Machine Learning for Music: Predicting song popularity using song characteristics

Pranav Bhargava[i] *& Rohini Iyer*[ii]

Introduction

With the advent of copious amounts of data, machine learning algorithms and greater computing power, data-driven applications are seeping into all spheres of life. One such application is in music, where music studios base their decisions of producing an album/song on a variety of metrics such as *commercial potential, artist potential, genre and style, and songwriting quality.* We wonder if there is a more objective way to judge a song without necessarily looking at the "artist." It helps democratize the process of song making and provides a level playing field to upcoming musicians and song composers.

In this pursuit, we aim to provide a novel way of predicting the popularity of a song using song characteristics. Currently, companies conduct market research and focus groups, rely on industry experts, and social media analytics to assess user engagement *after* the song has been released. We leverage Machine Learning to predict engagement, a proxy for popularity, *before* the song has been released. Machine Learning methods can identify trends and patterns in data that may not be immediately apparent to human analysts, making the talent selection for music companies more data driven. Predicting which songs will garner more engagement/popularity will also enable artists to create content with greater potential for mass appeal.

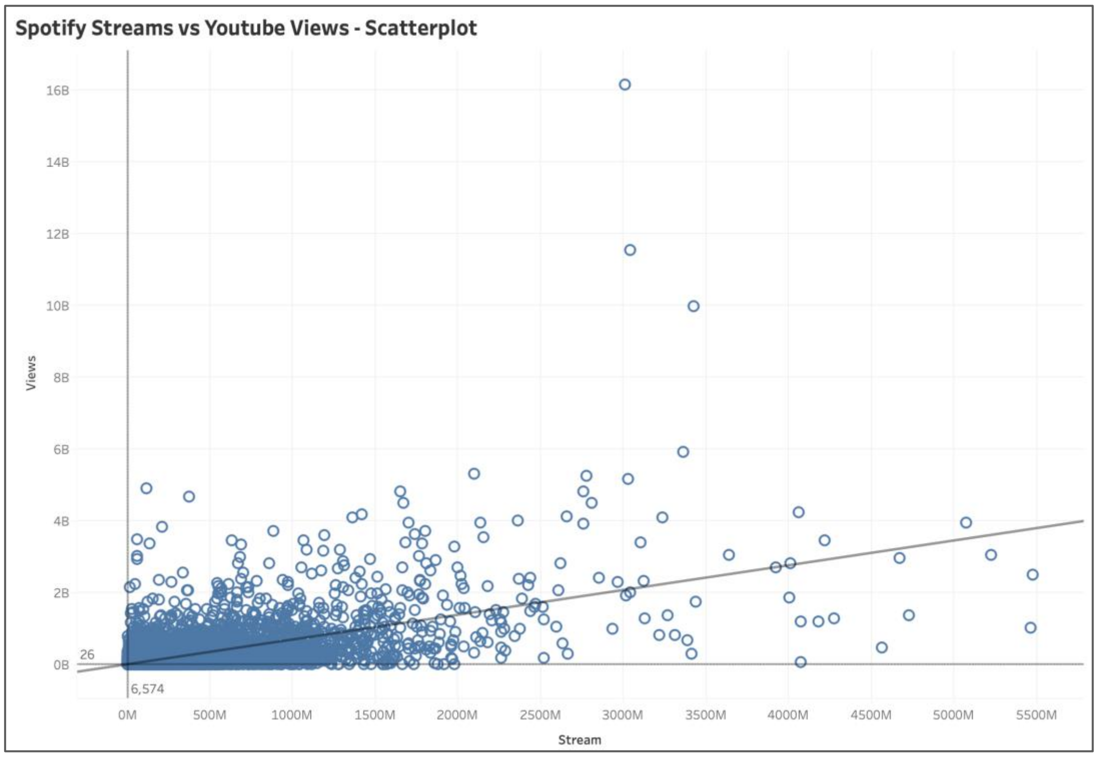
Approach

Our analysis is based on a dataset available on Kaggle[iii] that records a song's engagement on YouTube (views, likes, comments) and Spotify (streams) as well as has data on song characteristics (that we discuss later). This data covers the **top 10 songs** of **~2000 artists** from across the world, spanning different languages, genres, and music styles.

First, we define our outcome variable as **Streams per Views**. It is a metric that intuitively says how many Spotify streams the song gets for every view it receives on YouTube. There are a few advantages of using this transformed variable. First, it helps in making our outcome variable standardized and easy to compare across regions, artists, genres, and other dimensions. Second, this transformation gives us continuous, more interpretable values in both the training and test dataset. Lastly, based on preliminary analysis, we see around 75% of songs in our database (presented below) have a higher number of Spotify streams as opposed to YouTube Views. This might point to the fact that listening to songs on Spotify is preferred over watching the same song on YouTube, and hence, we seek to maximize the Spotify streams as a measure for greater popularity.

The second part of our approach is choosing the input variables or predictors that will help us predict the outcome variable. Out of the 27 variables in the dataset, we choose to focus on 10 key variables that give us an insight into the characteristics of the songs. These are ***Danceability, Key, Energy, Loudness, Speechiness, Acousticness, Instrumentalness, Liveness, Valence, Tempo.*** A detailed description of these variables has been provided in *Table 1* below. It is worth noting that the dataset provided on Kaggle does not provide any insight into how these measures were calculated. But the fact that data on these measures is available means something and provides an objective insight into it. Further, views and streams show a moderately strong correlation as evident from the graph below. Songs that are viewed many times on YouTube also tend to have higher streams on Spotify (as noticed from the trendline).

Figure 1: Scatterplot of Spotify streams v/s YouTube views



Data Description

We describe the variables we use for our analysis in the following table. These include the predictor variables i.e., the song characteristics and the outcome variables.

Table 1: List and description of predictor variables

Predictor	Description
Danceability	Describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
Energy	A measure from 0.0 to 1.0 that represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual

	features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
Key	The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g., 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.
Loudness	The overall loudness of a track in decibels (dB). Loudness is the quality of a sound that is the primary psychological correlation of physical strength (amplitude). Values typically range between -60 and 0 db.
Speechiness	Detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g., talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
Acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
Instrumentalness	Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
Liveness	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides a strong likelihood that the track is live.
Valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g., happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g., sad, depressed, angry).
Tempo	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
Views	Number of Views on YouTube
Streams	Number of Streams on Spotify

Model selection

We are interested in exploring regression models that would allow us to accurately predict the popularity or engagement of songs using characteristics like danceability, energy, etc. We start with a **Simple Linear Regression (OLS)** to model a relationship between the outcome variable "streams/views" and the predictor variables. We use the test error rate for the linear regression model as the baseline to compare all our other models against. Traditionally, linear regression has low variance but also high bias. Further, not all predictor variables in our dataset exhibit a linear relationship with the outcome variable of interest.

Hence, we explore additional machine learning methods such as decision trees. **Decision trees** can handle both discrete (e.g., key) and continuous data (other predictors) and they can also capture non-linear relationships between the dependent and independent variables. Decision trees are also easy to interpret and visualize. While they have low bias, they are not only prone to overfitting but also have a large variance. Hence, we also explored ensemble learning techniques such as random forest, bagging, and boosting methods that combine multiple models to improve performance, especially to reduce the variance of the decision tree model.

Accordingly, instead of relying on the results of a single decision tree, **Random Forest** allowed us to train a collection of decision trees on random subsets of the data, which helps to reduce the variance of the individual trees and improve the accuracy and robustness of the overall prediction. Additionally, from the table below which summarizes the outcome variable "streams/views", we observe that 75% of the observations have a "streams/views" value between 0 to 12. However, in the remaining 25%, the value of this outcome variable scales exponentially up to 38,637,558 – indicating the presence of outliers in the data. Random forest helps reduce the impact of such outliers and noise in the data, by combining and averaging the output of multiple decision trees.

This also makes this model less prone to overfitting than single decision trees. For this data, the model is configured to use 5000 trees to create the random forest.

Table 2: Summary of the outcome variable – Streams Per Views

Minimum	1st Quartile	Median (2nd Quartile)	3rd Quartile	Maximum
0	1	3	12	38,637,558

Since the relationship between the predictor and outcome variables is complex, we also used **bagging**, short for Bootstrap Aggregating, a technique used for reducing variance and increasing accuracy, by combining multiple models while maintaining the same bias as the decision tree. Sampling from the original dataset with replacement to create multiple bootstrap samples, we fit a separate regression model to each of these samples to improve the predictive performance of a single regression, thus reducing overfitting and increasing the stability of the model.

Further, to improve the generalization ability of the model so that it can be utilized beyond the scope of this dataset and to improve the accuracy and reduce the bias of regression algorithms such as decision trees and linear regression, we use **Boosting.** However, we proceed with caution because while boosting is useful when dealing with large, complex datasets that require more advanced modeling techniques to extract meaningful insights, it can also be prone to overfitting. For our data, the boosting model assumes the response variable to follow a Gaussian distribution. As in random forest, the algorithm creates a sequence of 5000 trees, where each tree can have a maximum depth of 4.

Lastly, we take advantage of a non-parametric supervised machine learning model - **K-Nearest Neighbors**. It can capture nonlinear relationships better than linear models like linear regression and does not make assumptions about the underlying data distribution. Contrary to linear regression, KNN usually tends to have low bias but high variance. Additionally, the major disadvantage of KNN that we encountered was that it was computationally time consuming and intensive for our data (~19,000 observations), as it required calculating the distance between each observation in the dataset. We use cross validation to identify the ideal "k" for this data and find that considering one neighbor i.e., k = 1, gives the lowest test mean squared error.

As described previously, our dataset contains approximately 20,000 observations across 28 variables. Given that the number of observations (n) is larger than the number of potential predictors (p) i.e., 27 (even though we used only 10 variables for the predictions); ours is not a high dimensional dataset. This eliminated the need to consider shrinkage and dimension reduction methods such as Ridge, Lasso, Principal Component Analysis (PCA), or Partial Least Squares (PLS). Furthermore, since we decided to tackle a regression problem, unlike using classification that has traditionally been used to answer similar policy research questions[iv], we skipped classification algorithms like logistic regression, K-means clustering, hierarchical clustering, and other similar models taught in class. Lastly, due to bandwidth, time, and resource constraints, we also restricted ourselves from exploring further regression models such as Smoothing Splines, Generalized Additive Models (GAMs), Polynomial Regression, and Stepwise Regression to name a few.

Results

We run the six models discussed above on our dataset using an 80:20 split between the training and test data. We use the metric Mean Squared Error of Prediction (MSEP) to evaluate the performance of the different models with one another. MSEP is a measure of how well a regression model fits the data and indicates the average squared difference between the predicted and actual values of the response variable. For each model, we compute the MSEP first for the training data, followed by the test data. The results are tabulated below.

Table 3: Mean Squared Error of Prediction (MSEP)

Models explored	Streams/Views (Training) (in 10^8)	Streams/Views (Test) (in 10^8)
Linear regression	981.86	0.64
K-Nearest Neighbors (k=1)	984.76	2.61
Decision Tree	982.14	0.38
Random Forest	252.36	3.08
Bagging	494.63	3.04
Boosting	6.12	34.62

Seed: 222 | **Original N:** 20,718
Missing data: 1,655 observations had missing data across one or multiple variables and hence were dropped
Training data: Contains 80% of non-missing data observations = 15250 observations
Test data: Contains 20% of non-missing data observations = 3813 observations

From the above table, we observe that the **Boosting model has the lowest MSE of 6.12 x 10^8 on the training dataset**, while all other models have much higher values. However, the goal of building an ML model is not just to fit the training data well, but also to generalize to unseen (test) data. Therefore, the performance on the test set is more important for evaluating the effectiveness of the model. Looking at the test MSE, we see that **the model with the lowest test MSE is the Decision Tree with an MSE of 0.38 x 10^8**. Thus, for this dataset, the **Decision Tree model performs the best on the test set in terms of minimizing the MSE**. In fact, the boosting model (which has lowest training MSE) has the highest test MSE among all the models, indicating that it overfit the training data (as expected and highlighted in the previous section).

It is also interesting to note that after the decision tree, the model that performs second best on the test data is the most simple and basic - linear regression (OLS) model. While more complex models like random forest and bagging having a higher test MSE than simple linear regression may feel counter-intuitive, it is worthwhile to note that the linear regression model assumes a linear relationship between the predictors and the response variable, which may indeed be a reasonable assumption in this case. Similarly, the KNN model has a higher MSE for both training and test data as compared to linear regression and decision tree models, indicating that it may not be the best fit for this data. One reason for this could be because KNN assumes that the underlying data is locally smooth, which may not be the case here. Lastly, Random Forest and Bagging models have relatively low MSE on the training data but perform worse on the test data compared to the linear regression and decision tree models. This again suggests overfitting on the training data and lack of generalizability on the test data.

Hence, we decided to finally proceed with the **decision tree** model for our current dataset because it has (1) **lowest test MSE**, (2) **low bias**, (3) **moderate variance,** and is **(4) less prone to overfitting** compared to the other models like KNN, random forest, and boosting.

Conclusion

The decision tree model gives us the best prediction out of all the machine learning algorithms we considered. For similar level of training MSE, we find a low test MSE among the Linear Regression, Decision Trees, and K-nearest neighbor models. Our results differ a bit from similar studies. For instance, a study trying to predict hit Afro beat songs using Spotify Data[v] found Random Forest and Neural Network to perform well, however, it was a classification exercise.

One limitation of our analysis is that it does not consider the timeframe in which the song gained a high streams per view. For a song to be considered a hit or popular, it is not just necessary that the song garners user engagement but that it also does so in short time. We would like to consider this in future iterations of this study. Another modification could be redoing this as a classification exercise, like in the study mentioned previously.

For future studies, we recommend using Decision Trees at the very least along with time as a key variable to enhance the prediction of song popularity using machine learning algorithms. In terms of implementation, the model can be made richer with more data. Spotify gives access to data on song characteristics using an easy to call API (Application Programming Interfaces). AT ideal implementation would enta music studio companies inviting musicians to send in a piece of their music, which when analyzed using our algorithm, lets them know the predicted user engagement.

---

[i] Pranav Bhargava. "MPA/ID 2023, Harvard Kennedy School." LinkedIn. https://www.linkedin.com/in/pranavbhargava93/.

[ii] Rohini Iyer. "MPP 2023, Harvard Kennedy School." LinkedIn. https://www.linkedin.com/in/rohini-venkitaraman-iyer/.

[iii] Salvatore Rastelli. "Spotify and YouTube." Kaggle. https://www.kaggle.com/datasets/salvatorerastelli/spotify-and-youtube

[iv] Madhan Karky: The lyrics engineer," Factor Daily, May 9, 2017. https://archive.factordaily.com/madhan-karky-lyrics-engineer/

[v] Adewale Adeagbo, et al. Predicting Afrobeat Hit Songs using Spotify Data, 2017. https://arxiv.org/pdf/2007.03137.pdf.