Hit Predictor: Forecasting Song Success with Machine Learning

Kritika Joshi and Iniya Nathan

The University of Texas at Dallas

ITSS 4382.001

Applied Artificial Intelligence/Machine Learning

Professor Lidong Wu

May 5, 2025

Introduction

Millions of songs are released each year, all of them with varying popularity. Sometimes people can guess whether or not a song will be a hit, perhaps because the artist or genre is popular at the time. However, with just guessing, it is hard to predict the next hit song from numerous potential hits. With the right data and tools, people can learn the trends in the music industry, figure out what features are becoming more popular, and which features are becoming old. By learning what parts of making music are crucial to making the next big hit, artists and producers can develop songs that are almost guaranteed to become popular, wasting less time with music ideas that would not be popular.

The goal of this project is to use what we learned in class and outside learning on AI and machine learning, to figure out what makes a song a hit song. By analysing a Spotify dataset we found on Kaggle, we aim to develop a predictive model that separates potential hit songs from the real deals with the use of supervised learning algorithms. By using AI, we can find the features, or feature combinations, that make songs popular by using AI to sort through current released music and label whether or not it is a hit accurately. Once the AI has been trained to be relatively accurate in distugishing between hit songs and not hit songs with the current data, as long as overfitting has not occurred, it should be able to perform with a similar accuracy on new songs, ergo identifying the next big hits before the music has even been released.

Findings

From the analysis of Spotify song data, we learned various vital findings regarding the predictability of song popularity using machine learning models. The information, comprising 41,099 tracks with 20 acoustic and metadata features, exhibited a weakly left-skewed distribution of popularity scores, wherein the majority of the songs banded together in the range of 30 to 60 on a 0 to 100 scale. This revealed that most of the tracks go moderately popular, but a lesser percentage go viral.

In order to facilitate operationalization of a "hit," we've operationalized it as songs in the top 25% of popularity scores (75th percentile cut-off). Surprisingly, this cut-off yielded a perfectly balanced data set, with 20,551 non-hits and 20,548 hits. The balance between the two groups probably aided in stabilizing the model training, since imbalanced class distributions are known to lead to biased predictions.

We tested three models—Logistic Regression, Random Forest, and Support Vector Machine (SVM). Random Forest performed the best at 79.5% and had perfect hit recall (84%) and low non-hit precision (82%). Its confusion matrix indicated that it correctly classified 3,454 hits and misplaced only 656, which established robustness for detecting trends pertaining to top songs. Logistic Regression, as simpler, achieved a 73.4% accuracy, suggesting linear correlations between attributes like danceability, energy, or tempo and popularity, but is not exhaustive. SVM trailed slightly at 78.1% accuracy, which might be attributed to its feature scaling sensitivity and computational constraints with large datasets.

Collectively, the results highlighted that while acoustic features are predictive, factors external to the dataset—cultural trends, advertising, or artist reputation—may have significant impacts on shaping popularity. The project highlighted the potential and limitations of data-only approaches in predicting subjective outcomes like music success.

Coding Explanation and Libraries Usage

The project leveraged several Python libraries to streamline data processing, visualization, and modeling. Pandas and NumPy form the backbone of data manipulation. Pandas was employed to load the CSV data, remove the unnecessary columns of track and artist.

It also helped in avoiding overfitting on specific names rather than acoustic features and handling missing values.

The ‘dropna’ method was used across rows with missing popularity scores to preserve data consistency. NumPy was employed to carry out numerical tasks, like calculating the 75th percentile cut-off as a measure for hits.

For visualization, Matplotlib and Seaborn generated histograms to analyze the distribution of popularity scores, which revealed its left-skewed nature. These visualizations were crucial to understand the data structure and to validate the choice of threshold for class balance.

For workflows including machine learning, Scikit-learn was essential. The ‘train_test_split’ function divided the data into train (80%) and test (20%) portions with stratification to ensure proportional hit and non-hit representation in the two subsets. This prevented skewed evaluations where one class may dominate the other.

The ‘SimpleImpute’ class replaced missing feature values with column means, a selection informed by the need to retain as much data as possible without inducing biases from row removal. ‘StandardScaler’ normalized features like loudness and tempo to a common scale, so that models like SVM, which are feature magnitude-sensitive, could work optimally.

All these pre-processing steps were wrapped into a ‘Pipeline’ so that the workflow would be automated and data leakage, where test set statistics might inadvertently influence training, could be prevented.

Three classification models were employed. Logistic Regression was employed as a baseline model since it is simple to interpret and efficient for linear relationships.

Random Forest, an ensemble of decision trees, was chosen because of its ability to learn non-linear interactions among features, for example, how danceability and tempo might together influence popularity.

SVM, using the ‘SVC’ class, was also attempted for its ability to handle high-dimensional data using kernel tricks, although its computational cost made scaling extremely important. The Scikit-learn-produced confusion matrices and classification reports provided detailed model performance feedback, extending past accuracy to values like precision and recall. To illustrate, the Random Forest's high recall for hits at 84% conveyed its ability to catch true popular songs, even if sometimes at the cost of calling false positives hits—a compromise it was willing to make for applications where hit identification was more valuable than being solely accurate.

Challenges

The project encountered several challenges that required iterative problem-solving. One major hurdle was addressing class imbalance. Initially, using the mean popularity as the threshold for defining hits resulted in a heavily skewed dataset, with non-hits outnumbering hits by nearly 4:1.

This imbalance risked training models that simply predicted the majority class. Redrawing the cutoff to the 75th percentile addressed this, generating a nicely balanced dataset, but raised whether such a cutoff agreed with empirical definitions of a "hit."

Another problem was handling missing values. Deleting rows that had missing popularity scores was simple, but handling missing values in feature columns was more thoughtful. Using SimpleImpler with the mean strategy preserved dataset size but introduced assumptions about data distribution, which would not hold true for all features.

Furthermore, there were challenges with model performance optimization. The SVM initially performed poorly with 70% accuracy due to improper feature scaling. Incorporating StandardScaler into the preprocessing pipeline further increased its accuracy by 8%, which emphasizes how normalization is critical for distance-based classifiers.

Computational limitations also crept in, particularly for SVM training on all 41,099 rows. Though subsampling was also proposed as a solution to reduce training time, we chose instead to simply use default hyperparameters in an effort to maintain richness in data.

Finally, interpretability was also a concern. Theoretically, Random Forest's feature importance may identify popularity drivers, but time restrictions prevented this analysis, leaving it unclear which acoustic properties had the biggest influence on predictions.

Outside Learning

The project introduced us to some advanced methods not learned in class that better informed us about the workflows of machine learning. Support Vector Machines (SVM) were unfamiliar to us. In contrast to Logistic Regression, where a linear decision boundary is fitted, SVM seeks to maximize the margin between class labels and is therefore resistant to outliers. For inseparable data, SVM utilizes kernel functions like the radial basis function (RBF) in mapping features implicitly to higher dimensions. This is a powerful method but computationally intensive, hence its slower training compared to Random Forest.

Though we defaulted on the RBF kernel, exploring other options like linear or polynomial kernels might have resulted in more improvement.

We also learned the usage of SimpleImputer and Pipeline. Missing data in teaching practice was often handled by row deletion, which is a bias-prone technique if missingness was pattern-associated. SimpleImputer performed missing value imputation using column means under the hood without changing dataset size or statistical power. Pipeline, which includes preprocessing operations and model training, was crucial in order to avoid data leakage. For instance, individually scaling the training and test sets could have introduced leakage if test set statistics influenced the training. By encasing these operations, the pipeline ensured that transformations were learned only from the training data and applied to the test set in a uniform manner.

Another new idea was Stratified sampling. Although we had performed basic train-test splits in class practice, stratification was necessary because it ensured that both subsets maintained the class distribution of the target variable (`is_hit`). This was particularly relevant, of course, with the balanced dataset we were dealing with, since a random split could have inadvertently imbalanced the hit or non-hit representation in either subset.

Finally, advanced model evaluation extended beyond minimum measures of accuracy. From review of confusion matrices and classification reports, we were able to glean information about precision-recall trade-offs. The Random Forest's high hit recall (84%) revealed its accuracy at defining actual positives, even when it labeled some non-hits as negatives every now and then.

This less extreme evaluation puts more importance on aligning measures of model performance with real-world objectives, such as maximizing hit identification for a music firm's marketing plan.

Conclusion

After completing this project, we have learned so much as a group. While working on the code, we learned so much about the different data that can be used from music and how that affects the popularity, giving us insights into the music industry that we had not really thought about before. There was more data and types of data collected from the music than we had realized would be, emphasizing that almost anything can be collected as data and used to predict and learn from.

However, more importantly, we learned a lot more about the real-world applications of artificial intelligence. While we had covered the potential and current uses of all the machine learning technologies we learned about in class, this was our first time using these techniques outside of a class assignment, where the dataset had already been chosen for us. This project highlighted the intricacies of real-world applications of machine learning, where theoretical concepts were adapted together with practical limitations.

While what was learned during class formed a solid base in algorithms such as Logistic Regression, we also had to do more learning outside of class to complete this project. Outside of class, we learned innovative methods such as SVM, pipelines, and stratified sampling, which required independent and extensive research and experimentation to understand and implement effectively.

Challenges like class imbalance and computational limitations hindered us shortly, but also were a great learning experience and showed us how artificial intelligence and machine learning technology are implemented in the real world, through trial and error. It also brought out the iterative nature of data science, with early failures being typically followed by more accurate solutions. In the end, we accomplished what we set out to do, which was to use machine learning algorithms to create a model that can predict the next big hit, and we learned a lot in the process of doing so.