Music Taste Analysis

Using Spotify

Compiled by:
Apurva Mulay
Fall 2019
CIS 563 - Introduction to Data Science

Instructor:
Dr. Reza Zafarani

Table of Contents

1. Introduction

It is rightly said that "Music is the bridge between abstract emotion and unspoken words". Music has become an integral part of everyone's life. People listen to a myriad of songs every week, every day or even different times of the day. In today's world, online music streaming services are dominating peoples' lives and also helping to gather user's interests. Thus, analyzing the music taste of users can help enrich their lives in various aspects.

In this work, I am predicting the likelihood of a user liking particular music based on features of the song. The dataset consists of songs from two different users including Indian music and western music. The question I am trying to answer here is - Which user has more probability of liking a song given features of that song?

2. Prior Work

The analysis of the music taste of users is being helpful in recommendation systems in various domains. The recommendations include not only for songs but also recommending friends on social media[4], food dishes in restaurants[3] and screen customizations for digital devices. Music taste also reveals the personality of users to some extent which can be further used to solve the cold start problem in recommendation systems[2]. The paper [1] studies the effects of music on brain activity, physiological response, and human-reported behavior. The songs from Spotify were analyzed and tagged as "happy" or "sad". One hundred participants who had not heard the songs were asked to listen and then take either fMRI scan or wear sensors and rate the intensity between 1 and 10. This data was fed to machine learning algorithms to determine the strongest feature of the predictor of responses. In the future, these models make a strong case to create highly evocative movie tracks or help patients with mental health challenges to stimulate specific parts of the brain. Thus, understanding music taste of users opens wide gates to solve complex problems.

3.Method

3.1 Extracting data from Spotify

The data about the artists, albums, and songs is extracted using the Spotify API available for developers. The python library Spotipy [libraries 4] is used which provides access to Spotify API. The dataset consists of songs of three artists along with the audio features of the song. The data for each artist is extracted separately in a CSV file as multiple API calls were causing a timeout. All the comma-separated files are then combined into single comma-separated file.

3.2 Data Preprocessing

The dataset consisted of 16 columns. However, only the 11 column features related to song tracks were considered. The features included 'acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo', 'valence', 'popularity'. Columns such as album, track-

_number, name, id, artist, and URI were dropped. The reason column artist was dropped is because it would be a categorical feature. The important point of categorial features is that it should be present in the train-set as well as test dataset. So, the model would fail if a new artist comes. As a result, the analysis considers music tracks irrespective of artists. A new column user was added into the dataset and was then labeled as 0 and 1. The label 0 is for the user with Indian songs and 1 with English songs.

3.3 Exploratory Data Analysis

Each feature was then analyzed using a box plot to figure out the distribution of data. This analysis revealed that features need to be normalized and hence normalization was carried out. Now, since features are in the proper format, the correlation matrix was created to understand the relationship between different features. Energy and loudness were highly correlated. There was also a correlation between valence and energy, danceability and valence, loudness and valence and energy and speechiness.

3.4 Training Set and Test Set

The code is implemented using Python and Scikit-learn. Two approaches are followed to understand the difference. The first approach consisted of splitting the data as a 70% train set and a 30% test set. The second approach used k-fold cross-validation with k=3. K-fold cross-validation gave better results. A point to note is that dataset was particularly made biased to understand the effect and how to fix it. The class_weight = 'balanced' parameter is set for all to fix the bias. The performance metrics showed slight improvement after providing 50-50 balanced dataset i.e 432 songs for user 1 and 419 songs for user0.

3.5 Model building

Three models were used for analysis - Logistic Regression, Decision Tree Classifier, and Random Forest. The concept of classification used here is <u>One-Versus-All multi-classification.</u>

The following sections will be discussing them.

1. Logistic Regression

Logistic regression uses a logistic function to model a binary dependent variable. Now, to understand the effect of the biased dataset, the dataset was particularly made biased to include 432 songs with label 0 and 223 songs with label 1. The logistic regression algorithm got biased towards label 0, resulting in predicting 0 every time. As a result, the model was tuned by setting the parameter of class_weight to 'balanced' which weighs classes inversely proportional to their frequency. Another parameter set is random_state to get the same output every time.

With train-test split of 70/30 respectively:

Before tuning: Accuracy = 0.65 and F1 score = 0

After tuning: Accuracy = 0.80 and F1 score = 0.74

With K-fold-cross-validation:

Before tuning: Accuracy = 0.69 and F1 score = 0

After tuning: Accuracy = 0.84 and F1 score = 0.77

2. Decision Tree Classifier

Decision tree is simple and widely used classification technique. The algorithm uses the Gini index to measure the impurity at the node. As a decision tree is also sensitive to biased data, it is tuned by providing a parameter of class_weight to 'balanced'. Interestingly, even if class weight is provided, it doesn't give warning, unlike logistic regression. However, F1 score decrease after tuning.

With train-test split of 70/30 respectively:

Before tuning: Accuracy = 0.88 and F1 score = 0.84

After tuning: Accuracy = 0.88 and F1 score = 0.82

With K-fold-cross-validation:

Before tuning: Accuracy = 0.84 and F1 score = 0.77

After tuning: Accuracy = 0.84 and F1 score = 0.75

3. Random Forest

Decision trees have low bias but more variance resulting in overfitting of data. To reduce this variance and overfitting, the random forest comes to rescue. Random forest generates many trees while training and outputs result which is the mode of classes. The bootstrap parameter is set to true by default, so, sampling is done with replacement. Again, the random_state parameter is set for output to be consistent.

The best part of random forest is that it gave nearly the same accuracy and F1score for cross-validation as well as with the train-test split without shuffling.

The K-fold cross-validation results will be considered for problem analysis with tuning.

4. Results and Findings

This section summarizes different model evaluation metrics and findings of the analysis done on data.

4.1 Model Performance Evaluation

This section represents metrics for model comparison and answers how one model was selected. The model with K-fold cross-validation with tuning is considered. The One-Versus-All multi-classification technique is used which means true positives and true negatives will be high.

The following table summarizes model evaluation metrics for all models.

Models	**Logistic Regression**	**Decision Tree**	**Random Forest for tuned class imbalance**	**Random Forest for class balance**
Confusion matrix	[[125 26] [9 58]]	[[133 18] [16 51]]	**[[143 7] [8 59]]**	**[[133 16] [9 125]]**
Accuracy	0.84	0.84	**0.93**	**0.91**
Precision	0.69	0.75	**0.89**	**0.88**
Recall	0.86	0.75	**0.88**	**0.93**
F1 Score	0.77	0.75	**0.88**	**0.90**
Area under ROC curve	0.85	0.82	**0.91**	**0.91**

As per Area under the ROC curve, the Random Forest model is selected.

Let's understand the confusion matrix in terms of the problem statement for class balance i.e 50-50 train-set for each class.

Predicted 0 Predicted 1

Actual 0 [[133 16] => [[True Negatives (TN) False Positives (FP)]

Actual 1 [9 125]] [False Negatives (FN) True Positives (TP)]]

0 = user0 and 1=user1

The numerical values on left are mapped to the corresponding terms on the right. Rows represent actual values and columns represent predicted values.
The row1 is class 0, row2 is class 1, column1 is class 0, column2 is class 1,
Note that, the total of elements in the above matrix is total instances in the test dataset i.e (163 + 9 + 16 + 125) = 313 rows.

Let’s interpret each value one by one.

1. **True Negatives** = 133 => This indicates that, for 133 songs, the prediction of which user likes which song is as expected in the test set. True Negatives means that 133 songs liked by user0 and model correctly predicted them.
2. **False Positives** = 16 => This indicates that, for 16 songs, the prediction of, which user likes which song is not as expected in the test set. False Positives means that 16 songs had label zero i.e liked by user1 but model incorrectly labeled them as song liked by user0.
3. **False Negatives** = 9 => This indicates that, for 9 songs, the prediction of which user likes which song is not as expected in the test set. False Negatives means that 9 songs had label 0 i.e liked by user1 but model incorrectly labeled them as song liked by user0.
4. **True Positives** = 125 => This indicates that, for 125 songs, the prediction of which user likes which song is as expected in the test set. True Positives means that 125 songs we labeled as liked by user1 and model correctly predicted them.

Let's also map precision and recall values to the problem statement.

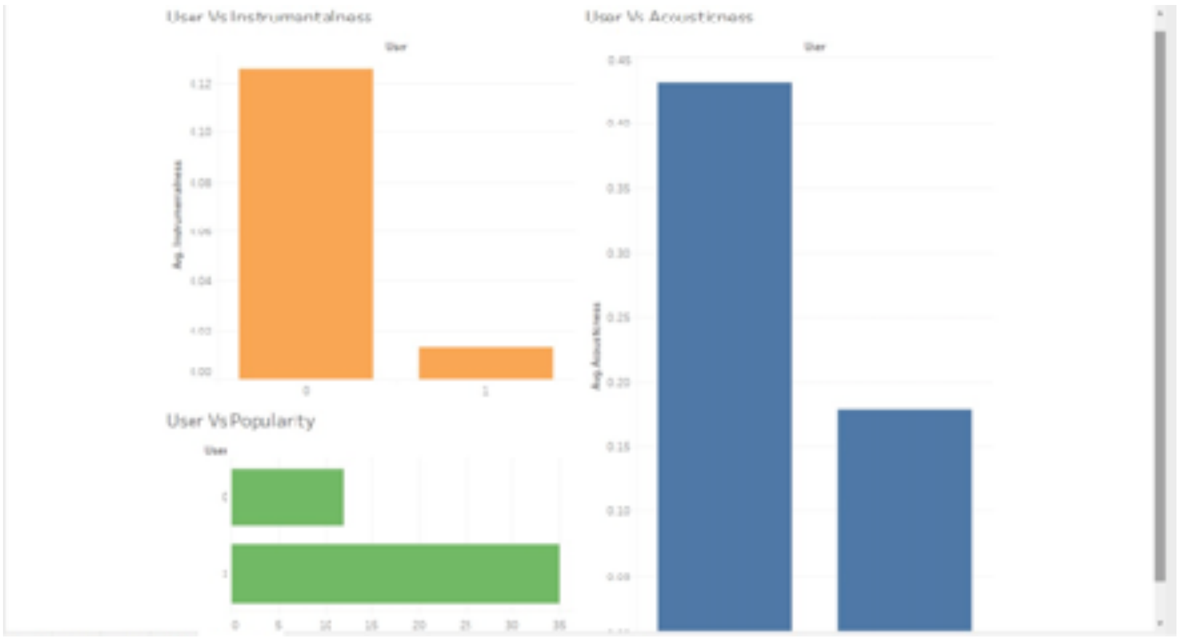
1. **Precision** = TP / (TP + FP) = 125/(125+16) = 125/141

As formula says, it considers all positive values. So, precision is out of all values model predicted as positive, how many were actually positive. So, the model predicted 141 songs as liked by user1, out of which 125 were actually liked by user1.

2. **Recall** = TP / (TP + FN) = 125/(125+9) = 125/134

The recall considers actual positive values. It measures that from actual positive values, how many the model predicted correctly. There were total of 134 songs liked by user1, model correctly recalled 125 of them.

4.2 Findings



The graph shows that user 0 likes songs with high instrumentalness and high accousticness while user 1 likes highly popular songs. The above graphs are prepared in Tableau.

Thus, given a highly popular song, the model would predict that it would be liked that user 1.

5.Conclusion

The data of the songs listened by two users is analyzed. The analysis is done to understand the music taste of the user. The features considered are the characteristics of the songs given by Spotify. The analysis showed that both users had different tastes. The user 0 liked more instrumental and acoustic songs while user1 liked popular songs. The Random Forest model is used to predict which user has more probability of liking a new song, given the features of the song. The F1 score is 90% and accuracy is 91% for class.

6. Libraries and resources:

1. Python was used for implementation
2. Scikit-Learn - machine learning toolkit for Python
3. Pandas for data manipulation and analysis
4. https://spotipy.readthedocs.io/en/latest/ for Spotify API library in Python
5. Matplotlib, Seaborn, and Tableau for data visualization

7.References

1. https://dl.acm.org/citation.cfm?id=3343031.3350867 - A Multimodal View into Music's Effect on Human Neural, Physiological, and Emotional Experience
2. https://dl.acm.org/citation.cfm?id=3347021 - The influence of personal values on music taste
3. https://dl.acm.org/citation.cfm?id=3007580 - Sound-enhanced gustatory experiences and technology
4. https://ieeexplore.ieee.org/document/6785749- Temporal influence over the Last.fm social network