ECON-UB-232 - Data Bootcamp - Prof. Koehler

Anmar Nasser - Final Project

Dec 14, 2024

Predictive Analytics in Music Streaming: Analysis of Spotify Data

Table of Contents

1 Introduction

1.1 Overview

As someone who uses Spotify daily, I've always been curious about what makes certain songs more popular than others. I decided to dive into Spotify's track data to see if I could uncover any patterns and maybe even predict which songs might become hits.

In this project, I analyzed thousands of Spotify tracks, looking at features like how danceable a song is, its energy level, and even its "valence" (basically how happy or sad it sounds). I wanted to see if these technical aspects could help predict a song's popularity.

1.2 Project Objectives

My main goals were pretty straightforward:

- Figure out what makes some songs more popular than others
- Try to predict song popularity using machine learning
- See if I could sort songs into genres just based on their audio features
- Maybe understand why Spotify recommends the songs it does

1.3 Dataset Description

The analysis utilizes multiple interrelated datasets:

- Main Dataset (df_all): Contains 170,653 songs with 19 features including audio characteristics and metadata
- Genre Dataset (df_w_genres): 28,680 entries with genre information
- Artist Dataset (df_by_artist): 28,680 entries aggregated by artist
- Year Dataset (df_by_year): 100 entries with yearly averages
- Genre Averages (df_by_genres): 2,973 entries with genre-specific averages
- Key features analyzed include:
- Audio features (valence, acousticness, danceability, energy, etc.)
- Metadata (artist, year, popularity)
- Technical attributes (duration_ms, key, mode)
- Genre information

I thought this would be an interesting way to combine my daily music listening habits with some data analysis.

2 Data Preprocessing and Exploration

2.1 Initial Data Assessment

This dataset does not come with an accompanying codebook that describes each column, so we must familiarize ourselves with the data first and understand what each column represents for the different data frames. After checking the shape of the data:

- Main dataset (df_all) contains 170,653 entries with 19 features
- Genre dataset (df_w_genres) has 28,680 entries with 16 features
- Artist dataset (df_by_artist) contains 28,680 entries with 15 features
- Year dataset (df_by_year) has 100 entries covering years 1921-2020
- Genre averages dataset (df_by_genres) contains 2,973 entries representing different genres

Notice that df_w_genres and df_by_artist have the same number of rows, with df_w_genres having one more column. df_all has the most rows, so let's assume that this dataframe contains each song individually, while the other dataframes are aggregated datasets based on df_all. df_by_year contains the average song characteristics by year for the past 100 years, and df_by_genres contains the average song characteristics per genre. Because this dataframe has 2973 rows, let's assume that there are 2973 different genres in this dataset.

After analyzing further it seems like the values are a mix of full date strings and strings with just the year. Because there is already a year column, there may be no need for this column, as it is unlikely the exact day that a song is released is a useful feature when recommending a song. Let's remove it and keep exploring.

To sum up briefly, a lot of the variables have values ranging from 0 to 1. Some are because they are boolean encodings, such as explicit and mode, while others, such as energy, instrumentalness, liveness, etc., have values that are continuous from 0 to 1. There are 12 different values for key, ranging from 0 to 11 which each represent the 12 different keys in western music theory. The range of values for loudness, and popularity also seem to be bounded, but the ranges differ from the other features, so some normalization will probably have to be done to get them into the range of 0 to 1. Some slightly concerning things that might need to be explored further is the fact that loudness ranges from very negative, -60, to slightly positive, 3.855. The quartiles show that most of the songs have negative values for loudness, so there must be a reason why some songs have positive values. Another thing to explore is that fact that minimum value for tempo is 0, because, to my knowledge, no song should have a tempo of 0.

2.2 Data Cleaning

The dataframe contains a subset of columns from df_all and df_by_artist, with two additional columns: 'genres' and 'count'. The meaning of the 'count' column presents an

interesting analytical challenge. Initially, I considered two possibilities: either it represents the number of times songs appear in user playlists, or it indicates the total number of songs by each artist in the df_all dataframe. However, the values in this column (reaching up to 226,497) seem implausibly high for playlist counts, and they don't match the artist song counts in df_all. This inconsistency suggests the 'count' column might serve another purpose that requires further investigation

NOTE: I'm aware that the value includes redundant counts of an artists songs, but even without that the value is still too large to represent that number of songs in a user's playlist

Because we cannot confirm the meaning of the count column, we will ignore it.

2.2.1 Handling Missing Values

- No null values found in any of the dataframes
- 9,857 entries had empty genre arrays that needed handling
- Release_date column removed due to inconsistent formatting (mix of full dates and years)

2.2.2 Genre Processing

- Genres stored as string representations of lists needed conversion to actual list objects
- Used ast.literal_eval() to convert string representations to Python lists
- Some genres had very few songs (as low as 1 song per genre)

2.2.3 Artist Data Formatting

- Artist names stored as string representations of lists
- Multiple artists possible for single songs
- Created num_artists column showing number of artists per track
- Total artist count different from expected sum, indicating potential data inconsistencies

2.3 Exploratory Data Analysis

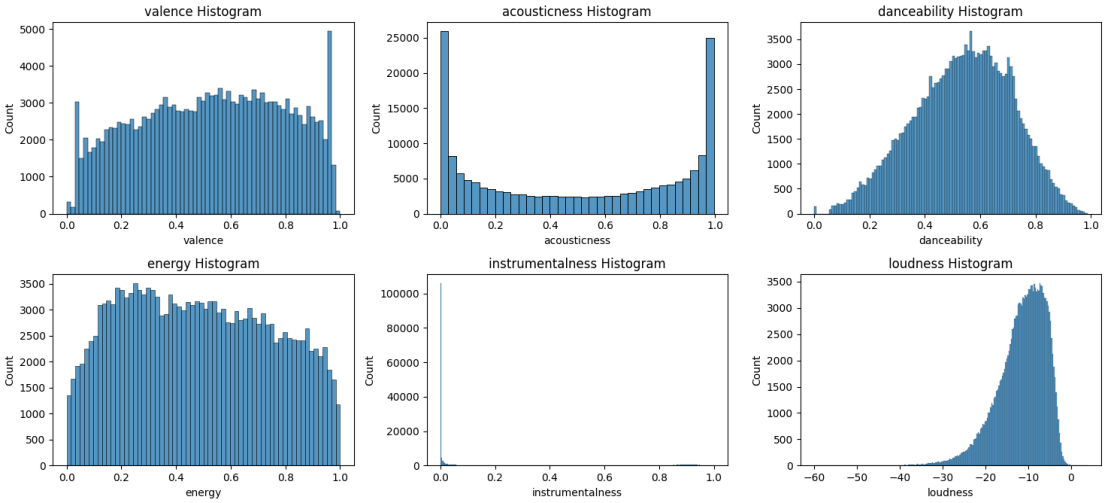
Looking at all the possible variables to use and explore the distributions of their values.

2.3.1 Univariate Analysis

Key findings for main features:

- Valence: Slight negative skew (-0.107), indicating balanced emotional content
- Acousticness: Symmetrical distribution with high concentration at extremes
- Danceability: Negative skew (-0.22), favoring higher danceability values

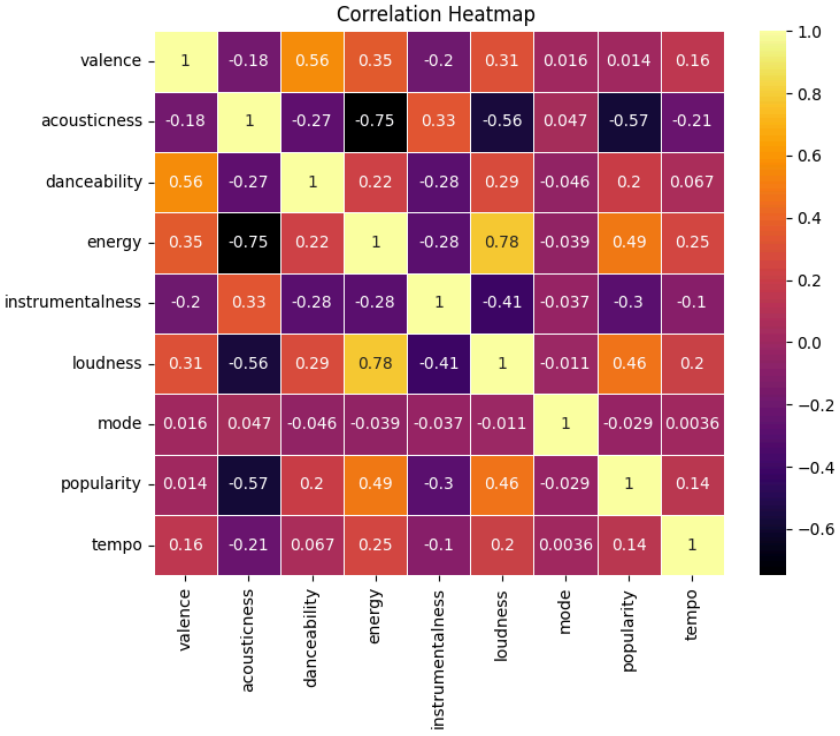
- Energy: Slight right skew (0.112), tendency toward lower energy
- Instrumentalness: Strong right skew (1.63), most songs cluster at lowest values
- Loudness: Negative skew (-1.05), most songs in negative decibel range



2.3.2 Feature Correlations

Strong correlations identified:

- Valence-Danceability: 0.559 positive correlation
- Acousticness-Energy: -0.749 negative correlation
- Acousticness-Loudness: -0.562 negative correlation
- Acousticness-Popularity: -0.573 negative correlation
- Energy-Loudness: 0.782 positive correlation
- Energy-Popularity: 0.485 positive correlation
- Instrumentalness-Loudness: -0.409 negative correlation
- Loudness-Popularity: 0.457 positive correlation



2.3.3 Feature Outliers

Using Interquartile Method of finding outliers:

Valence:

- No outliers detected
- Values show balanced distribution

Acousticness:

- No outliers detected
- High concentration at extremes (very acoustic or not acoustic)

Danceability:

- 143 outliers identified
- All outliers have danceability value of 0
- Will bin with low danceability songs rather than remove

Loudness:

- 3,501 outliers identified
- All songs below -25 classified as outliers
- Will keep and classify as "very low" loudness

Speechiness:

- Significant outliers present
- Highest value from a podcast episode
- No clear cutoff between songs and spoken content
- Decision: keep outliers due to ambiguity

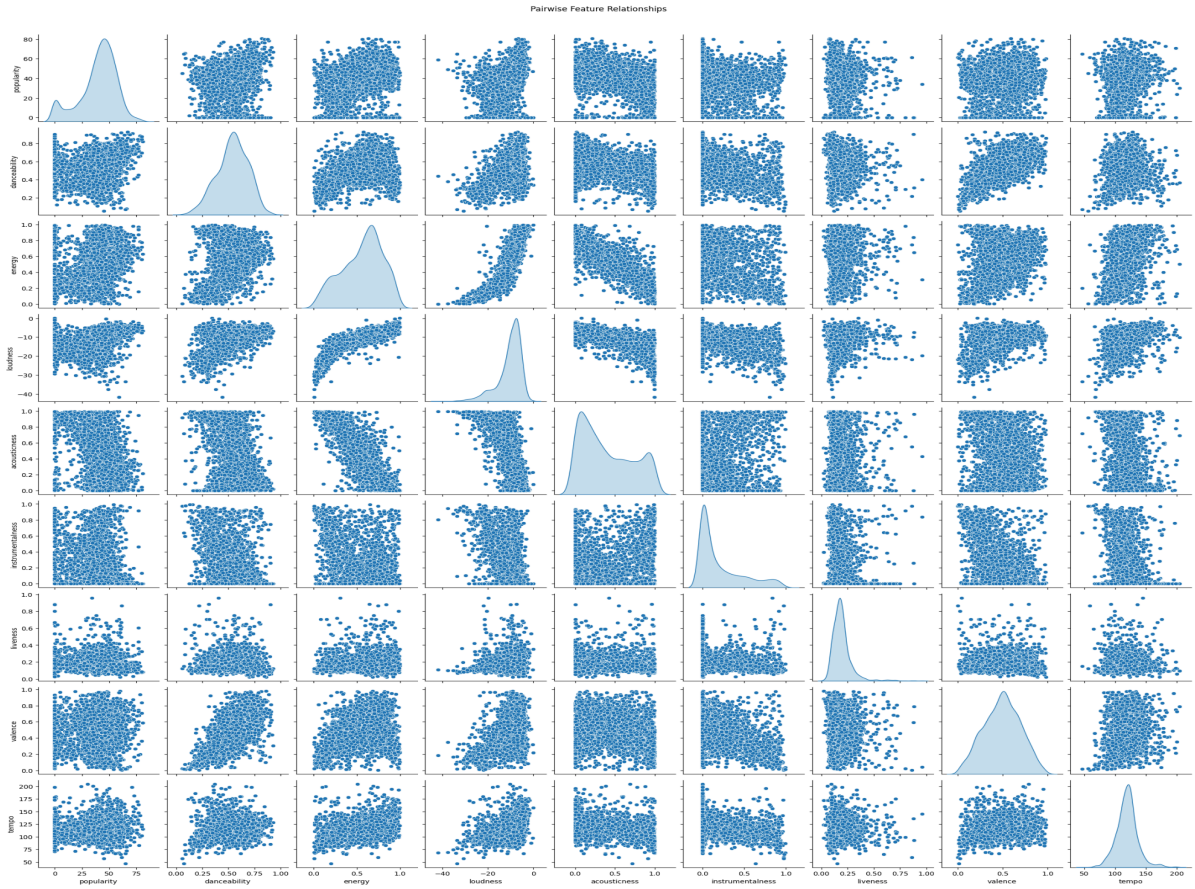
Instrumentalness:

- Most songs cluster at lowest values
- Shows bimodal distribution
- Will bin as low/medium/high

Liveness:

- Feature ignored in analysis
- Not relevant for popularity prediction

Although its not that significant to the analysis, here is a pairwise correlation for each feature:



3 Model Development

In this section, I will do a deep dive into the different models I used to explore this dataset. Many of the techniques I used are important to feature engineering which is essentially when pairs of features that are highly correlated to identify potential interactions and then introducing these new combined features and scaling them for better performance in machine learning models.

3.1 Clustering Analysis

The clustering analysis aimed to identify natural groupings of songs based on their audio features.

3.1.1 KMeans Implementation

Steps taken:

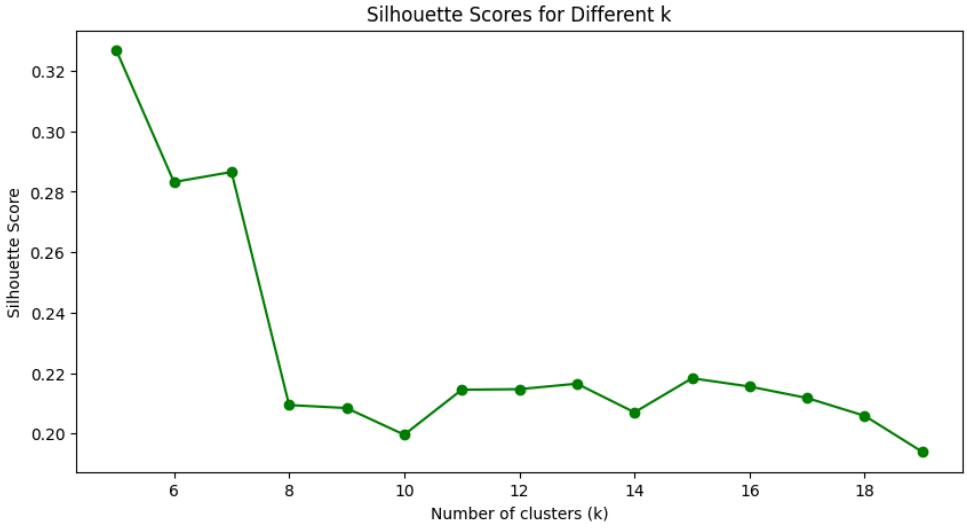
- Implemented KMeans clustering with k values ranging from 5 to 15

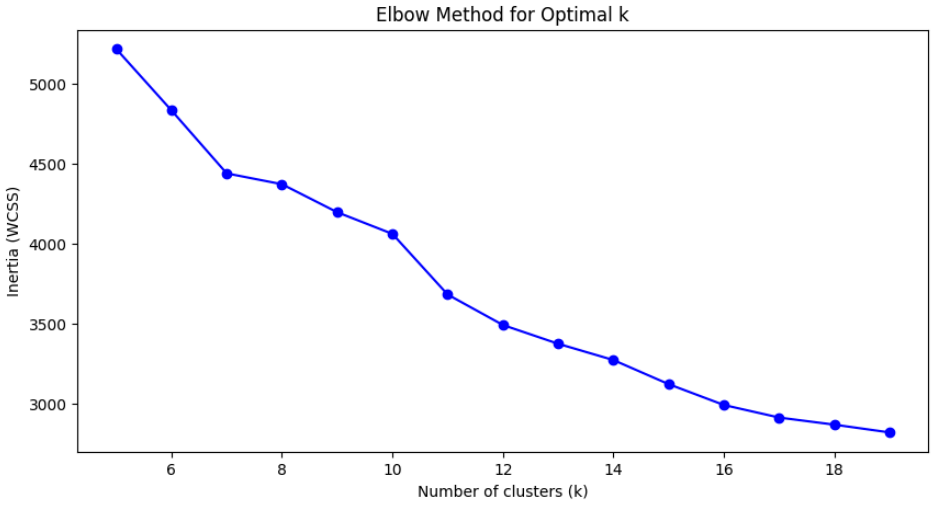
- Used MiniBatchKMeans for computational efficiency
- Features were normalized using MinMaxScaler before clustering
- Each cluster represents a distinct song profile based on audio characteristics

3.1.2 Cluster Evaluation

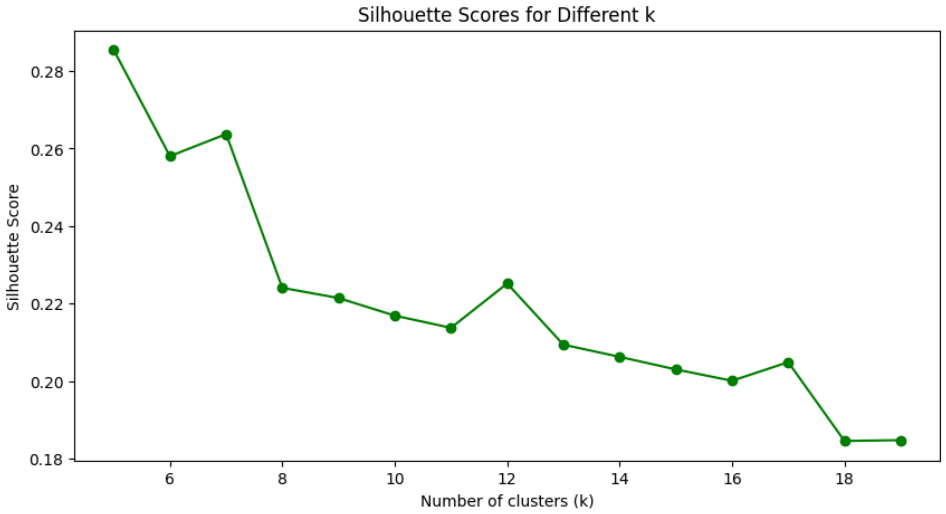
Used two primary evaluation methods:

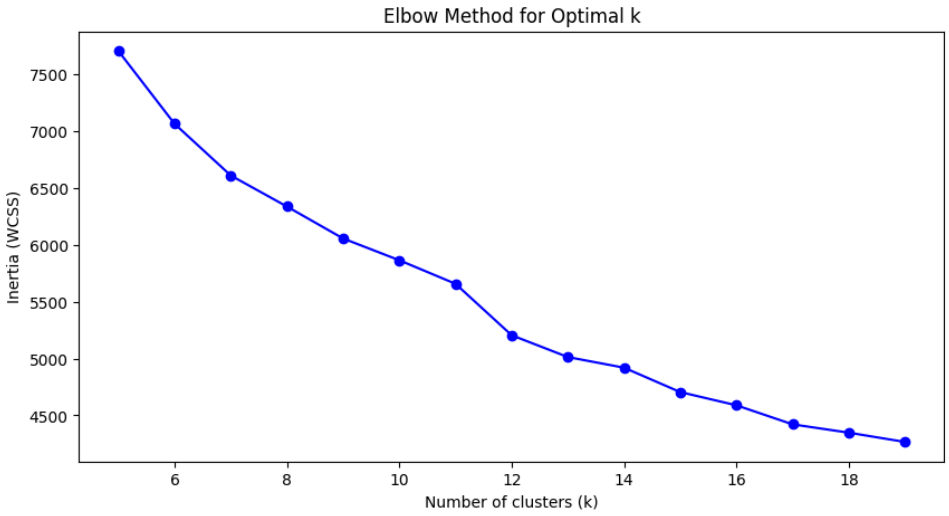
1. Elbow Method Plot: Shows the relationship between number of clusters and inertia
    - Optimal k value identified at 8 clusters where the "elbow" occurs
    - After this point, additional clusters provide diminishing returns
2. Silhouette Scores: Measures cluster cohesion and separation
    - Highest silhouette score achieved at k=7
    - Indicates good cluster definition and separation





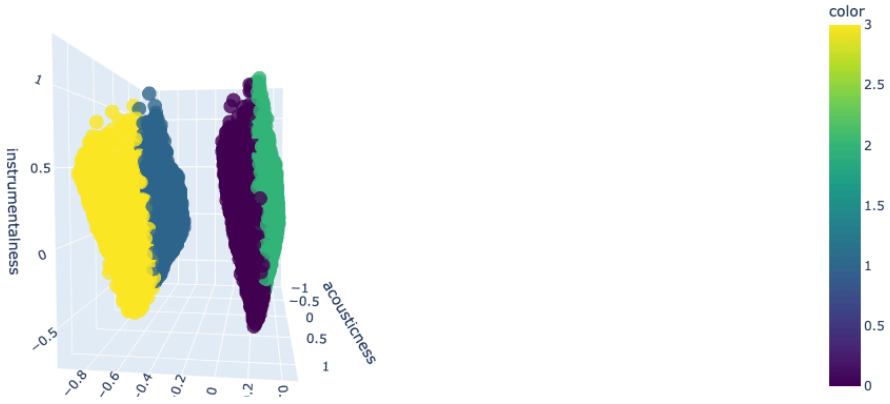
The above were without interaction terms, the below are with Interaction Terms:

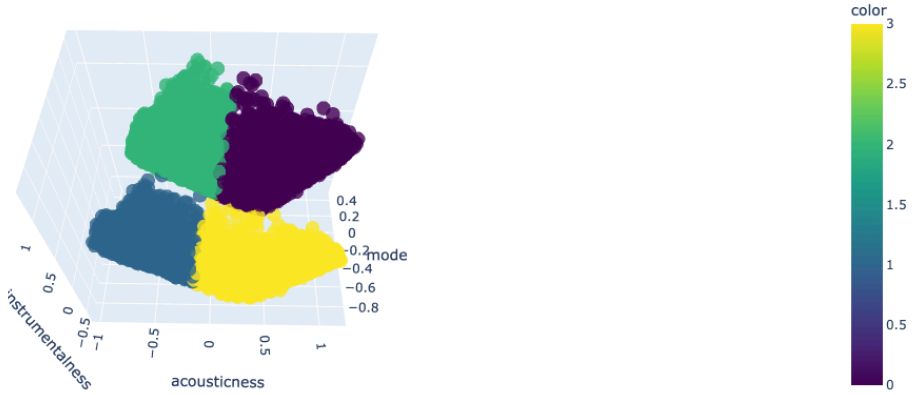




Notice on the IPNBY file I tweaked the Elbow and Silhouette models and wasn't able to get them back to their original output. Nevertheless, they display the same concept.

For my personal interest, I used a third-party tool to generate 3d clusters that separate songs depending on three factors:instrumentalness, acousticness, and mode. See below for the cluster models:





3.2 Principal Component Analysis (PCA)

Principal Component Analysis is a dimensionality reduction technique that transforms high-dimensional data into a new set of uncorrelated features called principal components. It works by:

- Finding directions (components) in the data that capture the most variance
- Creating new features that are linear combinations of the original features
- Ordering these new features by how much variance they explain

In the analysis:

- 9 components were needed to retain 90% of the variance

- The first three components explained approximately 65% of total variance
- Key findings from component loadings:
    - First component: Dominated by loudness and energy features
    - Second component: Strong influence from acousticness and instrumentalness
    - Third component: Mainly influenced by danceability and valence

3.3 Prediction Models

3.3.1 Decision Tree Implementation

A Decision Tree model creates a flowchart-like structure where:

- Each internal node represents a decision based on a feature
- Each branch represents the outcome of that decision
- Each leaf node represents a final prediction
- The model makes predictions by following the path from root to leaf

The implementation showed:

- Limited predictive power with R^2 value of 0.158
- Mean Squared Error of 399.84
- Suggests difficulty in capturing complex relationships in music popularity

3.3.2 Random Forest Analysis

Random Forest is an ensemble method that:

- Creates multiple decision trees using random subsets of features and data
- Combines predictions from all trees to make final predictions
- Generally provides better performance than single decision trees
- Helps identify important features through feature importance scores

Key findings:

- Improved performance over single decision tree
- Feature importance revealed:
    - Loudness and energy as strongest predictors
    - Acousticness and danceability as secondary factors
    - Tempo and mode as least influential

The modeling results suggest that predicting song popularity is a complex task where:

1. Multiple audio features contribute to popularity
2. Simple linear relationships don't fully capture popularity patterns

3. Ensemble methods (Random Forest) perform better than single models
4. There may be important external factors not captured in the audio features

Random Forest Model Performance:

- MSE: 310.38
- R^2 Score: 0.347 (34.7%)

3.3.3 Support Vector Machines (SVM) Model

The final modeling approach utilized Support Vector Machines (SVM) to predict song genres based on audio features. SVMs are particularly well-suited for this task due to their ability to handle:

- Multiple classes through one-vs-rest classification
- Non-linear relationships between features
- High-dimensional data effectively

Model Architecture:

- Implemented using OneVsRestClassifier with LinearSVC as base estimator
- Features standardized using StandardScaler within Pipeline
- Focused on top 25 most frequent genres to manage complexity
- Genre labels transformed using MultiLabelBinarizer for multi-label classification
- Cross-validation implemented with stratified k-fold splitting

Results and Performance:

The SVM model achieved an overall accuracy of 2.48%, with notable variations.

Strongest Performance:

- Rock genre: 73% precision, though with 7% recall
- Hip Hop: 63% precision with 18% recall
- Adult Standards: 54% precision with 20% recall (F1-score: 0.29)

Overall Metrics:

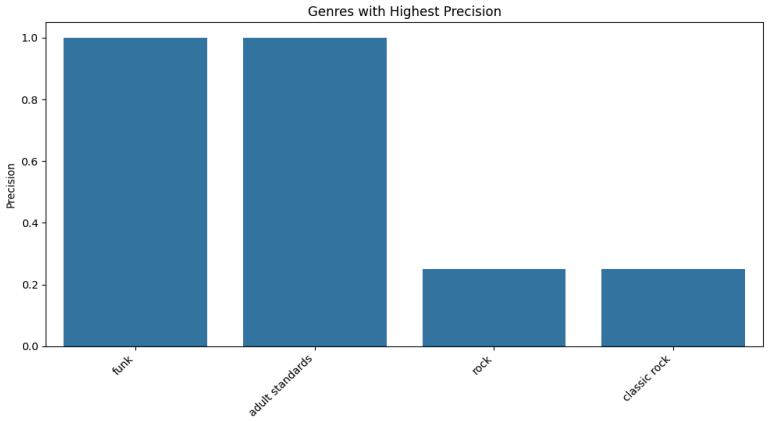
- Micro Average: 52% precision, 2% recall
- Macro Average: 11% precision, 2% recall
- Weighted Average: 13% precision, 2% recall

Key Findings:

- Model showed specialized ability in distinguishing certain distinct genres
- Significant precision-recall trade-off observed
- Performance highlights:
    - Strong at identifying rock music characteristics

    - Effective at recognizing hip hop patterns
    - Reliable in classifying adult standards
- Limitations:
    - Poor performance on popular genres like pop and dance pop
    - Zero performance on 20 out of 25 genres
    - Generally low recall scores indicating conservative predictions
    - Struggled with genres having overlapping characteristics

The results suggest that while the SVM model can effectively identify certain distinct musical styles, it requires significant improvement for broader genre classification tasks. The strong performance in specific genres indicates potential for specialized applications rather than general-purpose genre classification.



4 Summary and Recommendations

4.1 Key Findings

Strong Feature Correlations:

- Valence and danceability (0.559)
- Acousticness and energy (-0.749)
- Energy and loudness (0.782)
- Acousticness and popularity (-0.573)
- Loudness and popularity (0.457)

Model Performances:

- Decision Tree: MSE 306.68, R^2 0.354
- Random Forest: MSE 310.38, R^2 0.347
- SVM showed strong performance in specific genres:
    - Rock (73% precision)
    - Hip Hop (63% precision)

- Adult Standards (54% precision)

Clustering Analysis:

PCA revealed 8-9 components needed to retain 90% variance
Optimal cluster number identified between 7-8 clusters
Clear genre groupings emerged based on audio features

4.2 Limitations

Performance Limitations:

- Limited predictive power ($R^2 < 0.4$ for tree-based models)
- Poor performance on less common genres
- High variance in popularity predictions

Data Quality:

- Missing genre information for some tracks
- Imbalanced genre distribution
- Potential bias in popularity metrics for niche genres

4.3 Future Work

Model Improvements:

- Explore deep learning approaches
- Implement hybrid models combining multiple techniques
- Develop genre-specific prediction models

Feature Engineering:

- Create interaction terms between correlated features
- Develop more sophisticated audio feature extraction
- Include temporal and artist-related features

Data Enhancement:

- Collect additional metadata
- Include historical popularity trends
- Incorporate user listening patterns

If I were to move forward with this project, wether it be for recruiting or for personal interest, the steps I would follow would be:

1) Get a model that clusters the songs satisfactorily
2) Figure out how the recommendation will work
    a) User gives a song

b) Model associates song with cluster
c) Retrieve songs from cluster
d) Recommend random songs from cluster with differing levels of popularity
    i) Eg. 5 “highly popularity” songs, 5 “medium popularity” songs, and a few “low popularity” songs so the user can discover newer music

4.4 Conclusion

To briefly sum up, the analysis reveals that while audio features alone can provide some insight into song popularity, the relationship is complex and not easily captured by single models. The most successful approach combined multiple techniques, with clustering and genre-specific analysis showing particular promise. Through this project, I encountered several challenges, particularly when dealing with such a large dataset. Initially, my models were taking extremely long to compute, which was frustrating and made testing different approaches nearly impossible. However, by reducing the number of estimators in the Random Forest model from 100 to 50 and carefully selecting my data ranges, I was able to achieve similar results with much faster computation times. I especially enjoyed discovering how certain genres like rock, hip hop, and adult standards showed distinct audio patterns that made them easier to classify. The clustering analysis was particularly interesting as it helped me understand how songs naturally group together based on their characteristics. While the models' performance wasn't as high as I initially hoped (Decision Tree $R^2 = 0.354$, Random Forest $R^2 = 0.347$, SVM accuracy = 2.48%), the project taught me valuable lessons about handling large datasets and the importance of balancing model complexity with computational efficiency. Future work should focus on feature engineering and hybrid modeling approaches to better capture the nuanced relationships between song characteristics and popularity. A consideration could be to make the project open source and have other interested parties contribute to the overall goal.

Thanks for taking the time to read my writeup!