Jacob Farrar

CS 484

11 December 2018

Using A Song’s Audio Features To Predict When I Would Listen To It

Abstract

Each song on Spotify has a set of audio features that describe specific attributes, such as the tempo, time signature, how likely it is to be acoustic, or how happy it is. Since I have also kept track of my listening habits on Spotify for over a year and a half, I wanted to see if I could use these features to predict the month, day of the week, or time of day I would listen to it.

To solve this problem, I matched each listen I had accumulated to its audio features and did some analysis to find which features had the most variance among each time block. I then used both a Decision Tree classifier and a K-Nearest Neighbor classifier to fit the data. I used 2/3s of my data for my training set and 1/3 of the data for validation.

In the end, I found that these methods worked best with the fewer amount of buckets to fit the data in, such as finding which season compared to individual months. I also found that K-Nearest Neighbor worked best for this problem, giving a higher accuracy compared to the Decision Tree, specifically when K = 7. Finally, this method gave the best accuracy when predicting whether I would listen on a weekday or a weekend. However, this accuracy may be misleading due to imbalance of the data.

Introduction

Sometimes, I want to listen to music with a certain kind of aspect to it. For example, I may want to listen to instrumental music, happy music, depressing music, etc. However, I have never paid attention to when these moods occur for me. Once I figured out that Spotify had a set of features to describe what kind of mood each song has, I wanted to see whether I could use that to predict when exactly these moods occur.

Specifically, my hypothesis was to see whether Spotify’s audio features alone could accurately predict what month, what day of the week, and what time of day I would listen to a song. While the main reason I wanted to try this was out of curiosity, I do believe there could be some practical uses for this. For one, I think that finding the answer to this problem could have a use in automatic playlist creation. If we can accurately predict the times one would listen to a song, a playlist could be created using similar values to match the time the user wants to listen; for example, an automatic playlist for a Saturday in November. Also, if this algorithm was done on a large scope of users, we could find out if there are any trends among average listeners on when they like to listen to certain music.

Solution

The main two algorithms I used for this project was SKLearn’s Decision Tree Classifier and K-Nearest Neighbor Classifier. I first wanted to use a Decision Tree because I felt that the model would help find trends in data that I would not be able to notice myself. For example, it might decipher whether or not listening to more instrumental music correlates with more energetic music, or more depressing music. After trying that, I also felt K-Nearest Neighbor would work because I believe that it would help classify times by finding those whose features are more similar to its own, and using those labels to predict the time. I used SKLearn’s basic classifier and their functions to fit the data and build the classifier.

Before building the data, I had to narrow down the features for each of these times. To do this, I put all my data in Excel and used their built-in functions to find the averages of each feature for each month, day, and hour. Once I had those, I picked the features that had the varied the most. Here’s one of the graphs I used as an example:

. The difference between the highest and lowest valence is 4%, which when compared to the other features, is significant enough to include.

In the end, the features I included were as follows:

Month: Valence, Energy, Liveness, Tempo, Acousticness, Danceability, Instrumentalness.

Day: Valence, Energy, Tempo, Acousticness, Instrumentalness.

Hour: Valence, Energy, Tempo, Acousticness, Danceability, Instrumentalness.

Data

My data consisted of 27,645 listens from Spotify that I have accumulated consistently from March 2017-October 2018 (and some other short periods since 2011), which I kept track of using Last.FM. This includes every time I have listened a particular song, so there are repeats for each time I listen to a song. At first, the information included the artist, album, song title, and the time and date I listened to each song. I used that information to then connect to Spotify and get each track’s audio features.

One of the main challenges I faced in this project was that I could not keep all of my listens, due to songs not being on Spotify’s database, character mistranslation when downloading the data from Last.FM, and problems with apostrophes when gathering the audio features. Because of this, I lost 6.5% of my data, leaving me with 25,385 listens. While this is significant due to the amount of data loss, I do not think this affected it too much, as I think the data I lost varied enough so that no feature or time was affected significantly more than the other.

The other challenge was that the data was noticeably imbalanced. I have only kept track of one month of listening from November through February, compared to two months the rest of the year. Because of that, those months were generally more skewed on its average features than the other months. Also, I listen to less music on weekends and almost never between midnight and 6am, also skewing the average features. This means I have to keep this in mind when viewing the results of the experiment to see if it could have an effect.

Experimental Setup

After finding the features, I made different files for hours, days of the weeks, and months, which included the time for each listen and its corresponding audio features. The data was presorted by song title before making these files, since I thought that it was the most balanced way of training the data. I used the first 2/3s of this data for training and left the last third for testing. I measure the how well the classification methods performed by using the score function each classifier has, which returns the mean accuracy the testing data performed. This is supposed to give a rather harsh metric, according to the documentation, but I still feel it gives a good estimate on how well the data performed.

Results

Months:

I did two tests for each metric: one predicting the individual months and one predicting the season.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Decision Tree | KNN:  K = 1 | KNN:  K = 3 | KNN:  K = 5 | KNN:  K = 7 | KNN:  K = 9 |
| Individual Month Accuracy | 0.0978 | 0.0977 | 0.088 | 0.0957 | 0.0972 | 0.0953 |
| Season Accuracy | 0.2917 | 0.275 | 0.2717 | 0.2954 | 0.3 | 0.3 |

Overall, audio features were not a good predictor for individual months. Assuming a random classification would give 1/12 accuracy (8.33%), these metrics only perform about 1% better, and barely differ between the classifiers. And while there was improvement when predicting seasons, it still only performed 4-5% better over a theoretical random classification of 1/4. KNN with a higher value of K performs better than the others by a small margin, but it’s still fairly insignificant.

Day of the week:

For these tests, I also did one for each individual day of the week, and one for predicting either weekday or weekend.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Decision Tree | KNN:  K = 1 | KNN:  K = 3 | KNN:  K = 5 | KNN:  K = 7 | KNN:  K = 9 |
| Individual Day: | 0.1598 | 0.1513 | 0.147 | 0.1448 | 0.1579 | 0.1587 |
| Weekday/  Weekend: | 0.7074 | 0.7 | 0.764 | 0.782 | 0.8 | 0.803 |

While the classifiers did not perform well for individual days, it performed much better than expected when predicting whether I listened to a song on a weekday or weekend. KNN with K = 7 and 9 also perform considerably better than lower Ks and the decision tree. However, it is very likely this is due to the imbalance of the data. I listen to music on weekdays 82% of the time, which is a greater percentage than the accuracy received. It turns out that it simply predicted I listened to a song on weekdays every single time, since a point was likely surrounded by other weekday points every time. Due to this imbalance, the accuracy is misleading and it is likely not a good metric for prediction.

Hour:

For these tests, I had one test that split the times into three-hour buckets (12PM-3PM,etc) and one for comparing the workday to outside the workday (i.e. 12AM-9AM, 9AM-6PM, 6PM-12AM). For the first test, I had to combine 12AM-9AM due to the lack of listens between 12AM and 6AM.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Decision Tree | KNN:  K = 1 | KNN:  K = 3 | KNN:  K = 5 | KNN:  K = 7 | KNN:  K = 9 |
| Three-Hour Buckets. | 0.1912 | 0.194 | 0.2072 | 0.1986 | 0.203 | 0.203 |
| Workday Buckets | 0.5044 | 0.481 | 0.511 | 0.541 | 0.558 | 0.564 |

Three-hour buckets also did not perform too well with these algorithms. However, the workday buckets had a majority accuracy, which was better than expected. Curiously, the accuracy actually went up past K = 9, peaking at around K = 17 with 0.6. However, once again, this could possibly be explained due to imbalance, as I listened during the workday times 62% of the time.

Conclusion

In the end, it turns out that audio features alone are not a good predictor for when I listen to music. For individual months, days, and three-hour time blocks, the accuracy when testing the models was very low, rarely coming close to 20% accuracy. And when combining months to seasons, the accuracy was still too low to consider my hypothesis correct. However, predicting between weekdays and weekends came out at 80% accuracy, which seems enough to consider this possible; in addition to predicting workday hours or before and after with >50% accuracy. Despite this, I hesitate to consider those tests a success, due to the large amount of imbalance with that data. Also, I believe that my listens simply vary too much to accurately predict the times.

If I were to attempt this again in the future, I believe I would try a more complex algorithm. Genre could be a good place to start, since I do prefer to listen to certain genres at certain times. Also, I would find a way to put more weight on songs I really enjoy, since all listens were treated equally while fitting the data. Overall, adding these and perhaps some other aspects could more accurately predict my time, while still considering the audio features.

References:

SKLearn’s KNeighborsClassifier: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier>

SKLearn’s DecisionTreeClassifier: <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier>

Spotify’s Audio Features: https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/