. Defining the Research Question: The first step is to define the research question for the project, which in this case is identifying the key features that predict a song's popularity on Spotify.

ii. Selecting Data Sources: The primary data source will be Spotify, utilizing their API to gather detailed information on songs, artists, and user interactions.

iii. Developing a Data Collection Strategy: This involves deciding on the specific data points to collect from Spotify and determining the method of collection, such as direct API calls or using third-party data extraction tools.

iv. Data Collection and Preparation: Post-collection, the data needs to be prepared for analysis. This includes cleaning the data, handling missing values, and structuring it into an analyzable format.

v. Ensuring Data Quality and Diversity: For the algorithm to effectively predict song popularity, it's crucial that the collected data is diverse, representative, and accurately reflects the characteristics that influence popularity on Spotify.

b.Data Preparing

Data preparation is a vital stage in any data analysis project, involving the transformation of raw data into a clean, organized format suitable for analysis. This process includes cleaning data to remove inaccuracies, handling missing values, normalizing data, and categorizing data into structured formats, such as databases or spreadsheets. The goal is to ensure that the dataset is accurate, consistent, and ready for analytical processing.

For the "Song Popularity Prediction Using Spotify Dataset" project, data preparation took specific forms. Firstly, the raw data from Spotify, encompassing various song attributes and artist metrics, was cleansed of any irregularities or missing entries. This was followed by normalizing numerical features like loudness and tempo to ensure consistency across the dataset. Additionally, categorical data, such as the artist's country of origin, were standardized to a uniform format, facilitating smoother analysis. This meticulous preparation was crucial for the accurate modeling and prediction of song popularity on Spotify.

c. Data Preprocessing

Data Cleaning

Data cleaning involved several meticulous steps:

- Features Selection: A thorough examination of the dataset's features was conducted.
    - ID: As each track's ID is unique and doesn't contribute to the model's predictive capability, it was removed.
    - Name: With 132,940 unique values, the 'name' feature was deemed problematic for modeling and subsequently dropped.
    - Artists: Given the significant number of unique values (33,375) and the anticipation that the artist's popularity influences track popularity, this feature was retained for separate handling.
    - Release Date/Year: The 'year' feature, being a derivative of 'release_date', made one of them redundant. 'Release_date' was dropped due to inconsistency in its data (full dates and year-only entries).
- Duplicates & Nulls:
    - Prior to the removal of the aforementioned features, no duplicates existed in the dataset, attributed to the unique ID for each track.
    - Post-removal, 669 duplicates were identified and resolved by retaining only the first copy of each duplicate for the model.
    - The dataset was found to be free of null values, ensuring completeness and consistency for the analysis.

These steps ensured the dataset's cleanliness and relevance, setting a strong foundation for accurate model development and song popularity prediction.

Exploratory Data Analysis (EDA)

The Exploratory Data Analysis (EDA) was conducted with a focus on both numeric and categorical features:

- Numeric Features Analysis: The numeric columns were isolated and their correlations examined. A heatmap was created to visualize these correlations, particularly focusing on their relationship with song popularity. The features most correlated with popularity were identified and highlighted.
- Categorical Features Analysis: The 'artists' feature, with its high uniqueness, was analyzed for its influence on song popularity. Bar plots were used to showcase the most popular artists and those with more than 100 tracks. A targeted encoding approach was used for the 'artists' feature, replacing artist names with derivatives of their popularity. This involved analyzing the distribution of artist appearances and their average popularity.
- Special Feature Analysis: Certain features like 'explicit' and 'key' were analyzed for their impact on popularity. For 'explicit', a bar plot showed the distribution of explicit content and its correlation with popularity. For 'key', the distribution across records and its relation to popularity was explored using bar plots.
- Detailed Examination of Numeric Features: Features like 'acousticness', 'danceability', 'duration_ms', 'energy', 'instrumentalness', 'liveness', 'loudness', and 'speechiness' were analyzed for their distribution and impact on song popularity, using a variety of statistical plots like scatter plots and distribution plots.

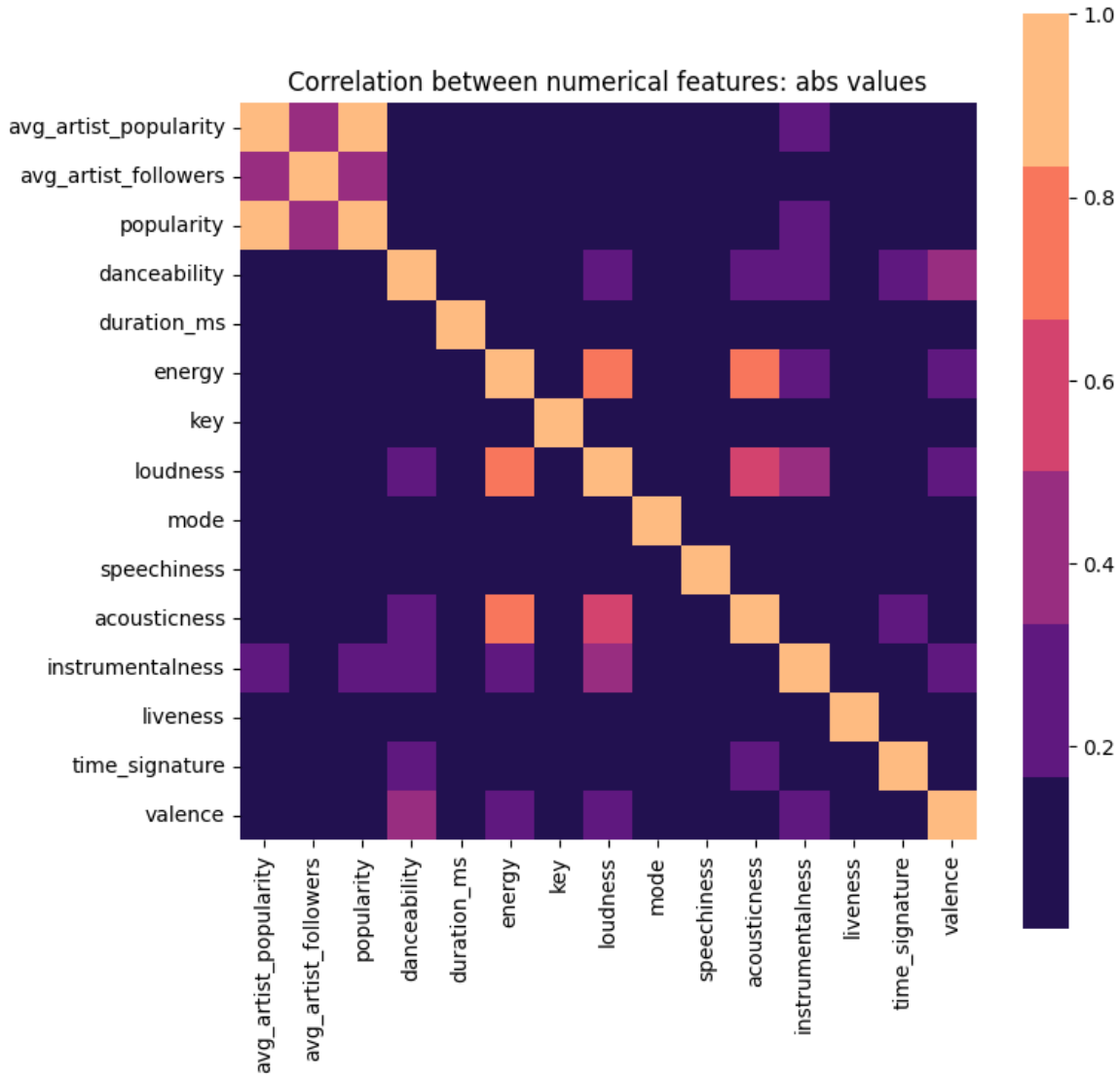
Each of these steps provided deep insights into how different features influenced the popularity of songs on Spotify, guiding the subsequent model building phase.

4. Data Visualization

We employ various data visualization techniques to present data using visual elements such as charts, graphs, and maps. This approach aims to simplify understanding, emphasize patterns, trends, and exceptional data points within extensive datasets. The primary goal of our data visualization efforts is to communicate complex data relationships and insights in a clear and accessible manner. These techniques encompass a range of methods, including area charts, scatter plots, pie charts, line graphs, histograms, pivot tables, box plots, and radar charts. We have utilized four distinct visualization methods in our work.

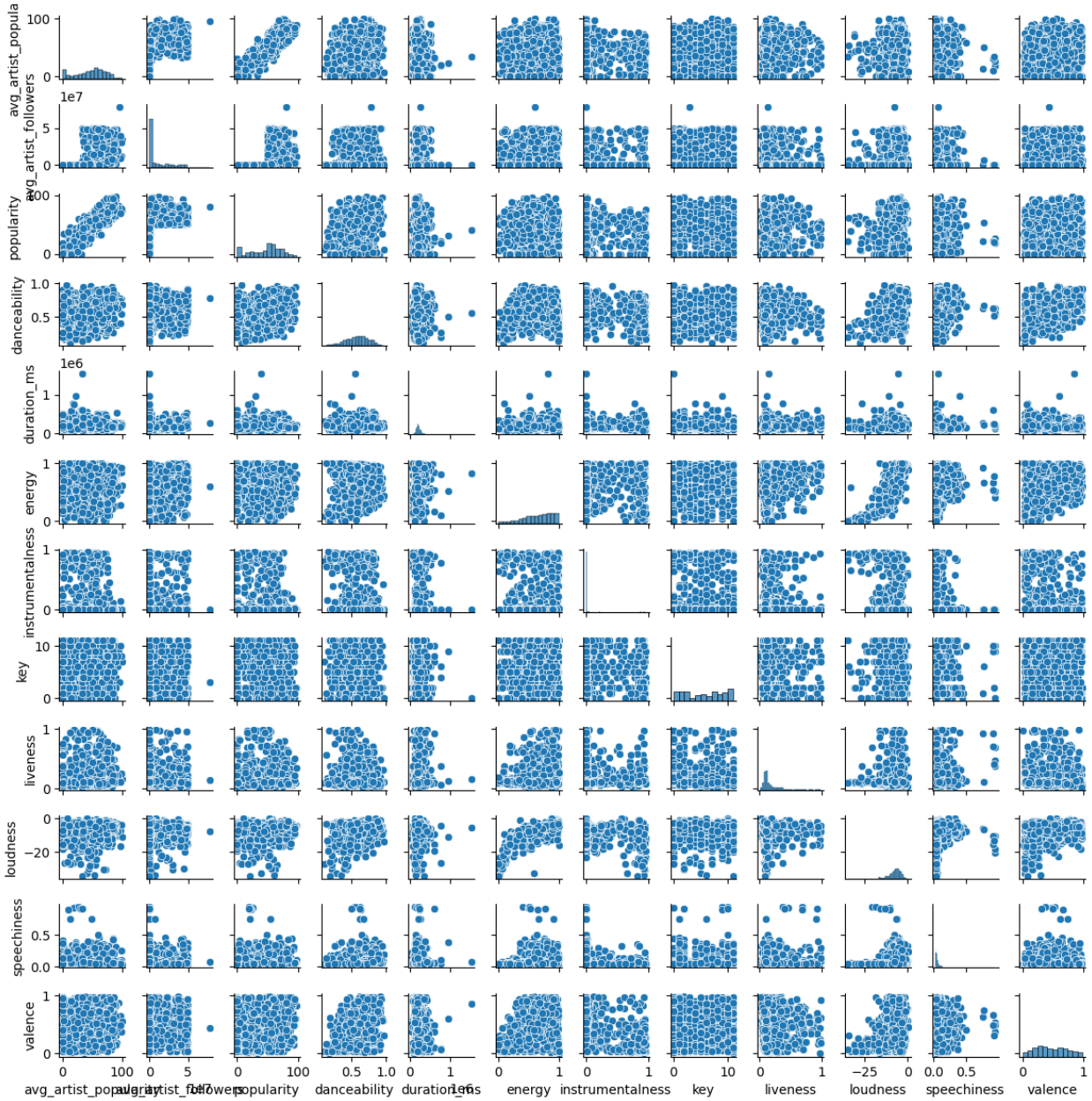
- ## Heatmaps:

A heatmap is created to visualize correlations between numeric features. This helps in understanding the relationships and dependencies among various numerical variables.



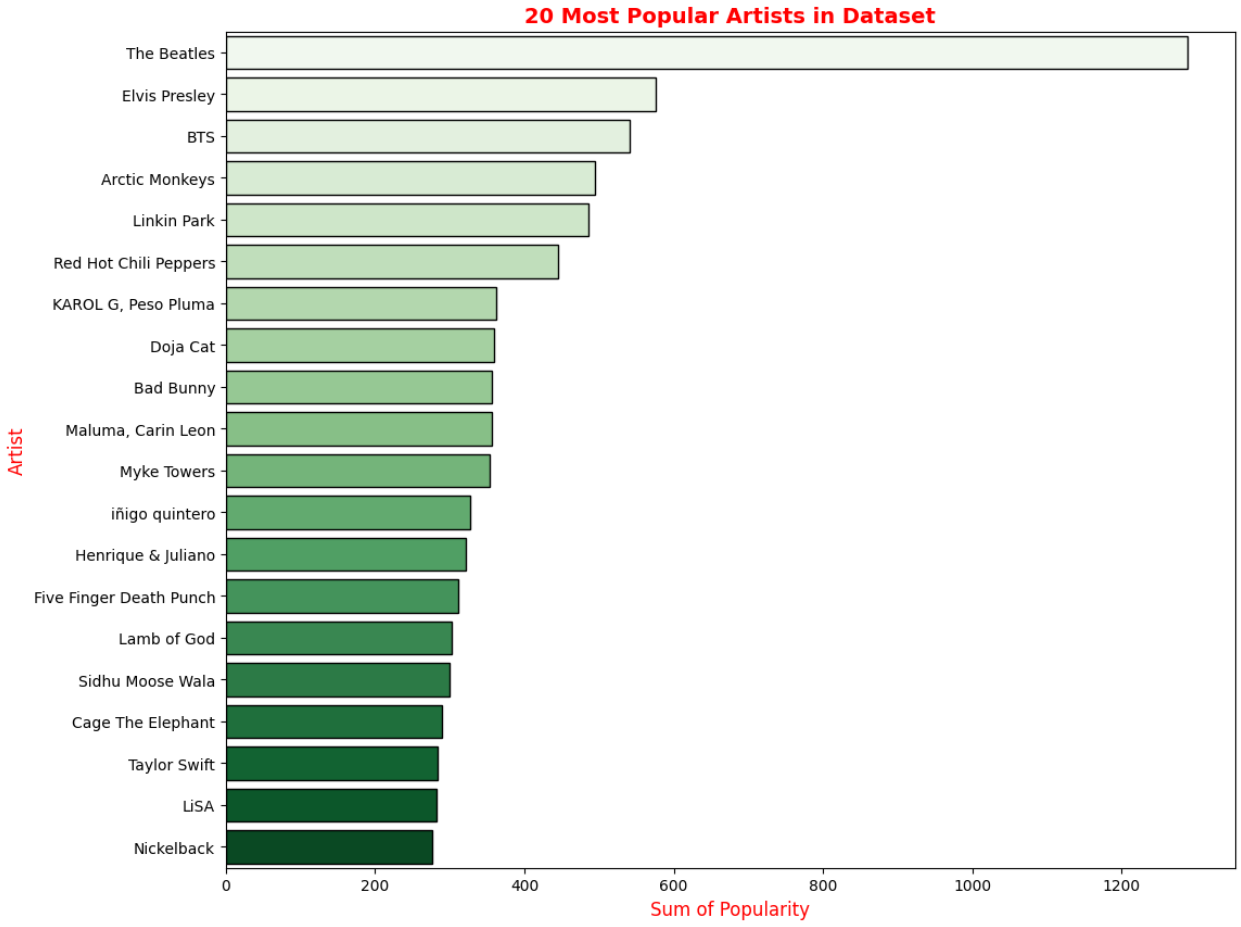
- Pairplot of numerical Features:

A pairplot (scatterplot matrix) is used to visualize the relationships between different numerical variables in the dataset. It provides a quick overview of how each variable interacts with the others.



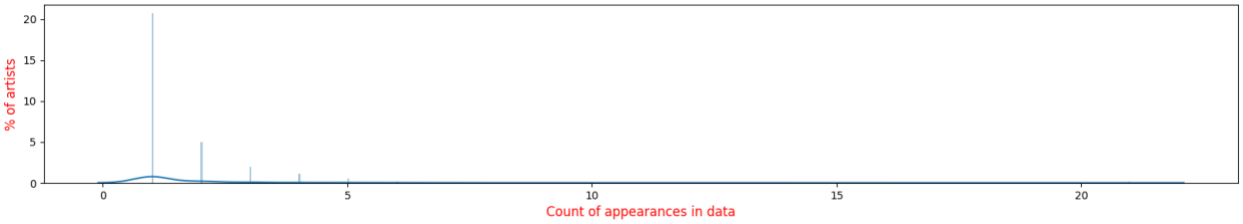
- ## Popularity of Artists:

This plot visualizes the 20 most popular artists based on the sum of their song popularity. It helps in identifying the top artists in the dataset.



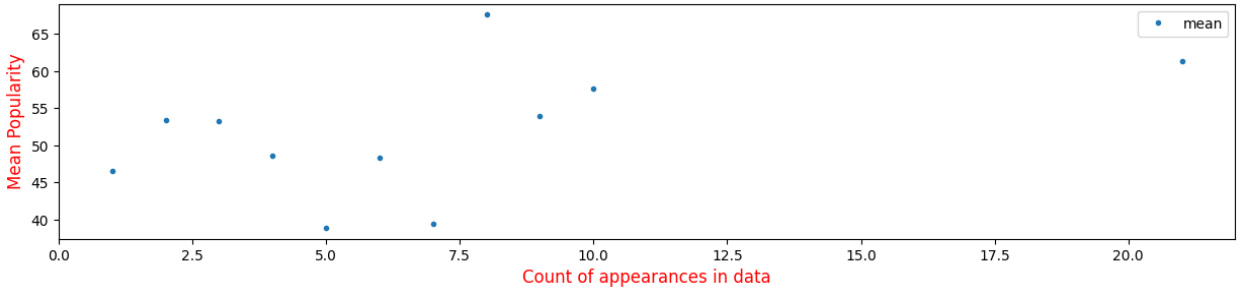
- ## Mean and Count of Song Popularity:

This involves adding new columns for mean and count of song popularity for each artist, which can be useful for further analysis.



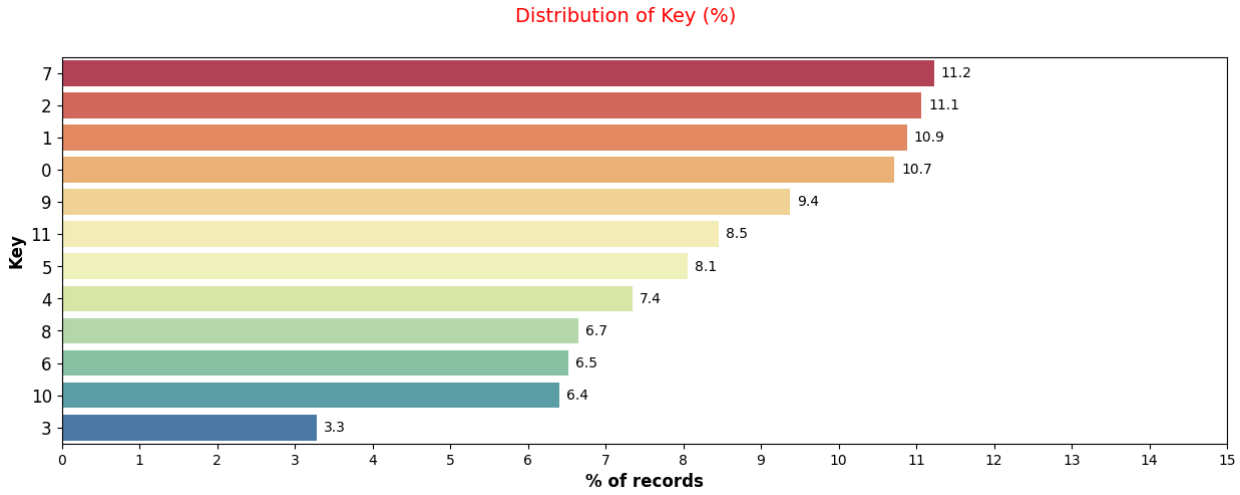
- ## Relationship Between Song Appearances and Popularity:

A plot showing the relationship between the number of song appearances and mean popularity for artists. This can reveal trends about how the frequency of songs by an artist relates to their popularity.



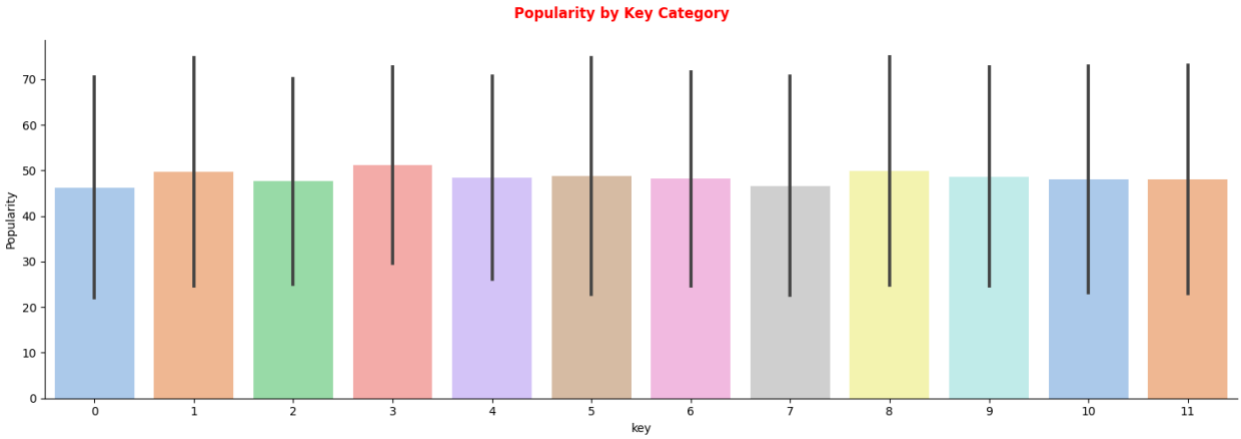
- ## Distribution of Musical Key:

This plot presents the distribution of musical keys in the dataset as a percentage, offering insights into the most common keys used in the songs.



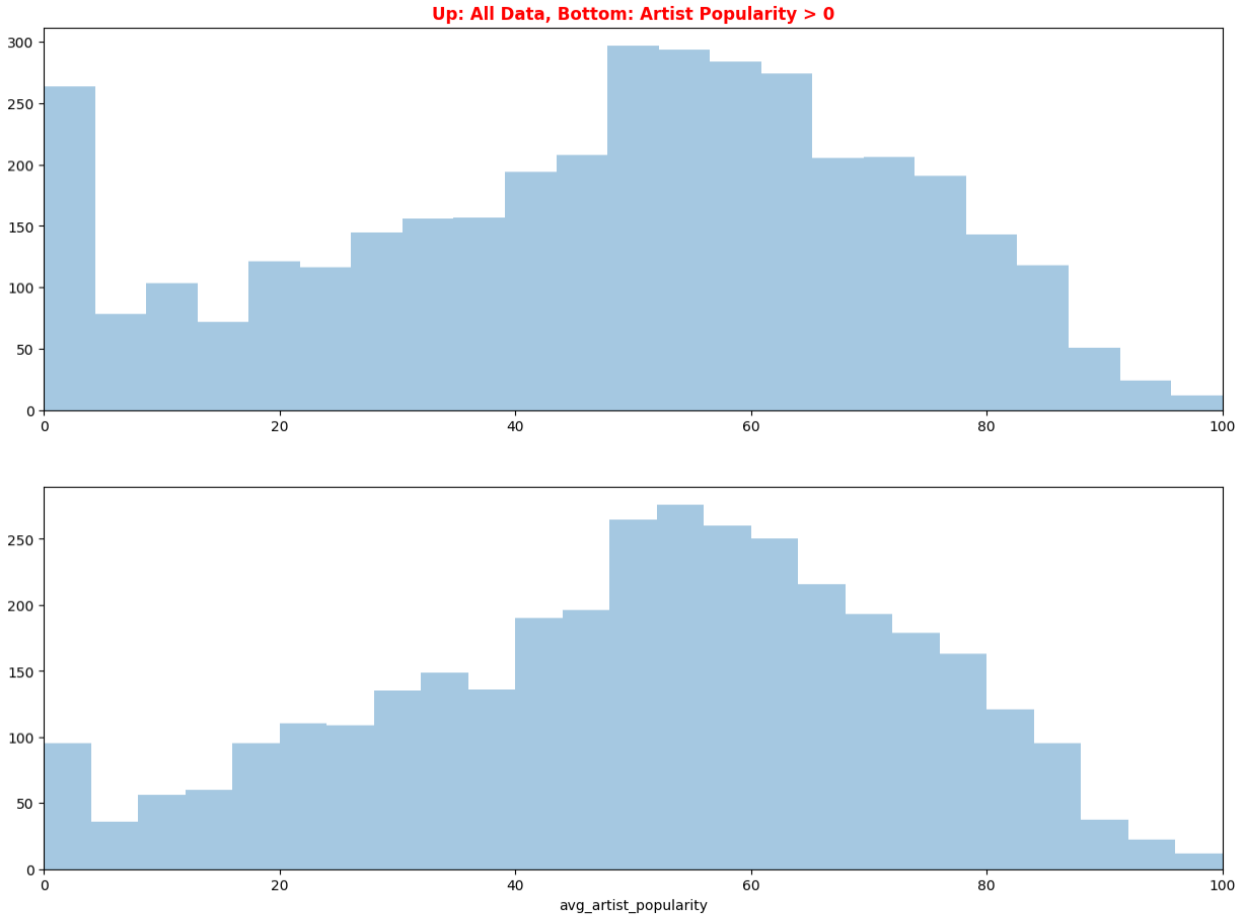
- ## Average Popularity by Musical Key:

A bar plot shows the average popularity of songs in each musical key, highlighting which keys are associated with more popular songs.



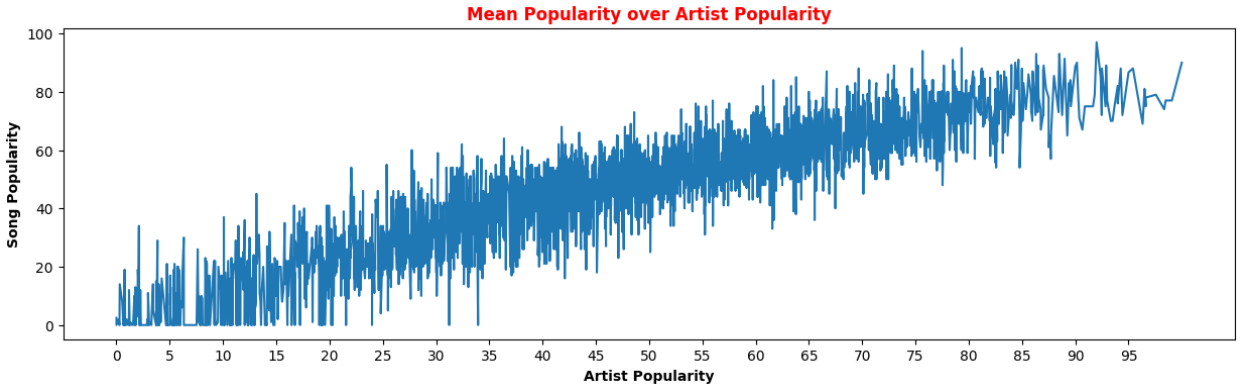
- ## Comparing Distributions of Artist Popularity:

Two subplots are used to compare distributions of average artist popularity, likely to understand the range and spread of this metric.



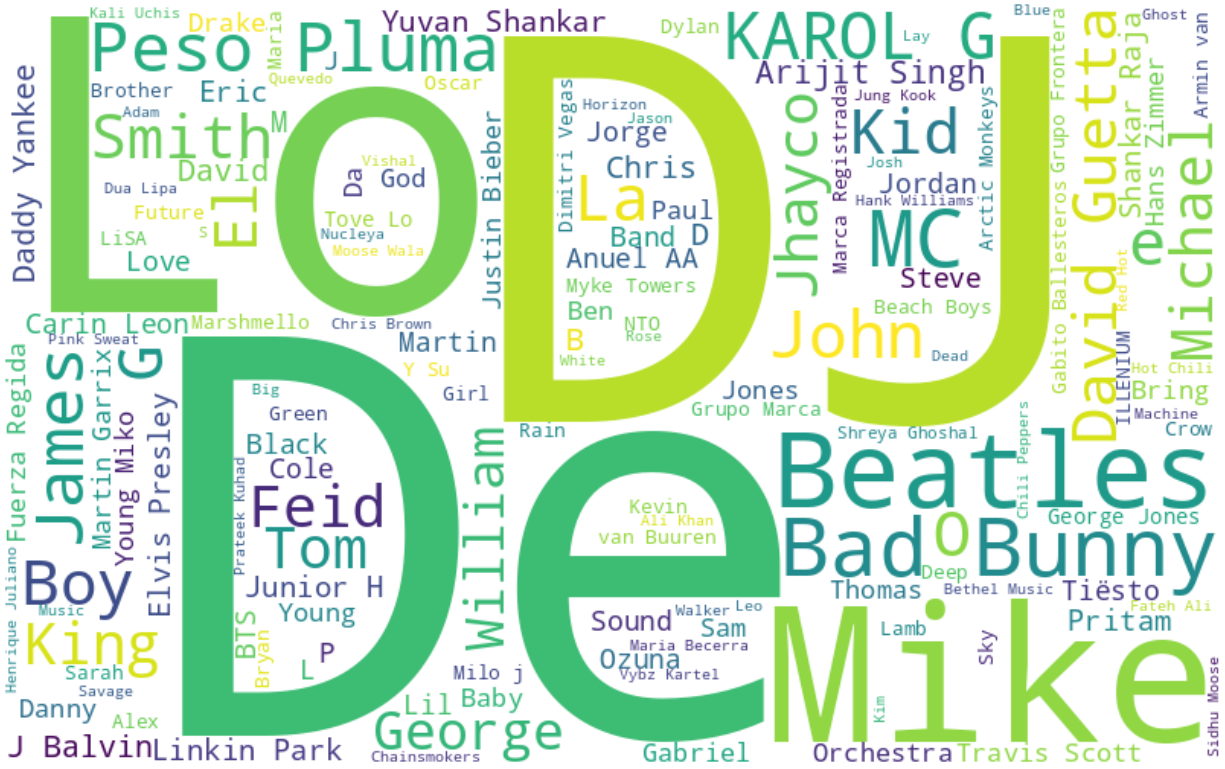
- Song Popularity vs. Artist Popularity:

This plot examines the relationship between mean song popularity and average artist popularity, potentially revealing whether popular artists tend to produce more popular songs.



- Word Cloud

A word cloud, also known as a tag cloud or text cloud, is a visual representation of text data. It is typically used to depict keyword metadata (tags) on websites, or to visualize free form text.

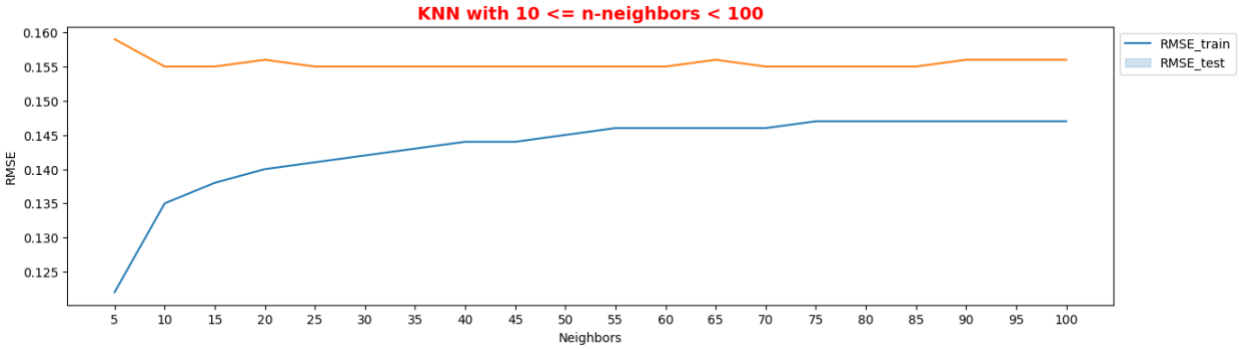


5. Technologies Used

- Python: The primary programming language for your project.
- Google Colab: The primary environment for running your Python code and Jupyter Notebooks.
- ndas: Used for data manipulation and analysis. It provides data structures like DataFrames that are essential for data handling.
- NumPy: A library for numerical operations in Python. It is often used for working with arrays and matrices.
- Matplotlib: Used for creating various types of plots and visualizations.
- Seaborn: A data visualization library built on top of Matplotlib, which provides a high-level interface for creating informative and attractive statistical graphics.
- Plotnine: A library for creating complex and customizable plots using a syntax similar to ggplot2 in R.
- Pydot: Used for working with DOT language graph descriptions.
- Tqdm: A library for displaying progress bars during lengthy computations or loops.
- SciPy: Used for scientific and technical computing. You import curve_fit from SciPy for curve fitting.
- scikit-learn (sklearn): A machine learning library in Python. You import various modules, including:
    - LinearRegression: For linear regression modeling.
    - KNeighborsRegressor: For k-nearest neighbors regression.
    - pairwise_distances: For computing pairwise distances between data points.
    - DecisionTreeRegressor and export_graphviz: For decision tree regression modeling and visualization.
    - mean_squared_error: For calculating mean squared error.
    - train_test_split: For splitting data into training and testing sets.

6. Proposed Models

- **KNN**



The code is for implementation of the K-Nearest Neighbors (KNN) algorithm for regression, specifically to predict a continuous variable (like song popularity). The code focuses on finding the optimal number of neighbors (K) for the model and analyzing its performance through Root Mean Squared Error (RMSE). It's divided into several key parts:

Part 1: Evaluating KNN for Different Values of Neighbors (5 to 100)

Iterative Training and Testing:

- The KNN regressor is trained for different values of neighbors (from 5 to 100, in steps of 5).
- For each value of K, the model is trained on the training set and predictions are made for both the training and testing sets.

Calculating RMSE:

- Root Mean Squared Error (RMSE) is calculated for both training and testing predictions. RMSE is a standard way to measure the error of a model in predicting quantitative data.

Storing RMSE Values:

- RMSE values for training and testing sets are stored in RMSE1_train and RMSE1_test lists, respectively.

Plotting RMSE Values:

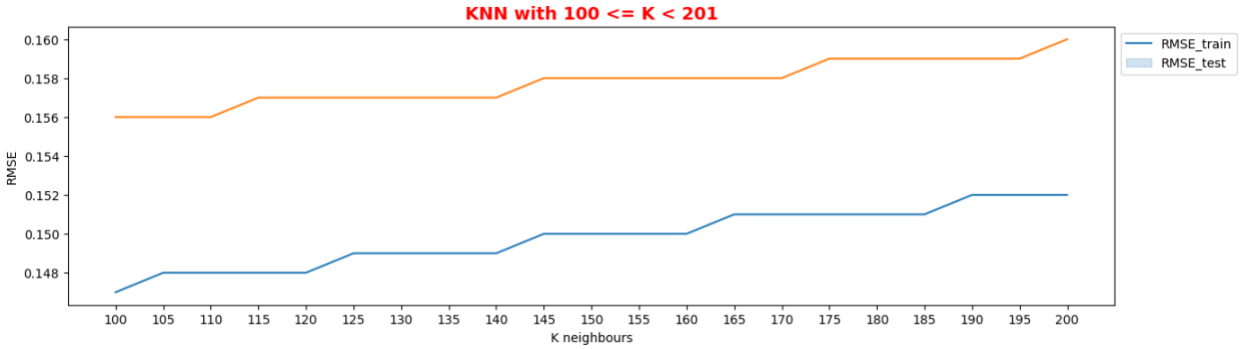
- A line plot is created to visualize how RMSE changes with different values of K. This helps in understanding the trade-off between bias and variance in the model.

Identifying Optimal Gap:

- The smallest gap between training and testing RMSE is identified. This point is considered optimal as it suggests a good balance between underfitting and overfitting.

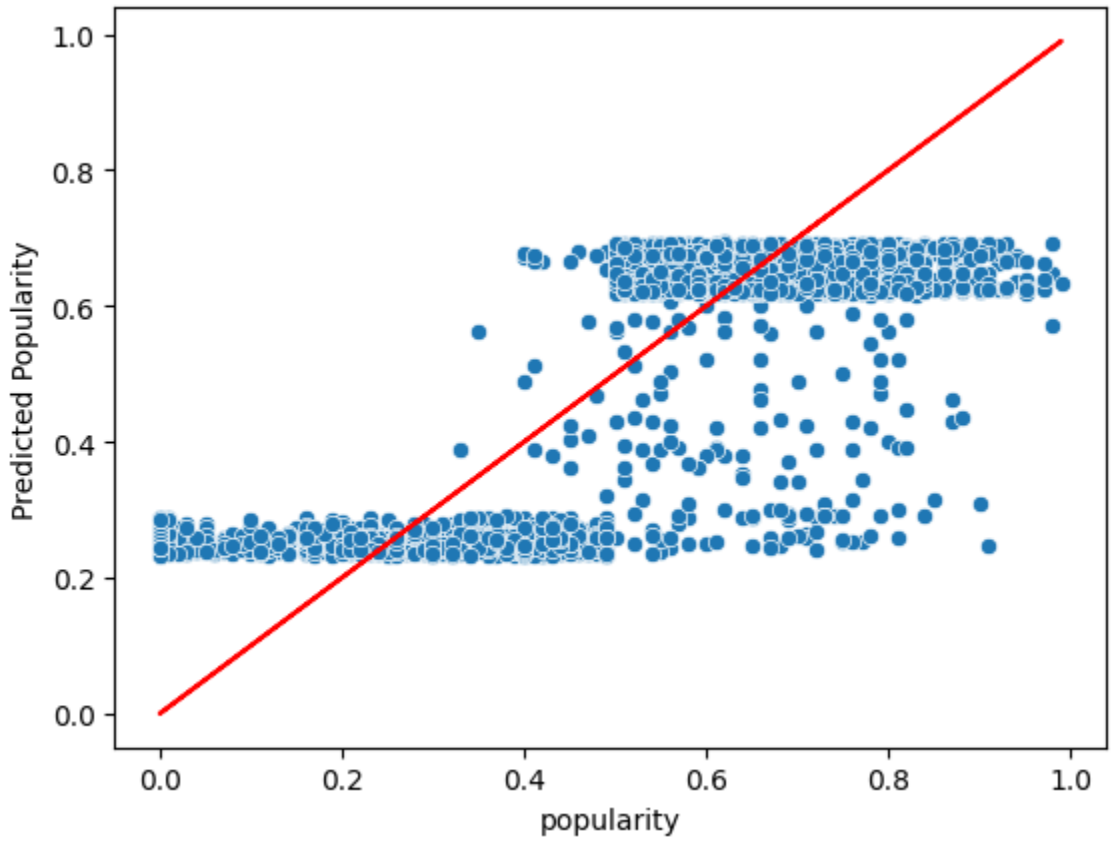
Part 2: Evaluating KNN for Higher Values of Neighbors (100 to 200)

This part repeats the same process as Part 1 but for a higher range of neighbors (from 100 to 200). It's done to check if a larger number of neighbors improves the model.



Part 3: Visualizing Predictions for Training Data

- A scatter plot is created to visualize the actual vs. predicted values of popularity on the training data. The line y = x (red line) represents the perfect prediction. The closer the scatter points are to this line, the better the predictions.



Key Takeaways:

- RMSE Analysis: Lower RMSE values indicate better model performance. By comparing RMSE across different K values, the optimal number of neighbors can be identified.
- Overfitting vs. Underfitting: A significant gap between training and testing RMSE often indicates overfitting (model performs well on training data but poorly on unseen data). The code aims to find a balance.
- Model Complexity: Higher values of K generally lead to a simpler model. This exploration helps in understanding how increasing K affects model performance.
- Visualization: Plots provide a clear visual understanding of how the model's performance changes with K and how well the model is predicting.

● Decision Trees

The code covers several aspects of evaluating and interpreting a Decision Tree Regressor model. Following is the break-down of the whole process,

1. Single Run Evaluation with Specific Hyperparameters

Model Initialization:

A Decision Tree Regressor is initialized with max_leaf_nodes=41 and min_samples_split=2000. These are hyperparameters that control the tree's complexity.

Model Fitting:

The model is trained (fit) on X_train and y_train datasets.

Prediction and Clipping: Predictions are made on both the training (X_train) and test (X_test) datasets. The .clip(0, 1) function ensures that predicted values are within the range [0, 1].

Root Mean Square Error (RMSE) Calculation*:*

RMSE is calculated for both training and test predictions, providing a measure of the model's prediction error.

Printing RMSE:

RMSE values for training and test sets are printed, offering an immediate evaluation of model performance.

2. Evaluating Decision Tree with Varying Max Leaf Nodes

Iterative Evaluation:

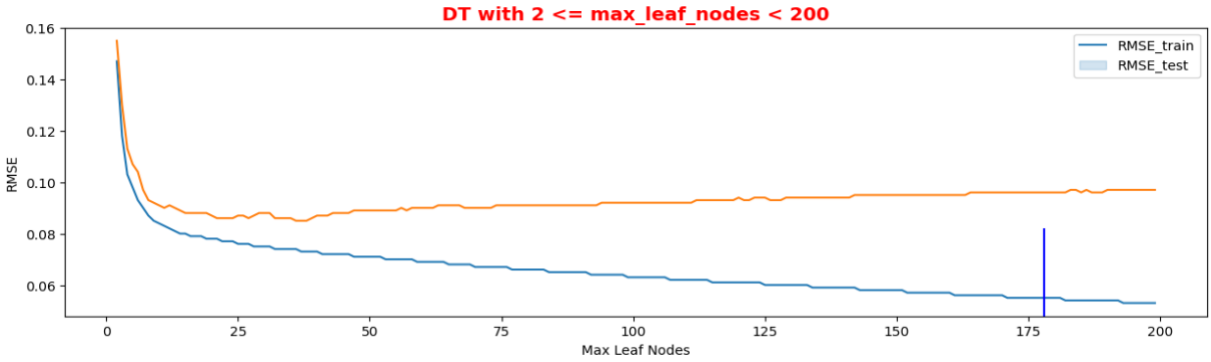
A loop runs from 2 to 199 (range(2, 200)), each time initializing and fitting a Decision Tree model with a different number of maximum leaf nodes.

RMSE Calculation in Loop:
For each model, RMSE for training and test sets is calculated and stored in RMSE3_train and RMSE3_test lists, respectively.

Predictions with Clipping:
Predictions are clipped between 0 and 1, as in the first part.

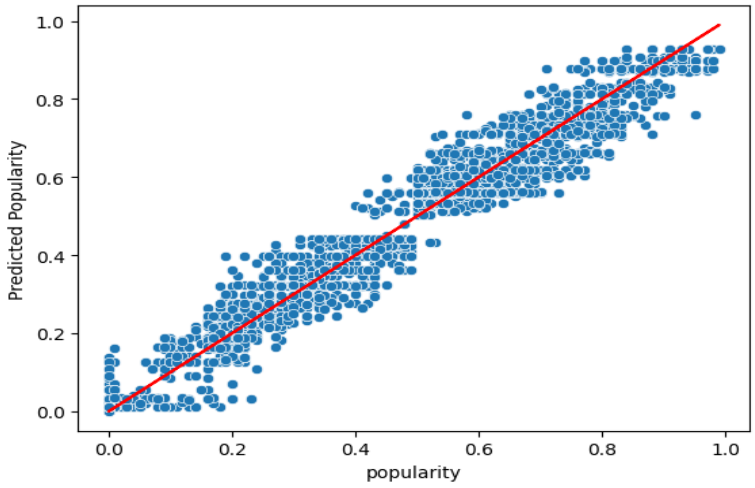
3. Plotting RMSE for Varying Max Leaf Nodes

Plot Initialization:
A plot is created with matplotlib and seaborn to visualize RMSE values.
RMSE Visualization: Two line plots are drawn for RMSE values against the number of max leaf nodes for both training and test datasets.
Additional Plot Features: A vertical line at x=178, a legend, and axis labels and titles are added for better interpretability.



4. Scatter Plot of Actual vs. Predicted Popularity

Scatter Plot Creation: A scatter plot is made using seaborn, showing the relationship between actual and predicted popularity on the training set.
Reference Line: A red diagonal line is added as a reference, indicating where actual and predicted values would be equal.



5. Calculating and Identifying Validation Gaps

Gap Calculation: The ratio of RMSE for training to test sets is calculated for each max leaf node setting. This ratio, termed as 'validation gap', indicates overfitting (the larger the gap, the more overfitting).

Minimum and Maximum Gaps: The minimum and maximum validation gaps are identified, along with their corresponding indices (max leaf nodes).

```
Minimum validation gap is: 54.639% in index number 191
Maximum validation gap is: 94.839% in index number 0
```

6. Printing Feature Importance

Feature Importance Display: For the Decision Tree model with the last set of hyperparameters used in the loop, the importance of each feature in X_train is printed. Feature importance helps in understanding which features are most influential in predicting the target variable.

In summary, this code performs a comprehensive evaluation of a Decision Tree Regressor model, exploring how different settings of max_leaf_nodes affect performance (measured by RMSE), visualizing model predictions against actual values, and examining the gap between training and test performance to assess overfitting. Additionally, it highlights the importance of each feature used in the model.

7. Future Work

- Enhanced Predictive Models: As machine learning and data analytics evolve, you can incorporate more sophisticated algorithms or deep learning techniques to improve the accuracy of your predictions.
- Real-Time Analysis: Future developments could include real-time popularity prediction, allowing artists and producers to adjust their strategies promptly based on immediate feedback from listeners.
- Broader Data Integration: Incorporating additional data sources such as social media sentiment analysis, YouTube views, or concert ticket sales could provide a more holistic view of a song's popularity.
- Personalized Recommendations: The methodology could be adapted to develop personalized song recommendation systems for users, enhancing user experience on streaming platforms.
- Industry Trend Analysis: Long-term data collection and analysis could reveal significant trends in music preferences, helping the industry to anticipate and adapt to changing tastes.

8. Conclusion

Our project has highlighted the synergy between machine learning and music analytics, offering a glimpse into the future of predictive models in the music industry. We're excited about the possibilities that lie ahead as we continue to refine our approach and drive innovation.

9. References

1. “SpotHitPy: A Study For ML-Based Song Hit Prediction Using Spotify” by Ioannis Dimolitsas1, Spyridon Kantarelis2and Afroditi Fouka3
2. Revisiting the Problem of Audio-based Hit Song Prediction using Convolutional Neural Networks
3. A. H. Raza and K. Nanath, “Predicting a hit song with machine learning: Is there an apriori secret formula?” in 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA), 2020, pp. 111–116.
4. E. Zangerle, M. Vötter, R. Huber, and Y.-H. Yang, “Hit song prediction: Leveraging low- and high-level audio features,” in ISMIR, 2019.
5. E. Georgieva, M. Şuta, and N. S. Burton, “Hitpredict : Predicting hit songs using spotify data stanford computer science 229 : Machine learning,” 2018.
6. K. Middlebrook and K. Sheik, “Song hit prediction: Predicting billboard hits using spotify data,” ArXiv, vol. abs/1908.08609, 2019.
7. T. K. Ho, “Random decision forests,” in Proceedings of 3rd international conference on document analysis and recognition, vol. 1. IEEE, 1995, pp. 278–282.
8. C. Cortes and V. Vapnik, “Support-vector networks,” Machine learning, vol. 20, no. 3, pp. 273–297, 1995.