Music Analysis using Machine Learning

Gopalika Sharma

Problem Description

What is it that attracts a person to a song? Music is an essential part of our lives, and one of the most accessible and respected forms of art in our society, but truly understanding what aspects contribute to our enjoyment is a continuing process. As the global music industry continues to grow in revenue—in part from the rise of streaming services—and provide us with new ways to explore emerging artists, learning more about how we interact with music is becoming more important than ever to remain relevant and successful in the field.[1] With many available options, creating an enjoyable and memorable experience is how to stand out.

With this in mind, for our project, we were attempting to improve a user's experience in music selection through classification of song *genre, mood, and popularity.* When a user listens to a track or builds a playlist, they may want to categorize their songs with respect to an intersection of these three mentioned aspects, and our models could help in producing an outcome playlist or sorted tracks. Creating a better understanding of what factors contribute to people's enjoyment of music could help in designing more effective recommendation systems. All three tasks mentioned above can be considered classification problems.

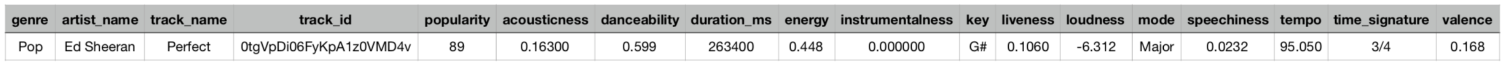
The *source code* for this project can be found here:
https://drive.google.com/open?id=1EIxyfErZWkswz5HuohVPwRRm9N-xKSac

The *presentation slides* can be found here:
https://docs.google.com/presentation/d/1aV7chmGsPFuzKjuQw3xDRRUCtva248774WsMK8Skgmc/edit?usp=sharing

Dataset

Note: We originally wanted to utilize the Million Song dataset for our tasks, but through preliminary exploratory data analysis, we found a lot of missing data in both the 10,000 subset and the full dataset; in particular for target variables, either more than half of the records had missing values (i.e. song_hotttnesss), or there was no real way to measure the variable at all with the data (i.e. mood). As a result we decided to switch to the Spotify Tracks DB dataset, which contains audio features and metadata for tracks taken from the Spotify API. Although this dataset has fewer features, it is a much more complete dataset, particularly for our prediction variables.

The Spotify dataset contains 232,725 records (songs) with 18 variables. Since we have three tasks, the response variables include popularity, genre, and valence, and may require some transformation to run classification models (see *Preprocessing and Feature Engineering* section). Feature types include audio information (such as energy, danceability, loudness) and song/artist metadata (including artist_name, genre, track_name) for an individual track. There are both continuous features (duration_ms, tempo) and categorical features (key, track_id) in the dataset. When examining the dataset, we did not find any explicitly missing or null data (although while 0 could indicate perhaps missing for continuous data, it is also a valid measure). We provide a sample song entry below:
Feature Definitions



genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence
Pop	Ed Sheeran	Perfect	0tgVpDi06FyKpA1z0VMD4v	89	0.16300	0.599	263400	0.448	0.000000	G#	0.1060	-6.312	Major	0.0232	95.050	3/4	0.168

We include basic definitions for a subset of the features in the dataset, including our potential response variables: genre, popularity, valence. The definitions are based on those provided from the Spotify API.[2] We visually explore some of these variables in the next section.

[1] https://fortune.com/longform/spotify-music-industry-profits-apple-amazon/

[2] https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/, https://developer.spotify.com/documentation/web-api/reference/tracks/get-track/

- **genre** - the track's associated song genre; this measure has 26 classes (i.e. R&B, Pop, Rock, etc.)
- **popularity** - the popularity of the track, between 0 and 100; continuous measure partly based on the total number of plays of the track
- **valence** - the musical positiveness of a track, between 0.0 and 1.0; high valence sounds are more positive (e.g. happy, cheerful) while low values are more negative (e.g. sad, depressed)
- **duration_ms** - the duration of the track in milliseconds
- **loudness** - the overall track loudness; loudness is averaged across the entire track (correlates with the amplitude) and usually ranges between -60 and 0 dB (decibels)
- **key** - overall key of the track; this measure has 12 classes including: C, C#, F, etc.
- **energy** - the energy level of each track; continuous measure based on the tempo, loudness and accoustiness of a track

Exploratory Data Analysis (EDA)

We first looked into the relationships between features using a correlation matrix (Figure 1). We observed that the most significant relationship is between loudness and energy, with a positive linear relationship of 0.82; this is understandable considering that energy is in part based off of the track loudness. Additionally we saw that energy has a strong negative correlation with acousticness (another feature that contributes to energy), of -0.73. When looking at our response variables that are originally in continuous form, we found that the strongest correlations with our popularity measure are -0.38 with acousticness and loudness with 0.36. For valence (our proxy for mood), we saw that the strongest relationships are positive correlations between danceability and energy, which are 0.55 and 0.44, respectively.

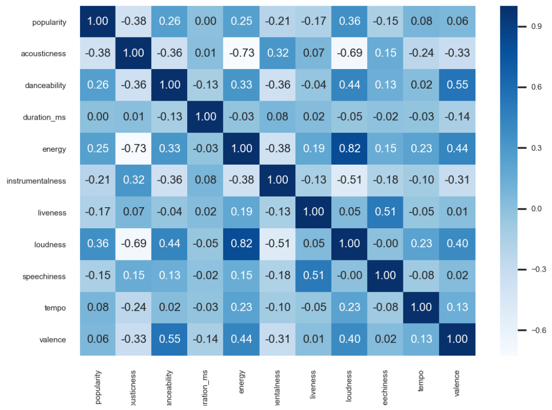
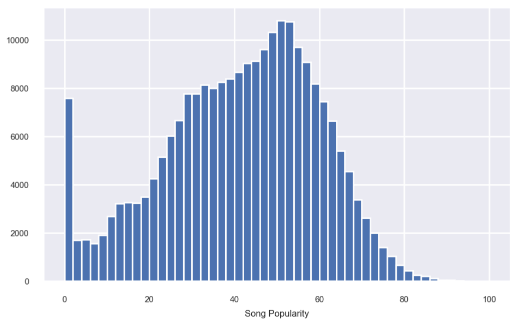


Figure 1: Correlation Heatmap

We found that the distribution of the popularity measure almost looks like a normal distribution, with the majority of values being located in the middle of the distribution, at around 50, and not much skew (Figure 2). However, we also found that there is a high count of values with scores of 0 for popularity. We further explored popularity by looking at the distribution among the 26 genres included in the dataset, and we saw that the genre with the highest median popularity is Pop, and the Movie and A capella categories have the lowest median values (Figure 3). Additionally, the genre with the largest interquartile range of popularity values is Children's Music, and the categories with the largest overall ranges in scores appear to be Pop and Dance.



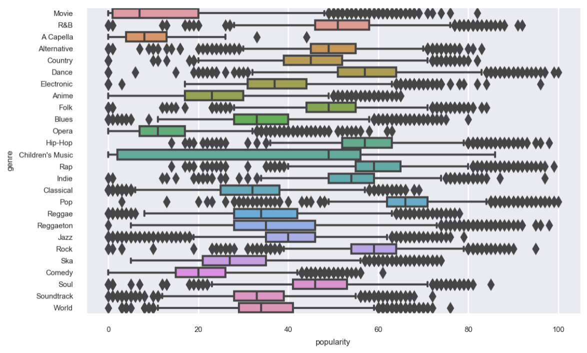


Figure 2: Song Popularity Histogram **Figure 3: Song Popularity by Genre Boxplots**

We also created scatterplots to explore popularity against continuous features (Figure 4). Popularity and loudness appear to have a positive relationship, confirming what we observed in our correlation results above. When looking at song duration, there does not really seem to be a significant linear relationship; it appears that most songs regardless of popularity, are below 2000000 milliseconds. Although there are a lot of data points making it hard to interpret the plots, it seems that song popularity and danceability may also have a general positive relationship.

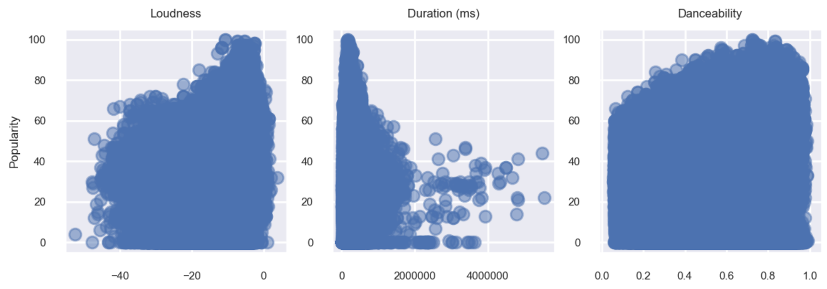


Figure 4: Scatterplots of Popularity vs. Loudness, Duration, and Danceability

In exploring valence against genre, we saw that Ska and Reggaeton have the highest median valence values, both close to 0.8 (Figure 5). Perhaps surprisingly, Classical music has the lowest median valence scores. All genres generally seem to have pretty variable representations of valence (both positive and negative tracks), based on the interquartile ranges.

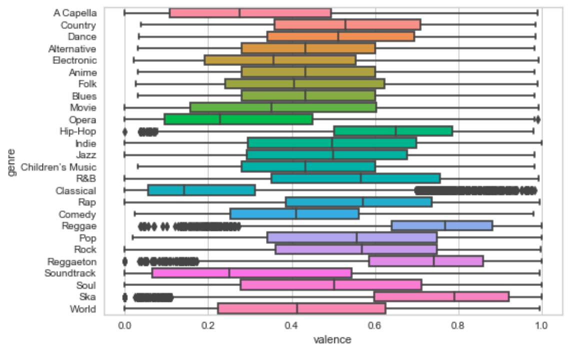


Figure 5: Song Valence by Genre Boxplots

We also examined additional feature relationships between potential model predictors (Figure 6). It seems that danceability and loudness have a somewhat positive relationship, but there is a set of tracks with 0 danceability regardless of the loudness. When looking at tempo and energy, we did not observe any relationship between the features.

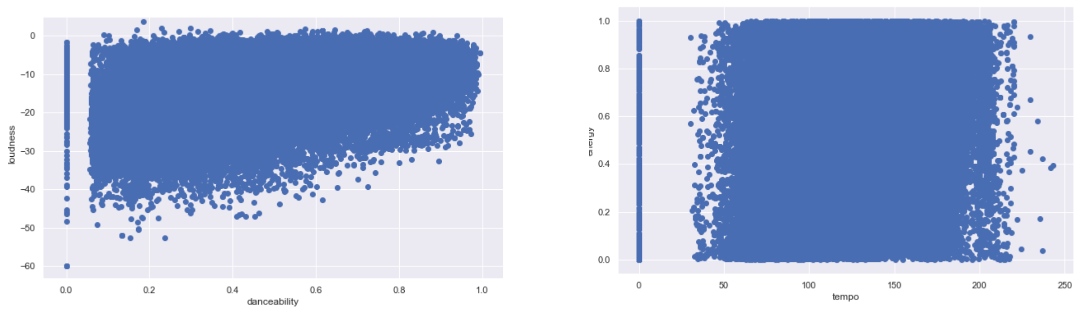


Figure 6: Scatterplots of Loudness vs. Danceability, Energy vs. Tempo

Finally we further explored our genre variable (Figure 7). Although the dataset contains 26 genre classes, we reduced down these categories and explored the data for 10 general categories (discussed more in the *Preprocessing and Feature Engineering* section). When looking at song duration, we found that Electronic music has some of the longest tracks (close to 5000 seconds), followed by some Jazz songs. However on average

Classical music have the longest songs. We also created a scatterplot to further break down the relationship between danceability and popularity by genre (Figure 8). While there is no clear pattern, we observe a slight increasing trend showing that Pop and Rap are both the most popular and most danceable genres. R&B sits in the middle as a genre that is not very popular but can have different degrees of danceability.

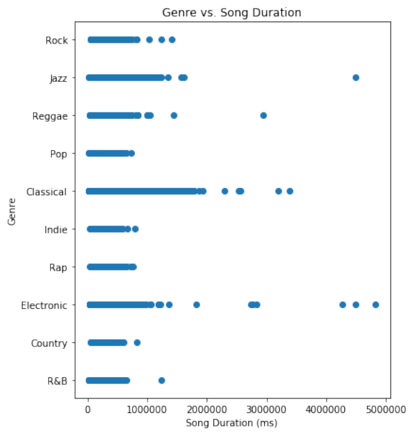


Figure 7: Scatterplot of Song Duration vs. Genre

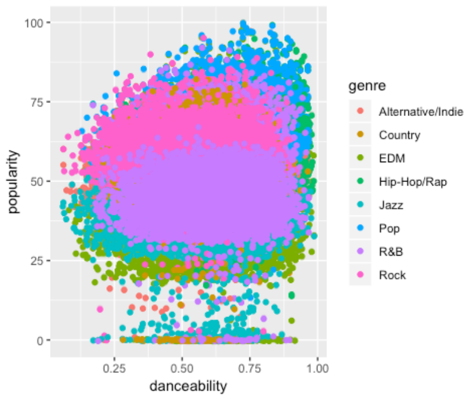


Figure 8: Scatterplot of Danceability vs. Popularity vs. Genre

Modeling

Preprocessing and Feature Engineering

We first conducted some general preprocessing that was applied to all three modeling tasks. Although many of our continuous variables are between 0 and 1, we decided to standardize our feature data before creating models to ensure features on different scales do not have differing influences. To include categorical features in our models (i.e. genre, key, etc.) we also applied one-hot encoding to these features.

Both the popularity and valence features in the dataset our continuous measures. However, since we are attempting to fit classifiers, we must convert these variables into categorical features. We created binary measures for both of these features, using the midpoint value as the cutoff threshold.[3] We made a popularity binary metric (indicating that the track is popular or not) using a threshold of 50, so songs with scores higher than 50 are indicated as popular. Similarly, for valence we created a binary variable (i.e. general mood measure) with a 0.5 threshold, where higher values are marked as positive mood (or valence) and lower values are assigned negative mood.

The genre feature has a large number of categories, which increases the difficulty of the classification task, so we attempted to simplify the feature dimension. Before training a model to classify song genre, we shortened the space of possible genres down from 26 to 10, to mainly account for the genres people most think of, as well as genres that are pretty distinct between each other. These 10 genres were Pop, Rap, Rock, Electronic, Reggae, Indie, Jazz, Classical, Country and R&B. We originally tried combining some similar genres (Soundtrack and Movie, Rap and Hip/Hop, Electronic and Dance, etc.) while leaving other genres (Pop, Country, etc.) but this created uneven distributions. All the genres had similar frequencies in the data (around 9000 records each) so combining some genres introduces bias for our classification tasks, as the models will favor the class that has more data. As a result we only reduced data, but did not combine.

Finally, for our individual classification tasks and specific models, we experimented with feature selection methods including filtering and regularization (we discuss implemented methods in the *Machine Learning Models* section). Additionally, we used k-fold cross validation to assess model performance and for some hyperparameter tuning.

Machine Learning Models

<u>Popularity Classification Task</u>

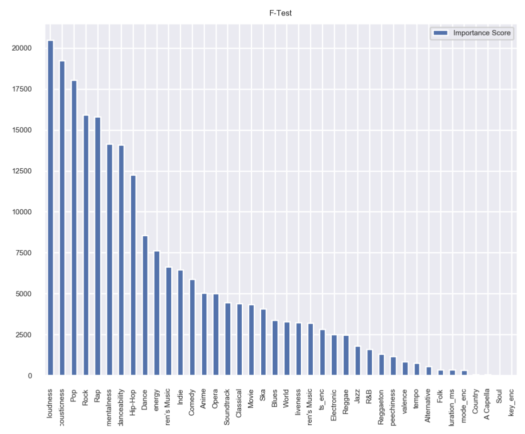
[3] https://www.researchgate.net/post/How_can_one_set_a_reference_point_cut-off_to_convert_quantitative_variables_into_binary_variables_to_be_used_in_logistic_regression

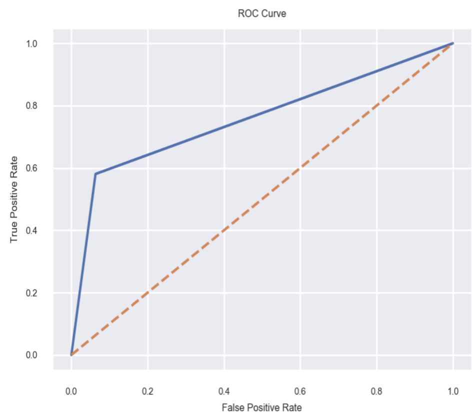
Our popularity prediction is a binary classification task, and we trained four classifiers using our created binary popularity metric: logistic regression, a decision tree, random forest, and a neural network. We split the data into a training set of 186,180 tracks and a test set of 46,545 songs (an 80/20 split).

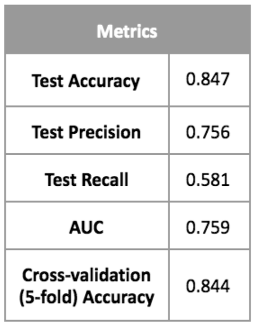
Logistic Regression Model

We first fit a logistic regression classifier to the dataset using the full feature set. We found that the test accuracy was 0.75, but in examining the ROC curve and seeing that the AUC score was 0.5, we confirm that this model does not really perform better than random guessing.

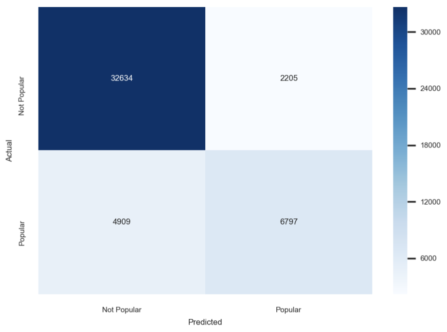
To improve this model, we performed some filtering to identify a better feature subset to train the model on. We tried various statistical metrics (i.e. correlation, mutual information) and through testing we saw that F test scores resulted in substantial model improvements.[4] Using this metric, we experimented with multiple thresholds, and found that setting the threshold to values above 7500 gave the best results. Our final model used the following feature subset: loudness, acousticness, instrumentalness, danceability, and the genres: Pop, Rock, Rap, Hip-Hop, and Dance. The test accuracy of this model increases to 0.85, as does the AUC value to about 0.76. The model also has a somewhat reasonable rate in correctly predicting popular songs (precision of 0.76), and a pretty low rate in identifying all actual popular songs (recall of about 0.58). We also performed 5-fold cross validation, and our validation accuracy is 0.844. We present the model results below:







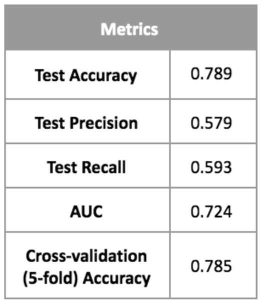
Metrics	
Test Accuracy	0.847
Test Precision	0.756
Test Recall	0.581
AUC	0.759
Cross-validation (5-fold) Accuracy	0.844



We also fit models with regularization (i.e. ridge and lasso) but did not find much improvement, so we do not include these results in the report.

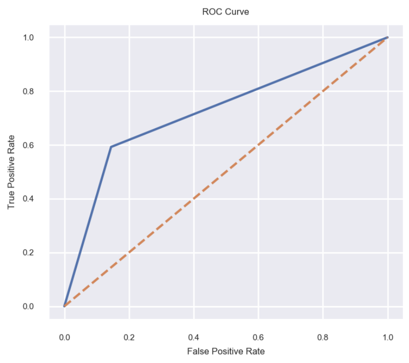
Decision Tree Model

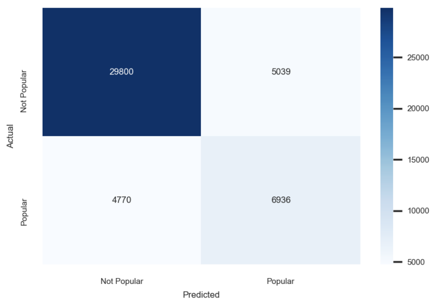
Before fitting an ensemble model, we first created a simple decision tree classifier on the full feature set. We found that a single decision tree is generally a worse classifier compared to logistic regression on all metrics: the test accuracy is 0.79, and both the recall and precision are both below 60%. However, we saw that this classifier performs better than logistic regression on the full feature set, and performs better than random (AUC value is 0.724). We will next attempt to improve our tree classifier by running a random forest. We briefly present the results below:



Metrics	
Test Accuracy	0.789
Test Precision	0.579
Test Recall	0.593
AUC	0.724
Cross-validation (5-fold) Accuracy	0.785

[4] https://towardsdatascience.com/why-how-and-when-to-apply-feature-selection-e9c69adfabf2



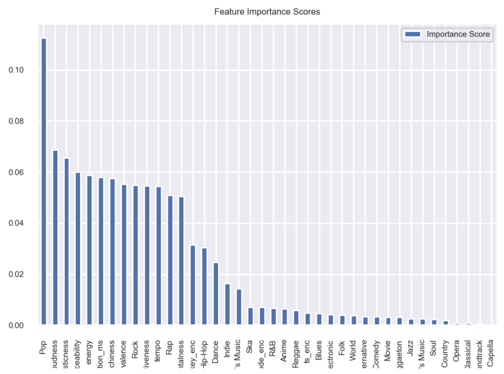


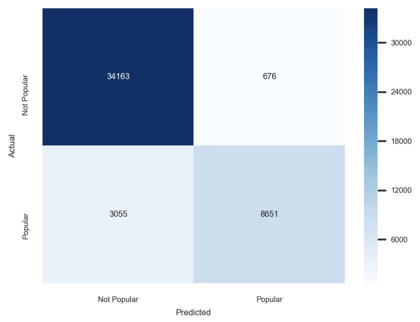
Random Forest Model

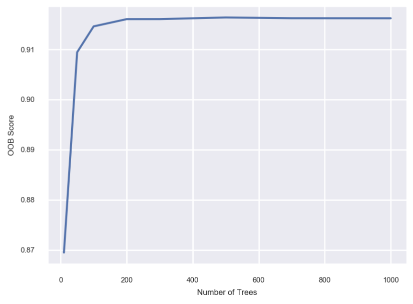
Next we fit a random forest model to predict popularity. Similar to with logistic regression, we first fit a model on the full feature set, and we see the test accuracy is about 0.828 and the AUC score is about 0.77, which is generally comparable to the results we got with our best logistic regression model.

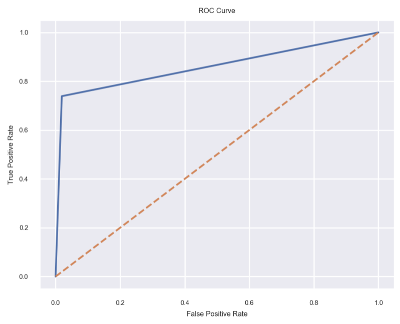
We then created a variable importance plot (using Gini impurity) to find the variables that most contribute to the popularity prediction.[5] We tested subsets of the feature set by varying the threshold, and ultimately found that using a 0.05 threshold resulted in the best results; the best model used the following features: loudness, acousticness, energy, danceability, duration, speechiness, valence and the Pop genre. Additionally, we conducted some hyperparameter tuning (using 5-fold cross validation), to determine the maximum number of estimators and maximum tree depth to use in our model. We found that the model continued to generally improve when increasing maximum tree depth and number of estimators; for our final model we did not set a depth limit.

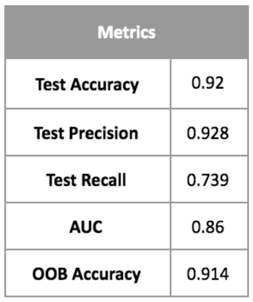
In our final model, we found that the test accuracy increased greatly compared to other classifiers, to about 0.92. We also saw improvement across all observed metrics with precision of 0.93, recall of 0.74, and AUC of 0.86. When viewing the ROC curve, we again found better results, as the curve moves closer to best performance compared to previous models. Finally when plotting the out-of-bag (OOB) accuracy against the number of estimators, we found that the scores level off (with minimal increase) after 100 estimators, thus for our final model we kept our estimators to 100 trees. We present the model results below:











Metrics	
Test Accuracy	0.92
Test Precision	0.928
Test Recall	0.739
AUC	0.86
OOB Accuracy	0.914

[5] https://www.datacamp.com/community/tutorials/random-forests-classifier-python

Neural Network Model

We finally attempted to fit a feed-forward network to predict popularity, utilizing the full feature set (we end up with an input vector of size 40, which accounts for the one-hot encoded categorical variables we created in preprocessing). We created several simple neural network models with varying architectures to experiment with various hyperparameters including: number of hidden layers, neurons per hidden layer, epochs, batch size, and learning rate. However, we did not observe significant differences across these models when looking at our various classification metrics. We also attempted to include some regularization with Dropout layers, using both 0.2 and 0.5 as the fraction, but again we did not see much of an improvement to the model. Since we had difficulty finding hyperparameters that resulted in better network performance in training, our final model architecture was pretty simple: two hidden layers (with 40 and 20 hidden units, respectively) with reLU activation, and a sigmoid activation for the binary classification (Figure 9).

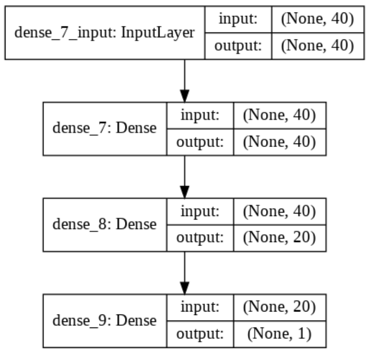
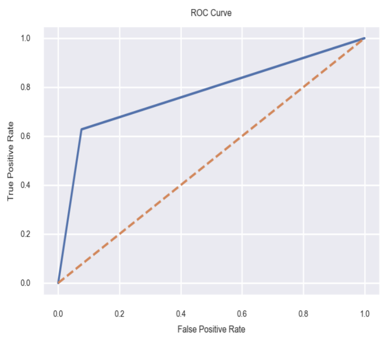
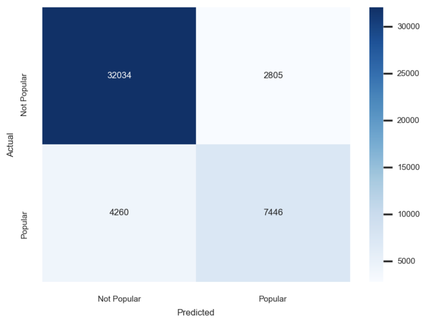
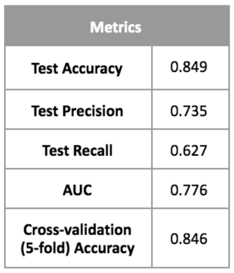


Figure 9: Song Popularity Prediction Network Architecture

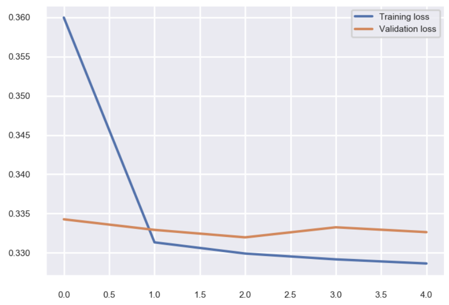
Our neural network model does not perform much better than our final logistic regression classifier, with similar test accuracy (0.85), cross-validation accuracy (0.85), and precision (0.74). The AUC score is about 0.78, which again is only marginally better than our logistic regression model. We do observe that our neural network has an improved recall compared to logistic regression, of about 0.62, so it performs slightly better in finding the actual popular songs in the data. However, our neural network results are not as good the results we found using random forest. When plotting our validation loss, we saw that the model quickly stops improving (and begins overfitting), so we limited our training to about 5 epochs. These results demonstrate that additional hyperparameter tuning (perhaps using a grid search implementation) and more data may be needed to get significant improvements when using a neural network for this task. We present the results below:







Metrics	
Test Accuracy	0.849
Test Precision	0.735
Test Recall	0.627
AUC	0.776
Cross-validation (5-fold) Accuracy	0.846



Thus we found when predicting song popularity, we get optimal results across all metrics when utilizing a random forest classifier, followed by a neural network and logistic regression with similar metrics (Figure 10).

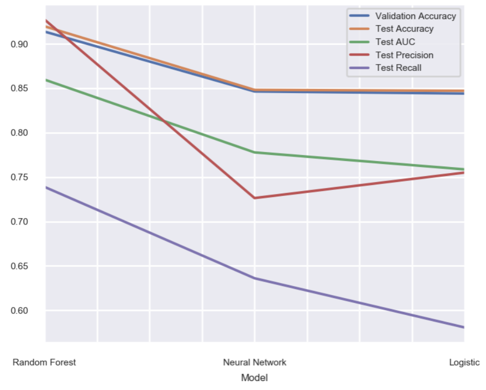


Figure 10: Song Popularity Model Metrics

<u>Mood Classification Task</u>

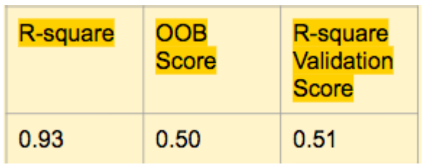
Our mood prediction is a classification task, and we trained the following classifiers using our created binary popularity metric (mood): Logistic Regression, Decision Tree, Random Forest, Naive Bayes. We also used valence, our original continuous variable, to conduct Linear regression and a Random Forest Regressor. We split the data into a training set of 186,180 tracks and a test set of 46,545 songs (an 80/20 split).

Linear Regression Model

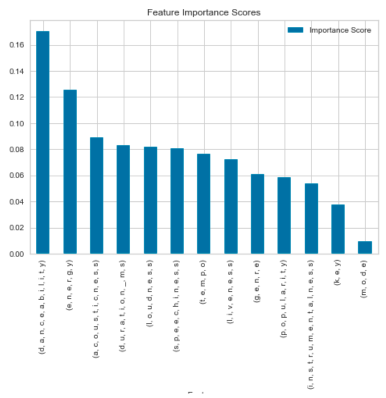
Linear regression was performed on the original dataset where the valence (mood indicator of the song) was the dependent variable and the remaining variables were taken as independent ones. Mean squared error (MSE) was calculated and we got a low value of 0.04, indicating the model was a near perfect. There is so much you can do with a continuous variable, but this measure does not help in the classification of the tracks as positive or negative music, hence we will use mood as the new target variable for our classifier models below. We decided to put in some significant parameter tuning and explore other models and different base classifiers to get a better accuracy and other metrics.

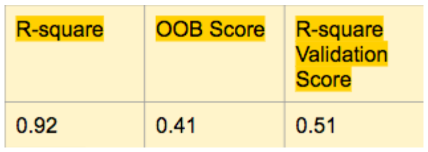
Random Forest Regressor

Before fitting a Random Forest Regressor, a feature importance graph was plotted to evaluate how the features affected the target variable; we saw that danceability holds the highest influence on valence, but there are a lot of other variables with significant and similar influence on the valence variable that must be given equal consideration. After having tested the features, the model was fit on the training set with respect to the test set and for testing set with respect to the full feature set. The R-squared value was calculated and we achieved an acceptable value of 0.93 and 0.92 respectively for both hence stating that the model does well, although having used OOB observations in the model gave us a fairly smaller value of 0.50 and 0.41 respectively indicating that the model overfits. As a result, we next moved from the continuous variable to predicting our binary mood metric with classifiers. The results of the model are as follows:



R-square	OOB Score	R-square Validation Score
0.93	0.50	0.51

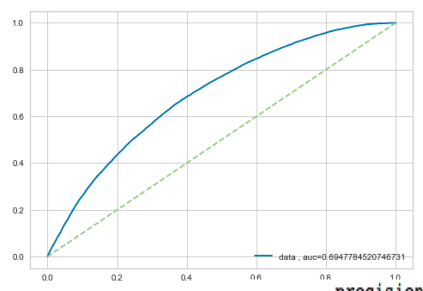


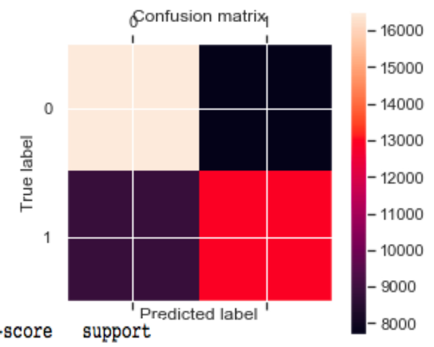


R-square	OOB Score	R-square Validation Score
0.92	0.41	0.51

Logistic Regression Model

Logistic regression was performed on the split dataset with newly created mood metric as the target variable and the remaining variables were the predictors variables; this model gave a test accuracy of 63 %. We included additional metrics to evaluate the workability of the model. The ROC-AUC curve was plotted and the value of AUC came around 69% which told us that the model was an okay fit but can do better. A confusion matrix was plotted to get the number of correctly classified songs or tracks to their moods. After getting a classification report for the logistic regression model we concluded the precision and recall values were low and that we needed to try different models to get improved results. The results of the model are displayed below:

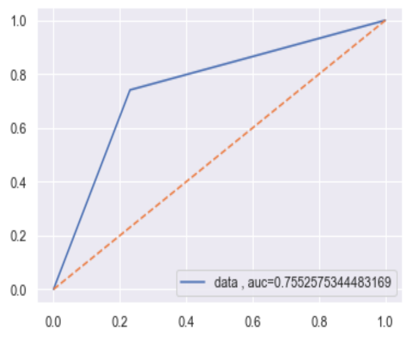


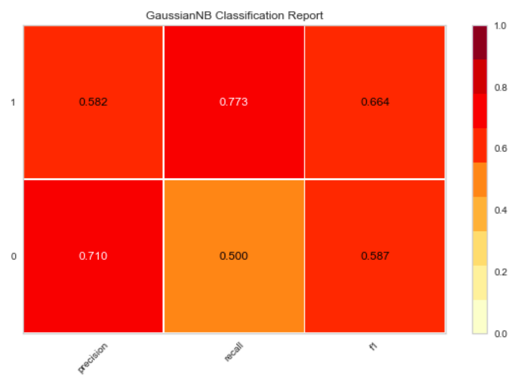


```
              precision    recall  f1-score   support

           0       0.65      0.68      0.67     24246
           1       0.63      0.59      0.61     21805

    accuracy                           0.64     46051
   macro avg       0.64      0.64      0.64     46051
weighted avg       0.64      0.64      0.64     46051
```

Naive Bayes

We tried to achieve a better accuracy by fitting a Naive Bayes (NB) on the split dataset, but it did not give us satisfactory results as the accuracy value lay in the lower 60s, similar to logistic regression. We plotted the ROC curve to evaluate the AUC value of 0.755, which indicated the model was good at correctly predicting the mood of a track; however, accuracy and precision values still needed improvement. We included some more metrics to evaluate the workability of the model. A confusion matrix was plotted to get the number of correctly classified songs or tracks to their moods. The results of the model are summarised below:

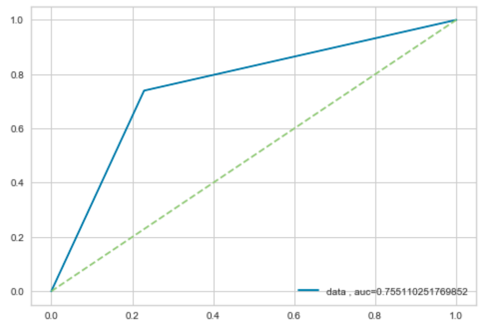


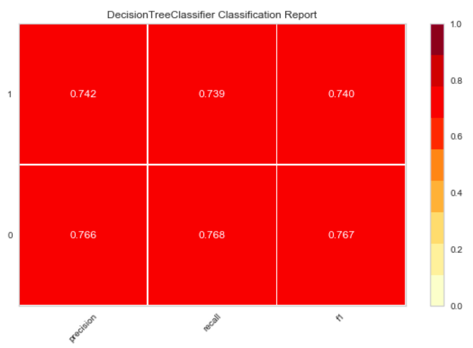


Accuracy	63%
Error	27%

Decision Tree Model

We next created a decision tree classifier on the split dataset. We found that a single decision tree was a better classifier compared to logistic regression on all metrics: the test accuracy was 75%, and both the recall and precision were 74 and 76% respectively. ROC-AUC curve was plotted and it showed better results as compared to the Logistic Regression, with a value as high as 76%.The results of the model are summarised below:



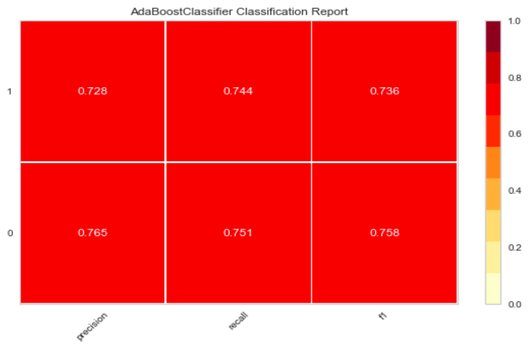


AdaBoost Algorithm Estimator: Trees)

(Base Decision

We built on our decision tree model by fitting a model using AdaBoost, using decision trees as our base estimators. We varied the number of estimators between 50-100 but not much difference was perceived in the accuracy value. Classification reporting using the *yellowbrick* package in Python was utilized to get values for precision and recall, and the results of the model are summarised below[6]:

```
Accuracy for n = 50 is 0.7559662113743458
Error for n = 50 is 0.24403378862565417
Precision for n = 50 is 0.7428860387073047
Recall for n = 50 is 0.7411144232974088
Accuracy for n = 100 is 0.7562702221450132
Error for n = 100 is 0.24372977785498684
Precision for n = 100 is 0.7431854746035395
Recall for n = 100 is 0.7414813116257739
Accuracy for n = 200 is 0.7560096414844412
Error for n = 200 is 0.24399035851555884
Precision for n = 200 is 0.7429766885833832
Recall for n = 200 is 0.7410685622563632
```

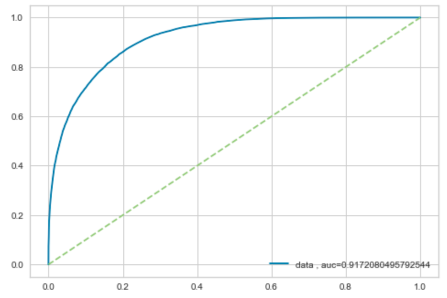


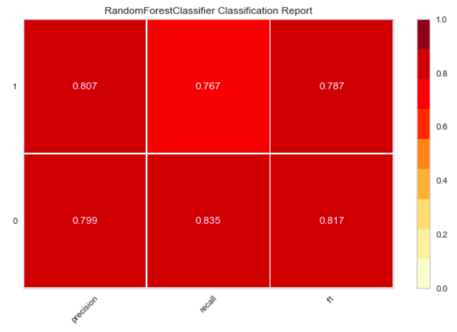
Random Forest

We next created a random forest model to classify mood. We made a variable importance plot (using Gini impurity) to find the variables with the strongest influence on mood classification. We tested subsets of the feature set by varying the threshold, and ultimately we kept most of the variables, as they were equally contributing to the mood variable. Additionally, we varied the maximum number of estimators from 50 to 150 and used entropy as our criterion in our model. We found that the model improved when increasing maximum tree depth, and for our final model there was no depth limit.

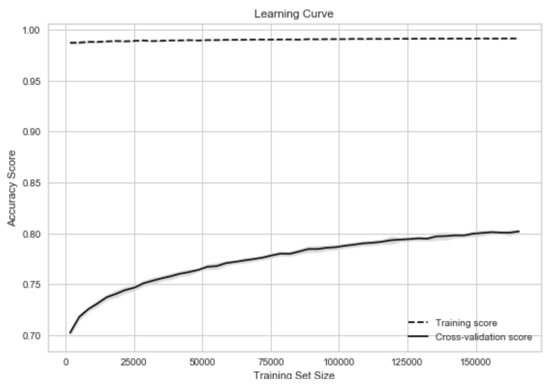
In our final model, we found that the test accuracy increased greatly compared to other classifiers, to about 0.82. We also saw improvement across all observed metrics with a precision of 0.80 and recall at 0.84. Even our ROC-AUC curve showed a value as high as 91% which was the best value by far for AUC. So the random forest greatly improved our prediction results and was decided upon as the best model for mood classification. We present the ensemble results below:

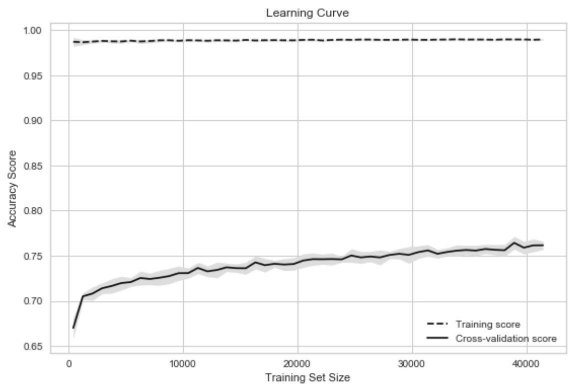
```
Accuracy for n = 50 is 0.8259104036828734
Error for n = 50 is 0.17408959631712662
Precision for n = 50 is 0.8050982474774296
Recall for n = 50 is 0.8343040587021325
Accuracy for n = 100 is 0.8283859199583071
Error for n = 100 is 0.17161408004169287
Precision for n = 100 is 0.8045211600806098
Recall for n = 100 is 0.8421921577619812
Accuracy for n = 150 is 0.828711645784022
Error for n = 150 is 0.171288354215978
Precision for n = 150 is 0.8037982973149967
Recall for n = 150 is 0.8443476266911258
```





[6] https://www.scikit-yb.org/en/latest/



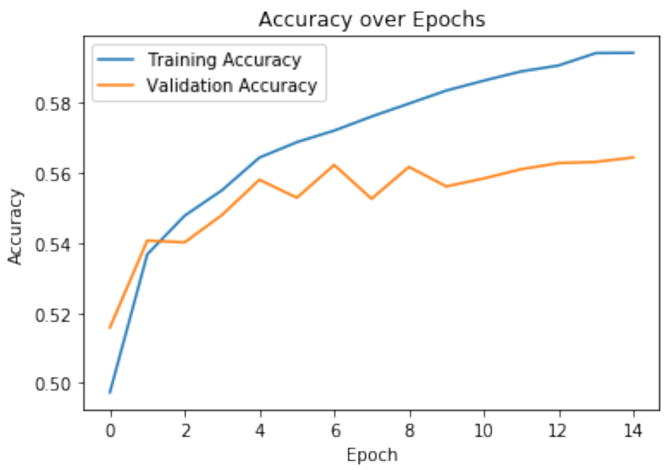


Genre Classification Task

Our final task was to build a multi-class classification model that can automatically tag and predict a song's genre. As discussed in the *Preprocessing and Feature Engineering* section, we reduced our space of 26 genres down to 10. The multi-class approach added a different dynamic and we ultimately found 3 models that performed the best for this task: a softmax neural network, a Gaussian kernel support vector machine (SVM), and random forest. For this task we split the data into 68,950 songs for training and 22,984 songs for testing (75/25 split with a filtered down dataset as a result of reducing genres).

Neural Network

The first model we fit was a neural network. Given our multi-class approach we utilized softmax activation in the output layer as a way to handle our multiple classes. To tune the network and find the right architecture we first started with a very basic model and decided to work our way up, by adding layers and/or neurons, and testing performance as we went. We started with 1 hidden layer of 500 neurons, and eventually we reached a threshold architecture that performed the best but added more complexity and actually reduced the accuracy. The final architecture had 30 neurons for our input layer (representing our 30 features), 3 hidden layers with 1000, 700 and 250 neurons respectively and 10 neurons for our output layer (representing our 10 genres). The 2nd and 3rd hidden layers had dropout rates of 35% and 15% respectively. We found that adding this regularization to our complex network greatly improved the validation and test accuracy. Our hidden layers used reLU activation and the output layer used softmax activation. The model training and validations results can be viewed in the plot below:



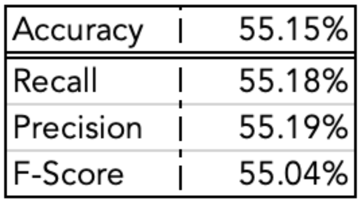
Viewing the training over 15 epochs, the training accuracy, to no surprise, continues to improve while the validation accuracy was rocky in training for the middle epochs, but then smoothed out and converged to 56% accuracy after 9 epochs. We initially thought this accuracy was pretty low but we considered a random model would have 10% (1/10) accuracy and this range we achieved was in line with other attempts at this problem.[7]

Gaussian Kernel SVM

[7] http://article.nadiapub.com/IJDTA/vol9_no5/13.pdf

Next we trained support vector machines. We first tried a linear SVM and compared these results with different kernel SVMs. The best performing model was the Gaussian (or RBF) kernel. This gave us insight into the non-linearity of our data. We then wanted to tune the hyperparameters of the Gaussian kernel SVM, specifically our C and gamma values, through cross-validation and grid search. Essentially creating a possible space of C and gamma values, cross-validation can evaluate a SVM model at each fold with each parameter value; it can then search through the entire grid space and return the best performing model.

We then wanted to plot the results of this search in a heatmap to gain insight into how our model performed in the grid space. Unfortunately, this grid search process proved too expensive for our machines. We tried running the lowest possible suitable grid search to actually gain an effective model, but our code would run for many hours with no result. We did however, overcome this challenge through some manual tuning to still find a near optimal C values of 3; this value gave us good results, but was not high enough to introduce too much bias in our model, or too low to create an overfitting model. We also used a gamma of 1 / (# of features * variance of training data). Our results can be summarized in the table below:

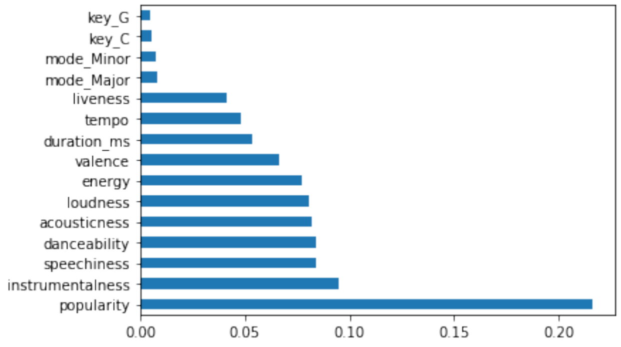


Accuracy	55.15%
Recall	55.18%
Precision	55.19%
F-Score	55.04%

Remarkably our accuracy, recall, precision and F-Score metrics were all right around 55%. In comparison to the neural network, this model's performance was only slightly below by 1% of accuracy. The precision and recall of our SVM also lets us know that the model performs the same in correctly predicting the right genre out of possible number of genres as well as when it predicts a genre, the genre actually being that genre the model predicted. Due to the nature of our multi-class approach, retrieving ROC curves and AUC values proved to be a difficult task.

Random Forest

Finally, we trained a random forest to compare to the SVM and neural network. We used trial and error to tune our hyperparameters, and found the optimal number of trees in the ensemble was 100 and entropy as the impurity measure. The random forest performed worse than both SVM and our neural network, but we were able to extract the feature importance of each feature to genre classification in the forest. While not being the most accurate model we gained insight into which features were contributing the most and more predictive for determining a song's genre.



Popularity is far and away the most important feature. This was at first a surprise as our intuition figured that the precise song measurements like the key, mode, loudness, etc. would be more impactful for determining how a song is classified differently from another song. This did not turn out to be the case as it was actually how popular a song is, given Pop and Rap are most popular, while a genre like Classical music is on the opposite end of the popularity spectrum. Given our low accuracy and intuitions proved wrong, determining a song's genre is a hard task to do even with lots of training data and precise measurements to go off of. We discuss future areas to explore on genre tagging in the *Future Studies* section.

Discussion

Our project was focused on better understanding what factors play into a person's experience in selecting and listening to music, and seeing if we could design classifiers to predict these aspects, specifically: song popularity, mood, and genre. We found that we could predict song popularity (i.e. whether a song is popular or not), with about 92% accuracy when using a random forest ensemble. For mood (our binary valence metric), we again concluded that random forest was the optimal model, with test accuracy of 83%. And when classifying song genre, we found that the feed-forward neural network produced the best results, with a test accuracy of 56%.

Challenges

One of the most challenging aspects of this project was acquiring an appropriate dataset. We originally had a dataset with one million songs, but it ended up having a lot missing data (especially for variables related to our prediction tasks), which resulted in reducing the amount of data we had to less than half of the original size. We were able to find a better dataset with appropriate data for all three of our classification tasks, but some of the drawbacks of this dataset were that there were fewer records and features with which to work. Another resulting challenge of switching datasets was that we had less time to finish our modeling tasks and go more in depth into hyperparameter tuning.

Another challenge was in creating and training effective architectures for the neural network models. There are many hyperparameters to tune, and it was difficult to determine what the best combination of parameter values were, and which ones were creating substantial performance improvements. As mentioned earlier with neural network for song popularity and SVM for the genre task, utilizing a search function such as grid search for tuning would result in better results. Unfortunately this was expensive to implement to account for all hyperparameters of interest.

Futures Studies

Some avenues of future research could include exploring different thresholds when creating binary measures for popularity and valence. Additionally, perhaps being able to incorporate lyric data (if it is available to scrape from the Spotify API) would help in developing a more robust multi-class mood metric; utilization of sentiment analysis could help in developing nuanced mood categories. Furthermore, if we are able to find a dataset with fewer genres (without having to filter out lesser known genres and losing data), perhaps that would help in improving genre prediction. To improve genre prediction we could look at gathering images of a song's wavelength. This could then turn into an image classification problem using Convolutional Neural Networks, which could perhaps increase accuracy. Also, this project could be extended into developing a song recommendation system, which is the current focus of many streaming services.[8]

Conclusion

This project sought to use machine learning to create effective classifiers in predicting important factors considered when analyzing music listening habits. Overall we were able to gain a better insight about song metadata and audio features, and we were able to produce models with relatively good performance in predicting popularity, general mood, and genre of a song.

References

Datasets

1. *Million Song Dataset*: http://millionsongdataset.com/

   This was the original dataset we chose to work on. Although there is a lot of missing data, this website provides information about tasks that can be solved using song datasets.

2. *Spotify Tracks DB:* https://www.kaggle.com/zaheenhamidani/ultimate-spotify-tracks-db

   This is the new dataset we decided to work with. The author of the dataset provided information about scraping the data from the Spotify API and the types of measures obtained.

Song Popularity

1. *Song Popularity Predictor:* https://towardsdatascience.com/song-popularity-predictor-1ef69735e380
   This tutorial goes over a procedure used to create a popularity predictor using the Million Song Dataset (MSD) and BillBoard Top 100 information to replace the missing popularity data.
2. *Predicting Song Popularity*: http://cs229.stanford.edu/proj2015/140_report.pdf

[8] https://towardsdatascience.com/how-to-build-a-simple-song-recommender-296fcbc8c85

An academic paper from Stanford discussing song popularity prediction models using the MSD. Provides information on both classification and regression models implemented.

3. *Predicting Song Popularity*: https://github.com/amninder-dhillon/PredictingSongPopularity/tree/master/src
   Another project that looks at song popularity using the full MSD dataset. Gave information about potential filtering methods to employ.
4. *Spotify: Analyzing and Predicting Songs*: https://medium.com/mlreview/spotify-analyzing-and-predicting-songs-58827a0fa42b
   A post about Spotify API data and potential tuning procedures and metrics to look at.
5. *Song Popularity Predictor*: http://www.maikaisogawa.com/portfolio/song-popularity-predictor/
   Tutorial reviewing song popularity combining Billboard, LyricsFreak, and Spotify APIs.

Song Genre

1. *Automatic Tagging of Songs:*
   http://article.nadiapub.com/IJDTA/vol9_no5/13.pdf
   This is a paper that used various machine learning algorithms to classify the genre of a song. The authors used the million song dataset.

Song Mood

1. *Music Mood Classifier:*
   https://github.com/rasbt/musicmood/tree/c942e2592ccc1b092d950a5cd3c9050419728a6a
   This project is about building a music recommendation system for users who want to listen to *happy* songs. Such a system can not only be used to brighten up one's mood on a rainy weekend; especially in hospitals, other medical clinics, or public locations such as restaurants, the MusicMood classifier could be used to spread positive mood among people.
2. *Music Mood: Predicting the mood of music from song lyrics using Machine Learning:*
   https://arxiv.org/pdf/1611.00138.pdf
   This is a full fledged paper written by Sebastian Raschka that is being used a guide book and we are basing our work on this as it has a detailed description on how music is corresponding to a happy mood and it can be detected with high precision based on text features obtained from song lyrics.