Music Genre Prediction

MD Ashraf Hossain Ifty

Mir Abir Hossain

June 2022

Contents

0.1 Abstract

Over the past few years, streaming services with massive catalogs have become the primary means most people listen to their favorite music. But at the same time, the sheer amount of music on offer can mean users might be a bit overwhelmed when trying to look for newer music that suits their tastes. For this reason, streaming services have looked into means of categorizing music to allow for personalized recommendations. One method involves direct analysis of the raw audio information in a given song, scoring the raw data on various metrics. We'll be examining data compiled by a research group known as The Echo Nest. Our goal is to look through this dataset and classify songs as being either 'Hip-Hop' or 'Rock' - all without listening to a single one ourselves.

0.2 Introductions

In the modern world, recommendations have been a major feature for streaming services such as Netflix, Spotify, Prime Video, etc. To be able to efficiently recommend content accordingly based on a client's preference is integral to keeping the client hooked on their platform. Before the recommendations can be done accurately, it is essential that the model can correctly predict the genre of a given song. A model should be able to classify between different genres before it can be used for recommendations. In our project, the main aim was to formulate a classifier algorithm that predicts with utmost accuracy. We devised various machine learning models to see which model best suits our case study. The search was to find the machine learning model with the best accuracy score and F1 score. We used machine learning models such as Logistic regression, Decision Tree, Bagging classifier, Random Forrest, XGBoost , Gradient Boosting, and Logistic Regression with PCA.

0.3 Data

0.3.1 Data Collection and Insight

We used the dataset composed of songs of two music genre (Hip-Hop and Rock), we will train a classifier to distinguish between the two genres based only on track information derived from Echonest(now part of Spotify). The dataset is available in Kaggle. We collected our datasets from Kaggle. We used the datasets because it was very streamlined, and organized and the dataset was assembled by a very reputed organization, making it trustworthy to use to get authentic results.

We used two datasets named ‘$echonest - metrics'$, which is in json file format, and ‘$fma - rock - vs - hiphop'$, which is in csv file format. The ‘$fma - rock - vs - hiphop.csv'$ file contains entries about a song's $'track\_id'$, $'bit\_rate'$, $'comments'$, $'composer'$, $'date\_created'$,$'date\_recorded'$, $'duration'$, $'favorites'$, $'genre\_top'$, $'genres'$, $'genres\_all'$, $'information'$, $'interest'$, $'language\_code'$,$'license'$, $'listens'$, $'lyricist'$, $'number'$, $'publisher'$, $'tags'$, $'title'$.A song is about more than its title, artist, and a number of listens. Hence, the dataset ‘$echonest - metrics.json'$ contains informations about a song's metrics such as $'acousticness'$, $'danceability'$, $'energy'$, $'instrumentalness'$, $'liveness'$, $'speechiness'$, $'tempo'$, $'valence'$, corresponding to a song's ‘$track\_id'$ from the ‘$fma - rock - vs - hiphop.csv'$ dataset. Each musical features of each track such as danceability and acoustics is on a scale from -1 to 1.

0.3.2 Data Preparation

At first, we created a Pandas DataFrame by carefully merging both datasets, where our features were all the columns(song metrics) from the ‘$echonest - metircs.json'$ dataset alongside just the ‘$track\_id'$ column from the ‘$fma - rock - vs - hiphop.csv'$ dataset. Our target feature was ‘$genre\_top'$ from the ‘$fma - rock - vs - hiphop.csv'$ dataset. The ‘$genre\_top'$ has only two unique inputs, either ‘$Hip - Hop'$ or ‘$Rock'$. This makes our problem a binary classification problem.

0.4 Exploratory Data Analysis

There were 4802 instances and there were no null values in our merged dataset. We typically want to avoid using variables that have strong correlations with each other – hence avoiding feature redundancy – for a few reasons: 1. To keep the model simple and improve interpretability (with many features, we run the risk of overfitting). 2. When our datasets are very large, using fewer features can drastically speed up our computation time. Hence, we checked pairwise relationships between continuous variables; To get a sense of whether there are any strongly correlated features in our data, we used built-in functions in the *pandas* package. We found that there weren't any strongly correlated features.

0.5 Feature Selection

As mentioned earlier, it can be particularly useful to simplify our models and use as few features as necessary to achieve the best result. Since we didn't find any particular strong correlations between our features, we used a common approach to reduce the number of features called principal component analysis (PCA).

It is possible that the variance between genres can be explained by just a few features in the dataset. PCA rotates the data along the axis of the highest variance, thus allowing us to determine the relative contribution of each feature of our data towards the variance between classes.

However, since PCA uses the absolute variance of a feature to rotate the data, a feature with a broader range of values will overpower and bias the algorithm relative to the other features. To avoid this, we first normalized our data. There are a few methods to do this, but we used the method of standardization, such that all features have a $mean = 0$ and $standard\ deviation = 1$(the resultant is a z-score).

After we have preprocessed our data, we used PCA to determine by how much we can reduce the dimensionality of our data. We used scree-plots and cumulative explained ratio plots to find the number of components to use in further analyses.

Scree-plots display the number of components against the variance explained by each component, sorted in descending order of variance. Scree-plots help us get a better sense of which components explain a sufficient amount of variance in our data. When using scree plots, an 'elbow' (a steep drop from one data point to the next) in the plot is typically used to decide on an appropriate cutoff.

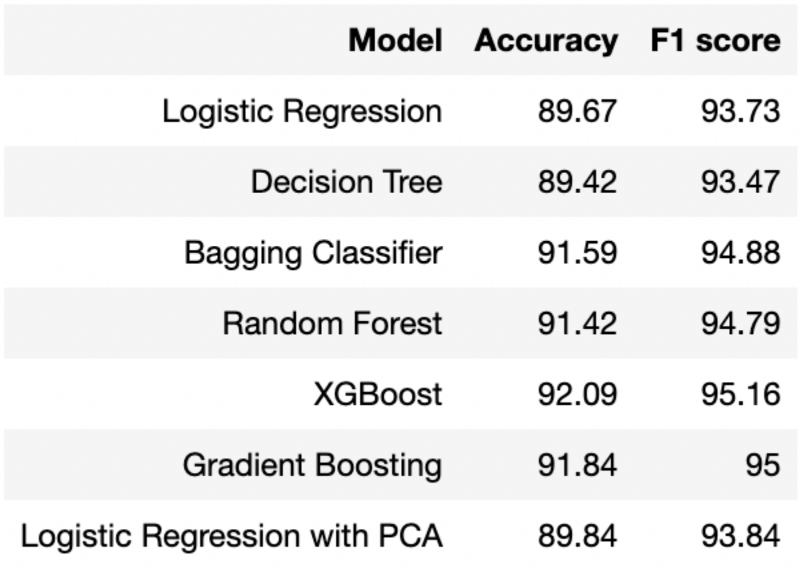
Unfortunately, there didn't appear to be a clear elbow in our scree plot, which means it is not straightforward to find the number of intrinsic dimensions using this method. Instead, we used the cumulative explained variance plot to determine how many features are required to explain, say, about 90% of the variance (cutoffs are somewhat arbitrary here, and usually decided upon by 'rules of thumb'). We determined the appropriate number of components, which we found to be 6. Later, we performed PCA with that many components, ideally reducing the dimensionality of our data.

0.6 Methodology

Now, we used the lower dimensional PCA projection of the data to classify songs into genres. To do that, we first split our dataset into 'train' and 'test' subsets,, where the 'train' subset will be used to train our model while the 'test' dataset allows for model performance validation. We split the main dataset into train and test subsets in the ratio of 0.75:0.25. At first, we used a simple algorithm known as a decision tree. Decision trees are rule-based classifiers that take in features and follow a 'tree structure' of binary decisions to ultimately classify a data point into one of two or more categories. In addition to being easy to both use and interpret, decision trees allow us to visualize the 'logic flowchart' that the model generates from the training data. We used the Gini impurity to measure the quality of a split. For a decision tree with a depth of 21, we got accuracy and F1 score of 0.875 and 0.923 respectively. We tried fine-tuning our decision tree using the early-stopping method. Upon studying our plot of accuracy score for different depths of the decision tree for training and test sets, we found that the optimal depth of the decision tree was four. Later, we trained our decision tree with that optimal depth and found a slight improvement in our accuracy and F1 score. Moreover, we also computed the plot of accuracy for different alpha for training and test sets and upon analysis found that the optimal alpha was 0.000887. Training a decision tree with this optimal alpha also yielded a slight improvement in accuracy and F1 score. Later, we used Bagging classifier, Random Forest , XGBoost, Gradient Booosting and Logistics Regression with PCA and without PCA to train our dataset on and recorded their respective accuracy and F1 score for comparison.

0.7 Experimental Results and Evaluation

We used accuracy and F1 score to evaluate our models.



Model	Accuracy	F1 score
Logistic Regression	89.67	93.73
Decision Tree	89.42	93.47
Bagging Classifier	91.59	94.88
Random Forest	91.42	94.79
XGBoost	92.09	95.16
Gradient Boosting	91.84	95
Logistic Regression with PCA	89.84	93.84

Figure 1: Accuracy and F1 score for various machine learning models

0.8 Conclusion

From our findings , we see XGBoost yields the best accuracy and F1 score, hence it can be deduced that it is the best model for our case study. On the other hand, We found that the Decision tree with a maximum depth of 21 gives the lowest accuracy and F1 score. In the future, we hope to further study the fine tuning of hyperparameters to produce accuracy and F1 score as close as possible to 0.99. Moreover balancing the dataset can also further improve our model's bias.

0.9 Code

Visit this link : Music Genre Classification to see the datasets we used and our codes.