ISyE 6740
Final Report

Project Title: **Optimizing Spotify Song Recommendations Using Audio Features**

Table of Contents

1. Problem Statements and Project Overview

As a leading music streaming service, Spotify offers personalized song recommendations to enhance the user experience. However, in 2023, the growth rate for monthly active users (MAU) decreased to 23%, down from 30% between 2017 and 2020. In the first quarter of 2024, MAU grew only by 19% year over year, significantly short of expectations. Furthermore, the growth rate for paid subscribers dropped from 40% in 2017, to 35% in 2018, and then to around 15% between 2021 and 2023. One contributing factor to the slowdown in growth may be that Spotify's users might not receive the most relevant song recommendations, potentially leading to less engaging user activities. The vast number of songs and genres complicates the recommendation process. There is a need for a robust model that can efficiently analyze audio features and provide personalized song recommendations to cater to the diverse preferences of Spotify's users.

This project aims to identify the best recommendation models for Spotify using audio features to explore and classify songs. We will utilize a real dataset extracted from Spotify's Web API from six main categories: EDM, Latin, Pop, R&B, Rap, and Rock. Our goal is to improve the accuracy and relevance of Spotify's song recommendations, thereby increasing user satisfaction and engagement.

2. Data Source

Our data is downloaded from Kaggle.com, with information on around 30,000 songs from the following genre: EDM, Latin, Pop, R&B, Rap, and Rock. The data collected includes various audio features such as tempo, key, mode, loudness, energy, danceability, valence, and acousticness. All these data are combined into a single large dataset for machine learning data analysis.

3. Methodology

Our methodology begins with data cleaning, where we pre-processed the dataset for missing values, correct inconsistencies, and ensure appropriate data types and encoding formats. Next, we performed feature selection to identify the most relevant audio characteristics and metadata that impact user engagement and listening preferences. For the proposed model, we explored multiple machine learning algorithms, including K-Nearest Neighbors (KNN), Naive Bayes, Support Vector Machines (SVM), Logistic Regression, and Random Forests, to determine the most effective recommendation approach. Finally, we conducted model training and evaluation using cross-validation and various performance metrics (precision, recall, F1-score) to assess each model's accuracy and predictive performance, and make sure our chosen recommendation algorithm matches personalized user experience.

3.1. Data Wrangling

We cleaned, normalized, and transformed the dataset to prepare it for model building. This process involved handling missing values, scaling features to standardize data, and transforming variables to ensure compatibility with machine learning models.

3.2. **Exploratory Data Analysis (EDA)**

We analyzed trends in song popularity, user preferences, and song attributes, as well as investigated correlations between various song features.

There are six playlist genres, each of which comprises 15%-19% of the entire dataset, indicating that the distribution of each genre is averagely weighted. See Figure 1 below.

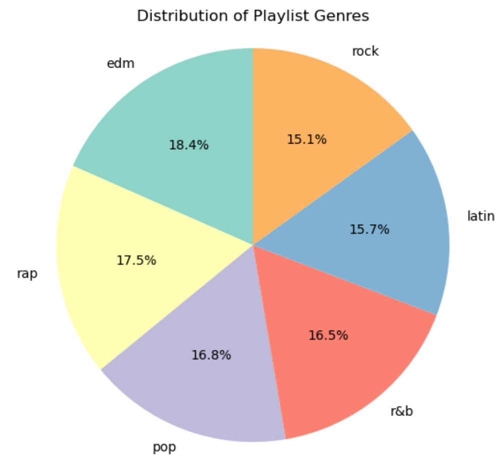


Figure 1: Distribution of Playlist Genres

The field "Track Popularity" indicates song popularity, where higher popularity means more popular a song is. The distribution of this field indicates most songs are in the scores between 30-80. See Figure 2 below.

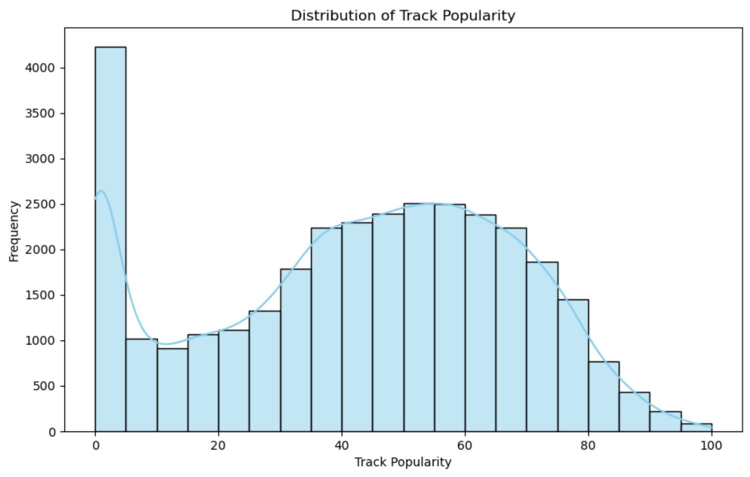


Figure 2: Distribution of Track Popularity

As shown in Figure 3 below, we created a correlation matrix with the numerical fields and found that there are no highly correlated variables.

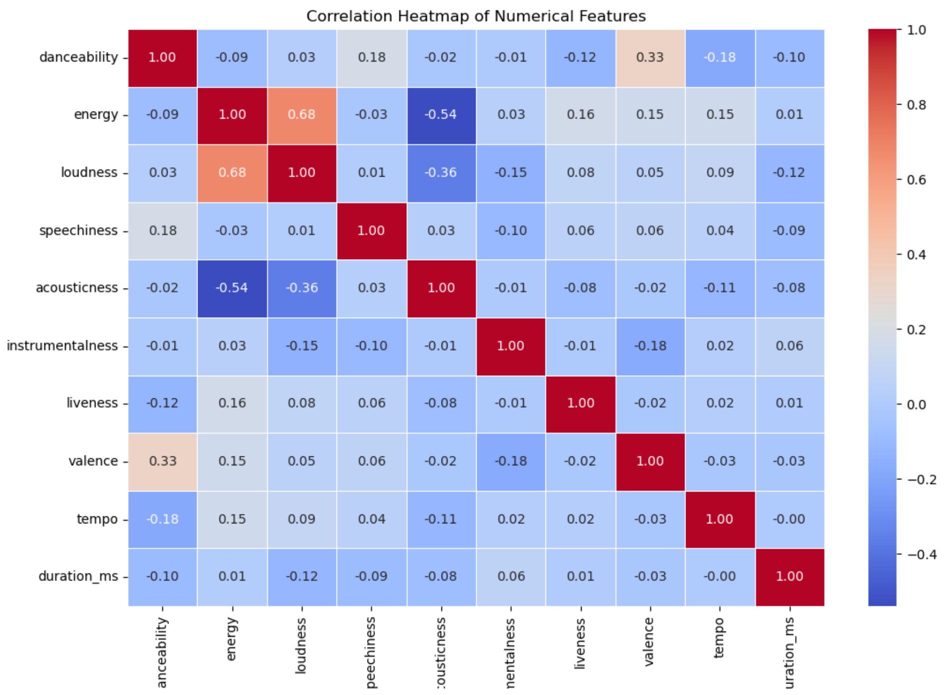


Figure 3: Correlation Heatmap of Numerical Features

Last but not least, there are 33.3% of missing values, which we decided to exclude those rows from our dataset for modeling. See Figure 4 below.

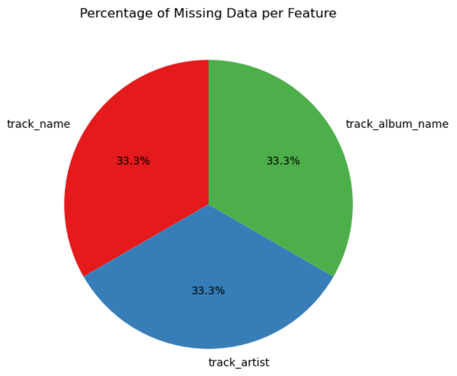


Figure 4: Percentage of Missing Data per Feature

3.3. Feature Selection

We performed feature selection to identify the most relevant variables for building an effective recommendation model. For the dependent variable, we created a new binary target variable, "Popular," by comparing each song's track popularity to the calculated median of song popularity. Those above the median were labeled as "True" and others as "False." For independent variables, we selected a set of numerical features, including danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo, and duration_ms, as predictors to capture various audio characteristics that influence user preferences on Spotify.

4. Model Building

4.1. Initial Model Selection

After data cleaning and exploration, we aim to develop a song recommendation model based on song popularity. By analyzing patterns and similarities in the dataset, we need to find a model that can suggest songs to users based on their listening history and preferences.

To achieve the goal, we explored a variety of machine learning algorithms, including KNN, Naive Bayes, SVM, Logistic Regression, and Random Forest. Each model was trained and tested using the preprocessed data. We also experimented with different combinations of audio features to determine their impact on the recommendation quality.

4.2. K-Nearest Neighbors (KNN)

KNN is a straightforward and intuitive algorithm that is easy to implement compared to many other machine learning methods. It is capable of handling both classification and regression tasks, which makes it a good candidate in our initial selection. The core concept of KNN is to predict a data point's label by considering the majority label among its 'k' nearest neighbors. The performance and accuracy of the algorithm can be significantly influenced by the choice of 'k' and the distance metric.

In the song recommendation model, we use KNN to cluster songs based on their similarities in audio features. This allows us to recommend songs that are likely to resonate with users' preferences. To determine the optimal number of clusters for grouping songs, we employed the elbow method (as shown in Figure 5 below), which helps identify the best balance between the number of clusters and the explained variance.

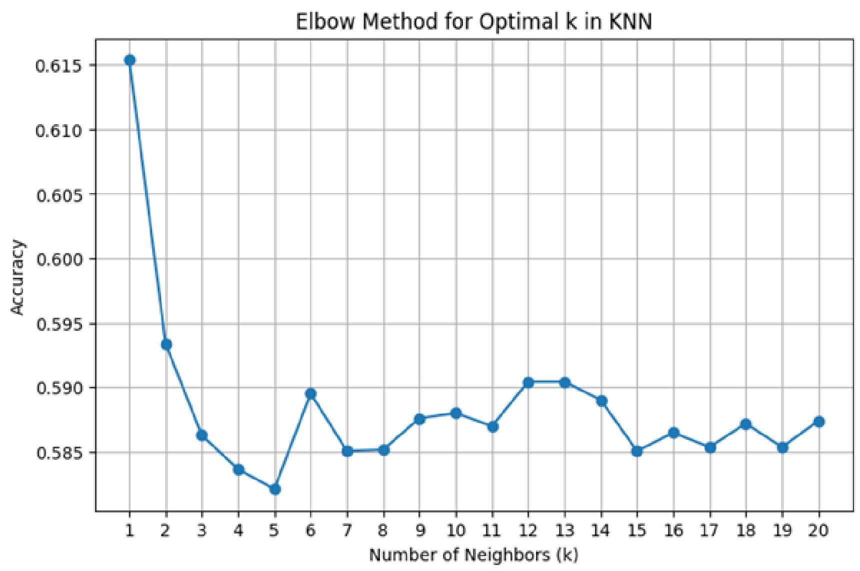


Figure 5: KNN - Elbow Method

4.3. Naive Bayes

Naive Bayes is a popular choice for classification tasks due to its known fast speed, simplicity, and better performance across binary and multi-class problems. It is known for being efficient and predicting relatively reliable results, even with large datasets.

For our project, Naive Bayes is included to classify songs into genres based on user preferences. Its assumption of feature independence makes it particularly useful for handling classification tasks efficiently, where features are loosely related, such as tag-based or metadata driven classifications.

4.4. Support Vector Machine (SVM)

We included SVM due to its ability to handle complex classification, regression, and outlier detection tasks through effective data transformations that define boundaries between data points based on classes, labels, or outputs. This makes it a suitable choice for clustering songs based on audio feature similarities, so that songs most likely to appeal to users are recommended.

4.5. Logistic Regression

Logistic regression is a good option for binary classification tasks, predicting the probability of discrete outcomes with a score between 0 and 1, which can then be mapped to class labels. For our project, logistic regression is employed to determine whether a user would like or dislike a song. It also helps classify songs as recommended or not recommended based on a probability threshold for more personalized and relevant suggestions.

4.6. Random Forest

Random Forest is a robust machine learning algorithm that can handle both classification and regression tasks. It builds a collection of decision trees and combines their outputs through ensemble learning. The randomness in this process can enhance the model's performance and reduce the likelihood of overfitting on the training data, which makes it a good option as the song recommend model. In can improve the accuracy of classifying song popularity. Furthermore, it allows us to assess feature importance, revealing that the independent features have similar levels of significance in the recommendation process.

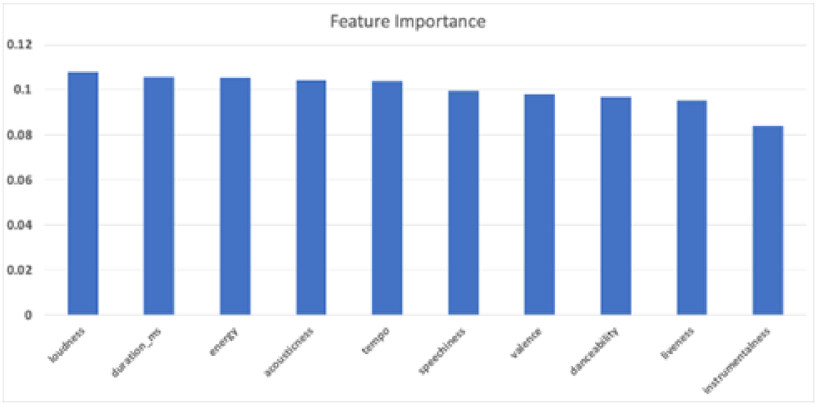


Figure 6: Feature Importance

5. Evaluation and Final Results

5.1. Evaluation of all Models

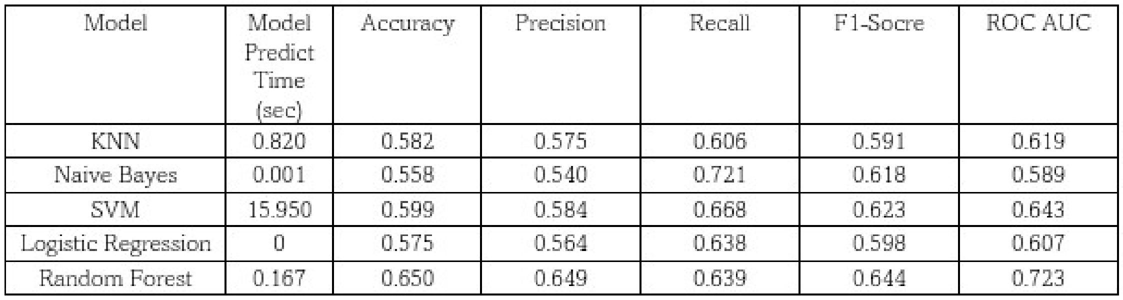
Our models were evaluated using several key performance metrics to understand their effectiveness, as shown in Figure 7 below:

Accuracy: This measures the overall correctness of the model. As shown in the table below, Random Forest achieved the highest accuracy at 0.650, while other models have accuracies ranging between 0.55 and 0.60.

Precision, Recall, and F1-Score: These metrics assess the model's ability to classify each class accurately. Precision evaluates how many predicted positives were correct, recall measures how many actual positives were identified, and the F1-Score combines both precision and recall into a single value. Random Forest outperformed other models in precision, recall, and F1-Score, highlighting its superior ability to classify songs effectively.

ROC AUC: This metric assesses the model's ability to distinguish between classes across different threshold levels. A higher AUC value indicates better overall model performance. In our evaluation, Random Forest achieved the highest ROC AUC of 0.723, outperforming other models.

Model Predict Time: The time it takes for each model to make predictions is also an important consideration, as faster execution times improve user experience and system efficiency.



Model	Model Predict Time (sec)	Accuracy	Precision	Recall	F1-Socre	ROC AUC
KNN	0.820	0.582	0.575	0.606	0.591	0.619
Naive Bayes	0.001	0.558	0.540	0.721	0.618	0.589
SVM	15.950	0.599	0.584	0.668	0.623	0.643
Logistic Regression	0	0.575	0.564	0.638	0.598	0.607
Random Forest	0.167	0.650	0.649	0.639	0.644	0.723

Figure 7: Model Performance Comparison

5.2. Cross-Validation

Cross-validation is a crucial method to make sure the predictive models are reliable and perform well on unseen data. We focused on the Random Forest and Naive Bayes models because they have shown effectiveness in handling datasets with many features, as indicated by the evaluation results in the previous section.

We used K-Fold cross-validation, a method where the dataset is divided into 'K' equal parts. Each part serves as a test set once, while the remaining K-1 parts form the training set. This approach ensures that every fold accurately represents the overall data distribution and maintains a balanced number of samples across classes.

Based on the cross-validation outcomes and previous evaluation metrics shown in Figure 8, Random Forest model shows a higher ROC AUC and better overall performance. These results indicated the model's capability to generalize across different subsets of data, making it the optimal choice for our project.

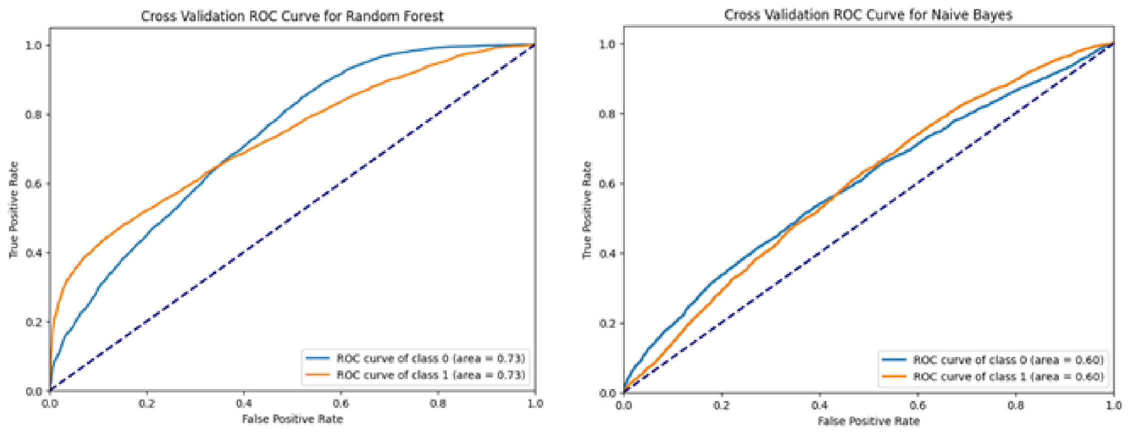


Figure 8: Cross Validation

6. **Conclusion**

In conclusion, looking at the comparison of different machine learning models, we can see each one has its strengths and weaknesses for recommending popular songs. The Random Forest model stood out as a strong choice because it provided reliable and accurate predictions, thanks to its ability to handle complex relationships in the data. KNN offers fast predictions but can sometimes fall short in accuracy for more challenging datasets. Naive Bayes is quick but doesn't perform as well in terms of recall and precision. SVM, while slower to train, delivered a balanced F1 score, showing good potential for classification tasks. Logistic Regression also performed reasonably well but didn't quite match the robustness of Random Forest.

Given these results, Random Forest was selected because of its combination of accuracy, robustness, and reduced risk of overfitting. Moving forward, there's room to fine tune other models and explore more advanced feature engineering to improve the overall recommendation system even further.