ORIE4741 Final Project - Genre Classification

Stephanie Zhou (sz244), Danny Yang (dzy4), James Chen (jzc8)

December 4, 2017

1 Objective

The objective of our project is to train a classifier for predicting the genre of a song given musical data about the song. We chose to use the Million Song Dataset[1], a 300GB dataset that contains machine-extracted (Echo Nest)[2] features on one million different songs. We start by exploring the structure of the data and any obvious underlying relationships between features and genre. Next, we reduce the dataset to a manageable size. Finally, we train and tune a variety of classifiers on the data and analyze the results.

In the process, we hope to understand the subtle differences that underlie different genres of music. We also hope that our work can be used as improvements for music recommendation engines such as Spotify or Pandora, where it is important to label the genres of songs in order to cater them to users' preferences.

2 Dataset

In addition to the unlabeled 300GB dataset, we had to choose a genre label dataset. We selected the Allmusic Top Genre Dataset (Top-MAGD) over other sets of genre labels for several reasons:

- This is the largest set of genre labels available, with 406,427 song genre labels
- This accounts for the 13 most commonly occurring genres, with the rarest label containing 4010 instances. This provides us enough examples of each genre to create a class balanced dataset that is still large enough for training a generalizable model.
- Each song only has a single genre label, simplifying our classification task. Other datasets had multiple genre labels per song.

3 Exploratory Analysis

We begin by investigating the class proportions of the dataset. Since the 300GB dataset is unlabeled, the best estimate we have of the true class proportion is to look at the class proportions in the genre dataset since these are the only knowns we have. The 406,427 MAGD labels are distributed as such:

Genre Name	Number of Tracks	Percentage of Total
Pop/Rock	238786	58.7%
Electronic	41075	10.1%
Rap	20939	5.2%
Jazz	17386	4.4%
Latin	17590	4.3%
RnB	14335	3.5%
International	14242	3.5%
Country	11772	2.9%
Reggae	6946	1.7%
Blues	6836	1.7%
Vocal	6195	1.5%
Folk	5865	1.4%
New Age	4010	1.0%

We immediately notice a significant class imbalance - the Pop/Rock genre accounts for 58.7% of the songs. Most of the genres only account for under 5% each. This class imbalance can result in an accuracy paradox when training the model. The model may decide that always predicting the maximum likelihood class, in this case, Pop Rock, is the best choice to do. We will discuss how we combat this problem later.

To run exploratory visualizations and analysis, we use a provided random 1% representative subset of the full 300GB dataset, which contains 10,000 songs. We analyze the class frequencies of this subset and they are similar to the frequencies above so we take the subset to be representative. Of these 10,000 songs, we only select those which have labels in the corresponding MAGD genre dataset. We are left with around 3000 songs.

We visualize some of the features by computing its mean and standard deviation for each genre and plotting the standard normal distribution using the computed statistic. The graph below is an example, it is a plot for tempo across all the genre.

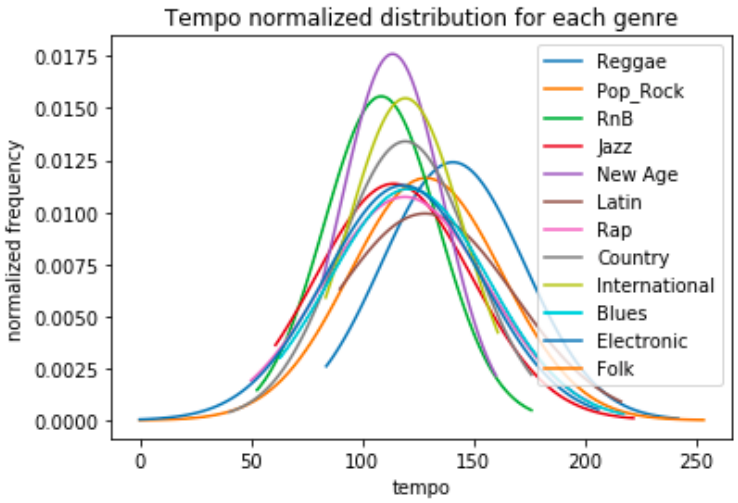

Figure 1: tempo normalized distribution

First, we wanted to understand each feature's relation to the response variable. We use the Pearson correlation coefficient, which measures the linear correlation between two variables. We discover that loudness is most directly correlated with the genre, with a correlation coefficient of 0.342. We also investigate the correlation between features. Almost all of the features have correlations below 0.1 between each other, implying that they are relatively independent with respect to the genre.

In terms of data cleanliness, there are not many N/A values. We simply replace any N/A's with 0's since all of the features are numerical. We calculated summary statistics for all of the numerical features as well. The results are not too meaningful, but we notice that there are interesting differences between the distribution for tempo, which might be a useful feature during training. In Figure 1, we plot the normalized tempo vs.

4 Feature Selection and Preprocessing

The Million Song Dataset contains 53 features, 21 of which are 1D or 2D array valued and 32 of which are single field int, float, or string typed. The full list of features can be found on the MSD website. One challenge was how to reduce the size of the dataset. Of the 1 million songs, approximately 400k songs had labels and could be used.

However, this would result in a 120GB dataset, which is still unusably large. Upon analysis of which features took the most memory, it was discovered that string fields and array fields accounted for nearly 98% of the memory usage after being loaded in a dataframe. This was not surprising because the array features consisted of hundreds of elements. Strings are also more expensive to encode than integers. We deal with both the single field data and array data separately to solve this problem.

4.1 Single Field Data

Of the single field data, we discovered that many fields could not be used as features. For example, we could not use any artist related fields such as artist name, artist id, and artist location. This is because it is likely that an artist will only write music of one genre. It is conceivable a model trained with artist data would overfit on the artist name as a strong indication of the song genre. If the model was presented a song with an unseen artist, it would not perform well. We also removed many uninterpretable or useless features. These were mostly Echo Nest generated features such as song hotness or danceablity because they are subjective and the details of how they were generated are unclear.

We were left with these 10 numerical features.

Feature	**Description**
duration	The duration of a track in seconds
end_of_fade_in	The end of the fade-in introduction to a track in seconds
key	The estimated overall key of a track. The key identifies the tonic triad, the chord, major or minor, which represents the final point of rest of a piece
key_confidence	The confidence of the key estimation, a float between 0 and 1.
loudness	The overall loudness in decibels (dB), averaged across the entire track
mode	Indicates the modality (major or minor) of a track
mode_confidence	Confidence of the mode estimation.
tempo	The overall estimated tempo of a track in BPM. Tempo is the speed or pace of a given piece and derives directly from the average beat duration.
time_signature	An estimated overall time signature of a track. The time signature is a notational convention to specify how many beats are in each bar.
start_of_fade_out	Start time of the fade out, in seconds, at the end of the song.

The key of the song is originally encoded as a numerical categorical number from 1 to 12. This will not be interpreted properly by a model. In fact, the model will learn a natural ordering to the key, which is not the case, since the key is purely categorical. Therefore, we one hot encode the key as a 12 dimensional vector and add it as 12 more feature columns.

4.2 Array Data

These features were difficult to analyze since the length of the vectors varied from song to song. For example, if a song had 100 bars then the bars_start feature would be a vector with 100 entries, but other songs have different numbers of bars. In the end, we decided to analyze only the summary statistics (mean and standard deviation) of each vector feature. This helped us dramatically reduce the size of the dataset by reducing vectors with hundreds of entries into two scalar values.

Of the original dataset, we kept the summary statistics for the following 6 features.

Feature	**Description**
bars_start	Start time of each bar (segment of time defined as a given number of beats)
beats_start	Start time of each beat (basic time unit of a piece of music)
sections_start	Start time of each section (Sections are defined by large variations in rhythm or timbre, e.g. chorus, verse, bridge, guitar solo, etc.)
segments_start	Start time of each segment (a set of sound entities, typically under a second, each relatively uniform in timbre and harmony)
segments_loudness_max	Peak loudness value within the segment
tatums_start	Start time of each tatum (the lowest regular pulse train that a listener intuitively infers from the timing of segments)

For each of the 6 features, we also create 6 more features for the standard deviation for each.

4.3 Final Dataset

To create our final dataset, we take 4000 songs from each genre. We do so in order to create a perfectly class balanced dataset of 52,000 training points and labels. This is done to combat the class imbalance. With a class imbalance, it is likely that any model trained on the data will just classify any input song as having the most frequent genre label. After feature creation, our feature space is 33 dimensions, yielding a 52000×33 matrix. Naturally, we also obtain a 52000×1 label matrix.

5 Models

5.1 Multi-class Classification

We first reduced the dimensionality from 33 to 5 after performing PCA on the normalized data. We chose 5 dimensions because the explained variance ratio for 5 dimensions is already 0.91, so adding dimensions would not contribute much. The graph on the right shows the data points after PCA in 2D by picking the top 2 dimensions with the color representing the label. It shows that our data does not form any distinguishable cluster in 2D space. Nevertheless, we tried two methods of unsupervised learning, K-nearest-neighbors and K-means to perform multi-class classification.

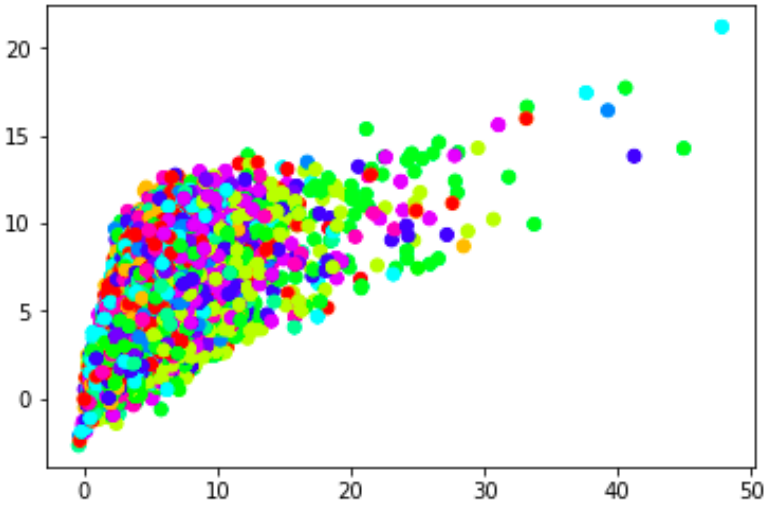

Figure 2: PCA in 2D with genres color coded

5.1.1 K-Means

For K-means, we first cluster the data into 13 clusters since we have 13 different classes. We split the data into training and testing set, took the majority label of the points in the training set for each cluster and assign that label to the the cluster as our classification result. For each point in the test set, we used the majority label of the cluster as its genre prediction.

5.1.2 K-Nearest Neighbors

For K-nearest neighbors, we picked k=10. For each point in test set, we took the majority label for its 10 nearest neighbors as its predicted genre.

Both methods were unsuccessful, resulting in a less than 10% accuracy for both methods, compared with a baseline of 7.7%. The accuracy with and without PCA yielded similar result. This gave valuable insight into the structure of our data, and showed that the classes were not organized in clusters. We decided to approach the problem from a different direction.

5.2 Binary Classification

Instead of using a one vs. all classifier, which is sometimes done in the multiclass classification setting, we train a 13 binary classifiers, one for each genre label. For a given song, we evaluate the probability of the song belonging to a genre. We repeat this 13 times for all genre specific classifiers and return the genre that has the highest probability. This is why we partitioned our data into a perfectly class balanced dataset. For each genre model, we take all 4000 songs from the genre and 4000 random songs from other genres. We convert the labels of theses latter 4000 songs to the same label. So if the genre was 'Rap', the other 4000 songs would be labeled 'Non_Rap'. Therefore we have a binary classification problem. Of the 8000 data points, we use 6400 for training and 1600 for testing (4:1 split). We perform 5-fold cross validation during model training. Using this split, we train 7 different models for this binary classification problem and analyze their performance.

5.2.1 Linear Regression

To convert this to a regression typed classification problem, for each genre, we set the label for to be 1 for that genre and -1 for everything else. We fit a linear regression model on it using `scikit-learn`. If the model outputs a positive number for a test point, we interpret the point as having the genre. If the model outputs a negative number, it predicts the point as not belonging to the genre. The mean square error was around 0.7 for each class. Normalizing the columns had minimal effect on the validation accuracy. Therefore, we decided to to normalized the columns which enabled us to interpret the coefficients in future analysis.

An interesting observation is that the result of prediction is worse after performing PCA on the data. The more dimensions we keep, the better the performance. This is an indication that if we might want to retain more features and that our model is suffering from a lack of expressive features.

5.2.2 Huber Regression

We also performed Huber regression on our data because it is less sensitive to outliers and we do expect our data to contain mislabels and errors since the features are extracted automatically by Echo Nest. Huber regression is an intermediate between Lasso and Ridge regression, being differentiable around 0 but also linearly increasing with larger losses. The results produced by Huber regression yield very similar results. One explanation for that is a more robust regression model would assume that the data points are drawn from a straight line, but our underlying model might not be linear in the first place, or that the benefit of Huber regression is not significant on our dataset.

5.2.3 Support Vector Machine

We also train support vector machines (SVMs) on the data. We do so because we believe that data from one genre can be separated from data from not that genre in high dimensional space. We train the SVM with both linear kernels and radial basis function kernels. Using the linear kernel means we assume a linear (with respect to the weight vector) hyperplane that can separate the two classes. The radial basis function (RBF) kernel can be used to create a non-linear hyperplane that can define curved boundaries around points. We train both linear and RBF kerneled SVMs using 5-fold cross validation. We also perform hyperparameter tuning with C = {0.01, 0.03, 0.1, 0.3, 1, 3} and gamma = {0.01, 0.03, 0.1, 0.3, 1, 3, 10}. We find that for the linear model, the best parameter of C is 1. For the RBF model, the best parameter of C is 0.3 and the best parameter of Gamma is The C parameter is a "softness" value that represents how much we penalize incorrectly classified points. For larger values of C, the SVM will choose a smaller margin hyperplane but attempt to correctly more points correctly. For smaller values of C, the SVM will choose a larger margin hyperplane at the expense of misclassifying a few points. A large value of C will result in the SVM trying to overfit the training data. Intuitively, a smaller gamma value will define a Gaussian function with a larger variance. Two points can be considered similar even if they are far away in space. A larger gamma value will define a Gaussian function with a smaller variance - two points are considered similar even if they are close to one another.

5.2.4 Logistic Regression

For this method we trained a binary classifier to predict each genre using a logistic loss function and L1 regularization, with the maximum number number of iterations capped at 100. We experimented with different numbers of iterations and the algorithm converged within 100 iterations for all genres. For most genres, L1 and L2 regularizers had similar (less than 0.5% difference in accuracy) performance, but our logistic regression model with L1 regularization significantly outperformed L2 regularization in classifying Country (1.3%), Vocal (1.4%), and Electronic music (2.4%).

5.2.5 Decision Tree

We trained a single decision tree binary classifier for each genre, limiting the maximum depth to 6 layers. We selected the number of layers by testing trees with 2-15 layers, and found that accuracy was highest for trees with 5-7 layers but dropped off after 7. Limiting the depth of the tree prevents overfitting.

5.2.6 Random Forest

We trained a random forest binary classifier for each genre, using 100 trees. We arrived at that number by testing out random forest with different numbers of trees, and found that there were no performance improvements after 80-90 trees on any genre. Random forests work by combining a large number of complex models with high variance and low bias. Aggregating these complex models reduces variance, resulting in a model with both low variance and low bias.

5.2.7 Convolutional Neural Network (CNN)

We also attempted running our data though a convolutional neural network. We transformed our data into a 2D input. A CNN is a network that contains convolutional layers, which compute the cross correlation between the filter(weight matrix) and the input in each step. One advantage of a neural network is it can easily capture non-linear relationships in data without the need for much feature engineering. In the final step, the neural network essentially performs a linear classification. The disadvantage of the neural network is that it is almost a black-box, so we do not have a good way to analyze how it classifies the data.

CNNs would have been more suitable for the task if we included the pitch features since CNN takes advantage of the local connections of the 2D input. The reason that we decided to run our data on a neural network is that neural networks tend to have better performance at capturing complex non-linear relationships. They also overfit easily so we used it to act as a sanity check to make sure that our other models did take full advantage of the data.

We tried several different learning rates, batch sizes and filter sizes for different genres. We decided on a learning rate of 0.001 and a batch size of 5 which perform the best on the validation set. Our network consists of 2 convolutional layers, since our number of features are rather small. Adding more layer would increase the training difficulty and make the model prone to overfitting. We used a ReLu (f(x)=max(0,x)) for the non-linear activation layer after each convolutional layer. It's standard for CNNs and does not suffer from the vanishing gradient problem.

We trained our model for 10 epochs since the loss did not decrease much after 10 epochs. Judging from the results, our model requires further tuning. The reason that it performed poorly is that it may have overfit on the noise of the data. It would have also been useful to tune different parameters for different genres. Since the neural net did not give a significant increase in performance, it showed that our other models sufficiently captured the complexity of the data and that increasing the performance would probably require more features.

6 Model Evaluation

The highest performing model (averaged across all genres) was logistic regression, although simple linear regression and random forest were close. Logistic regression either had the highest accuracy or was within 0.5% of the highest accuracy model for 10 of the 13 genres. As expected, random forest outperformed the single decision tree in the

majority of genres (9 of 13). Both random forest and decision tree methods performed better on the larger stratified dataset compared to the dataset we used during exploratory analysis due to more available features enabling a better fit to the data; notably, they performed significantly better than other models when classifying Electronic and Folk music.

The false positive and false negative rates varied across different models and genres, although all models tended to have higher false negative than false positive rates when trained to classify Folk, Country, or Blues music. For our highest performing models, the false positive and false negative rates were relatively similar to each other (around 10% of predictions were false negatives and another 10% were false positives).

The degree of success for which our models were able to classify the genre varied greatly across the different genres. We think that this is because some genres labels are very broad and have a large variation in the musical qualities we measured. The genres that we had the most success with classifying were Rap, Reggae, and New Age (with 79-80% accuracy on our highest performing models), suggesting that these genres are more narrowly defined and relatively distinctive. We had moderate success with Country, Electronic, and Vocal (with accuracies in the mid-70's). In contrast, we had the least success with the International, with no model performing better than 60%; suggests that International music is a very broad genre with a lot of variation and the features we selected do not account for all of the variation within this genre.

7 Results

For each genre, we trained every model on a set of 8,000 songs (4,000 from that genre, 4,000 from other genres) selected from the set of 52,000. So the baseline of the task is 50%. We should expect to do no worse than random guessin. The results below are the accuracy on the test set (size 1,600), with the highest-performing model in for each genre highlighted. The accuracy is just the fraction of correctly classified tracks, calculated with $\frac{TP+TN}{FP+FN+TP+TN}$ and expressed as a percentage in the table. T/F stands for True/False and P/N stands for Positive/Negative.

	Linear Regression	Huber Regression	Logistic Regression	SVM Linear	SVM RBF	Decision Tree	Random Forest	CNN
Pop_Rock	72.3%	73.8%	73.5%	70.62%	65.4%	67.9%	64.6%	68.7%
Electronic	70.3%	67.1%	73.3%	63.8%	68%	73%	74.1%	68.0%
Rap	76.3%	75.2%	77.3%	53.6%	65.3%	72.9%	76.5%	74.3%
Blues	64%	60.3%	67.4%	56.5%	59%	64.1%	64.7%	54.5%
Country	72.5%	71.8%	75.4%	73.5%	73.6%	72.1%	72.7%	66..8%
Jazz	66.9%	65.2%	69.7%	62.6%	70.9%	70.2%	68%	61.3%
RnB	65.3%	63.0%	65.4%	60.1%	62.6%	63.6%	63.7%	60.5%
Reggae	79%	77.9%	79.3%	75.3%	73.8%	73%	75.1%	74.8%
Latin	64.2%	61.6%	66.6%	55.3%	58.6%	64.4%	64.2%	64.3%
New Age	79.1%	77.4%	79.8%	77%	73.8%	76.5%	78.9%	70.3%
International	55.0%	57.2%	57.0%	54.5%	53.8%	53.4%	56%	52.3%
Folk	65.8%	65.2%	67.5%	59.2%	64.2%	68%	68.3%	62.9%
Vocal	74.5%	70%	76.8%	72.6%	74.6%	75.5%	75.5%	66.3%
Average	69.6%	68.1%	71.5%	64.2%	66.4%	68.8%	69.4%	65.4%

Figure 3: Accuracy results for all models with highest performing models highlighted

In addition, we tested the accuracy of some of the classifiers trained using the balanced training sets on the labeled part of the 1% subset (size 3,600) to see how it would perform on a set with genre distributions that were more unbalanced and more closely matched the full dataset. Accuracy scores were within 2% of the values we got on our other test set for logistic regression and random forest, but the performance of the decision tree classifier had a sharp drop, scoring as low as 15% less than on the balanced test set.

8 Interpretation

The highest performing model (averaged across all genres) was logistic regression, although simple linear regression and random forest were close. Logistic regression either had the highest accuracy or was within 0.5% of the highest accuracy model for 10 of the 13 genres. As expected, random forest outperformed the single decision tree in the majority of genres (9 of 13). Both random forest and decision tree methods performed better on the larger stratified dataset compared to the dataset we used during exploratory analysis due to more available features enabling a better fit to the data; notably, they performed significantly better than other models when classifying Electronic and Folk music.

The false positive and false negative rates varied across different models and genres, although all models tended to have higher false negative than false positive rates when trained to classify Folk, Country, or Blues music. For our highest performing models, the false positive and false negative rates were relatively similar to each other (around 10% of predictions were false negatives and another 10% were false positives).

The degree of success for which our models were able to classify the genre varied greatly across the different genres. We think that this is because some genres labels are very broad and have a large variation in the musical qualities we measured. The genres that we had the most success with classifying were Rap, Reggae, and New Age (with 79-80% accuracy on our highest performing models), suggesting that these genres are more narrowly defined and relatively distinctive. We had moderate success with Country, Electronic, and Vocal (with accuracies in the mid-70's). In contrast, we had the least success with the International, with no model performing better than 60%; suggests that International music is a very broad genre with a lot of variation and the features we selected do not account for all of the variation within this genre.

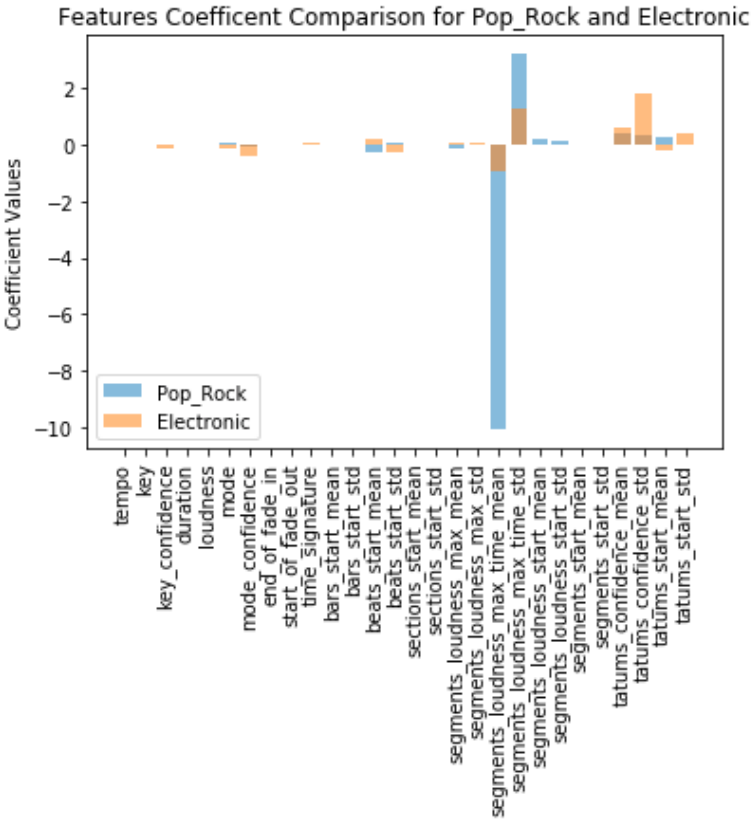

Figure 4: Feature Coefficient bar graph

We looked at the distribution of the feature distribution from the linear regression model with features normalized. Here is an interesting example that we picked out.

We observe that loudness mean, which is the mean loudness of each segment and its standard deviation is the main feature that help us distinguish pop rock from non-pop rock. Note that the loudness in this dataset are all negative value due to the log scale. We can expect pop-rock to be louder than other music on average.

In contrast, the coefficient for electronic is more evenly distributed. Tempo and key also have very little effect. The

coefficients tell us which of the features distinguishes the type of music from others.

9 Conclusion

In the end, our most important finding was that some genres are more musically distinctive and easier to identify, while others are broader and have more variance. Our highest-performing models failed to perform significantly better than baseline when evaluated against a purely random sample selected from the entire dataset, due to the high false-positive rate when classifying the less common genres, and the uneven distribution of labels in the original dataset (recall that 10 of the 13 genre labels occur with less than 5% frequency). Another insight we gained is that with only information about loudness, duration, standard deviation of loudness, we were able to predict the whether a song is of a particular genre fairly, which is a very light-weight solution. But to achieve higher accuracy, we need a more complex model that accounts more features from the original dataset.

There are several areas that can be explored in the future to further improve our models. Since our error is still higher than we want for both training and test sets, we think that our models may be underfitting the data. To solve this, we can increase the complexity of our models by increasing the number of features. There were some features in the original dataset and other compatible datasets that we did not make use of, including data on pitch and timbre (these features are formatted as large 2D arrays with different length for each song, and we think adding that might make our dataset too cumbersome). If we can find ways to transform this data and our other vector formatted features to preserve the structure instead of only using summary statistics, then we should be able to build models that more closely fit the data and classify these genres more accurately.

Our model is not ready for commercial use yet since our classification accuracy is not high enough, but in the future one practical application is for a music services to automatically classify the genre of music without the need for someone to manually label each track.

References

[1] `https://labrosa.ee.columbia.edu/millionsong/`

[2] `http://docs.echonest.com.s3-website-us-east-1.amazonaws.com/_static/AnalyzeDocumentation.pdf`