Hit Song Classification of Billboard 100 Songs

Josh Kowal
Kunal Rai
jak13@rice.edu
ksr3@rice.edu
Rice University, DSCI 303

ABSTRACT

This project is focused on the Hit Song Science problem with the goal of predicting which songs will become Billboard Top 100 hits based on their audio features, lyrical features, and artist information. For the purposes of this project we curated a data set of approximately 3,600 non-hit songs and 3,600 hit songs using the Spotify 160k+ Kaggle Dataset, the Spotify Web API, and the historical Billboard Top 100 Charts. From the results of this project, we were able to predict if a song is a Billboard Hit with an AUC of 0.74 at best on our test sets, with the best performing models being decision tree based models such as Random Forest, XGBoost, and Gradient Boosting Classifiers.

1 INTRODUCTION

Hit songs are important for artists as they generate more revenue and allow the artist to share more of their music with more people. Releasing a hit song makes an artists' name well known, which increases the number of searches for the artists' name and the frequency with which someone listens to their songs. As most song platforms, including Spotify and Apple Music, pay song artists every time someone streams their songs, releasing a hit can generate additional revenue for the artist.

In regards to song production, hit song prediction is important as artists could use information about hit song qualities and common features when creating their next tracks or albums. Labels could choose to invest in artists whose music tends to contain more specific audio or lyrical features that make the song more likely to become a hit [1]. Certain genres of music go in and out of fashion, so an artist may choose to deviate from their typical genre in order to release a song under a genre that is currently more popular. Alternatively, if an artist wants to share a song with a smaller audience, they may choose to release a song under a less popular genre with certain lyrical and audio features that make it less likely to become a hit. Further within the realm of machine learning, hit song prediction could be used to guide computer production of music.

2 PRIOR WORK

Previous works in hit song science have taken different approaches to ours when it comes to feature engineering and the data utilized. One approach concluded the feasibility of predicting hit songs from specific features better than arbitrary guessing and that a song's tempo largely indicates its popularity in a given time period [2]. As our approach focused on hit classification across multiple decades, we did not try to predict popularity of songs within specific time periods, so we focused on the first result for our project.

Another approach sought to classify songs into genres based on their audio features and lyrical sentiments [3]. For instance, the predominant audio features and lyrical sentiments for rap songs may be different than the predominant audio features and lyrical sentiments for pop songs. This could be useful for predicting hit songs since a significant portion of the hit songs may fall into one or two genres; thus, assigning a genre to a song would give our models an additional feature that could better classify whether a song is a hit or a non-hit.

The approach we sought to emulate the most worked on our specific problem with a much larger data set than the one we used in our work [4]. In this approach, Middlebrook 2019 used the Spotify API and the Billboard API to create a data set of roughly 24,000 songs with 12,000 hit songs and 12,000 non-hit songs. They also engineered features such as artist-past-performance and time spent on Billboard Top 100 chart. Their models achieved roughly 80-90 percent accuracy when predicting whether a song was a hit or not [4]. Because of their great results, we decided to engineer a few similar features in addition to our own. We also utilized these results to motivate creation of similar models during our modeling stage.

3 METHODOLOGY

In this section, we discuss how we acquired our data and performed feature engineering. Then, we discuss data pre-processing and model selection.

3.1 Data Acquisition

In order to curate our data set, we needed to acquire songs from the past that had audio features available on Spotify as well as hit songs that made it to the Billboard Top 100. We used the publicly available Spotify 160k+ data set as our starting point for generic non-hit songs [5]. To get hit songs we web-scraped all the Year-End Billboard Top 100 Hit songs from 1970-2017 and retrieved the Spotify audio features for these tracks. These data sources were merged to form the initial data set of hit and non-hit songs.

3.2 Data Exploration

The Spotify 160k+ Track data set features over 160,000 tracks across multiple languages, genres, and decades. In Figure 1, the Spotify audio features for the initial data set are shown over time. Over the decades, acousticness decreased significantly while energy has increased, and the other core audio features have varied slightly. We can attribute these changes over time to how some genres have risen to popularity while others have become less mainstream.

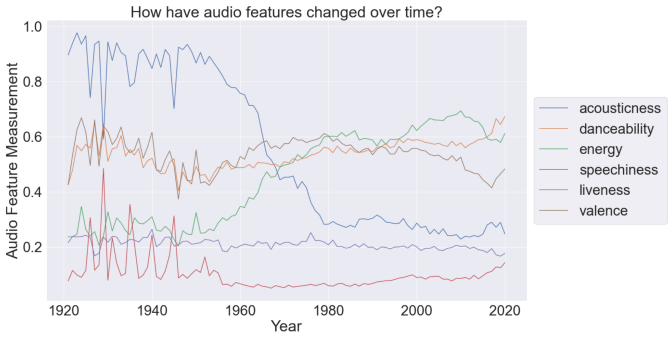

Figure 1: How Audio Features Changed By Decade

3.3 Feature Engineering

3.3.1 Spotify Audio Features. Using the Spotify Web API, audio features for a given track can be retrieved. Examples of these audio features can be seen in Table 1.

Table 1: Spotify Audio Features

Feature	Details
Danceability	how suitable a track is for dancing based on musical elements
Liveness	presence of a live audience in a recording
Speechiness	presence of spoken words in a track
Valence	measure describing musical positiveness conveyed
Tempo	estimated tempo in beats per minute

These are only a subset of all of the Spotify audio features; the list of all features we retrieved via the API and their details can be seen in the Spotify Web API documentation. The audio features available from the Spotify API are of different data types and have different scales. Some are binary, some are integer valued, and others are continuous. We dealt with these data types in the pre-processing stage.

3.3.2 Lyric Based Features. From prior work, it has been shown that bag of words features, n-grams, and rhymes can be used in varying effectiveness for genre classification problems. However, lyrical sentiment has been used to distinguish genres effectively [3]. We extended the use of lyrical features to hit song prediction through feature engineering. From the combined data set of Spotify songs and Billboard Hits, we reduced the data to include only English songs and retrieved the lyrics from Genius Lyrics for these songs. Using the lyrics, we extracted all tri-grams (three word phrases or strings) to create a feature of repeated phrases. We considered a phrase to be a repeated trigram using a threshold of 10 to determine how many times a trigram must appear in order for it to be a repeated phrase. This threshold was determined by manual testing of different values. With the lyrics, we also wanted to extract sentiment features. Using the TextBlob package, we were able to create lyrical subjectivity (how subjective lyrics are), and lyrical polarity (whether songs are mostly positive or negative) features to capture the sentiment of a song.

3.3.3 Additional Features. Further, we added word count (number of words in the song lyrics), words per minute (average number of words present in each minute of the song), and playlist genre and sub-genre (ex. the genre/sub-genre of playlists that the song appears in). Rather than general genres, we hypothesized that songs placed in curated playlists had higher likelihood of correct genre classification. Finally, within the data set we created an artist past performance feature which was a count of how many Billboard Hits an artist had created before the given track, only for songs contained in this data set. To create our target variable, we used the list of Billboard songs from 1970-2019 to provide a binary value for songs that either were a hit or were not.

3.3.4 Final Data Set Information. Due to the analysis of English only lyrics and lyric availability, our data set was reduced to 3,600 hit songs and 3,600 non-hit songs with Spotify audio features and our added features. Within this data set we had 18 features, 9 from the Spotify API and 9 produced through feature engineering and augmentation.

3.4 Pre-Processing

Due to the original data being sourced from the Kaggle data set and the reduction of our data set size with lyrical availability, we did not have to deal with missing values. From the feature engineering stage, we did create both categorical and numerical features. In order to standardize the data for our training matrix, we used label encoding for our categorical features in order to allow them to be interpreted by the machine learning models. We did not conduct feature selection or scaling at a general level before passing it to models as different models would use different feature selection methods (wrapper and embedded).

3.5 Models

Since our problem is a binary classification problem (hit or not), we looked at prior work as well as models that we thought would be conducive to this data set to determine which models to use. We chose some linear and non-linear models to see which type of model would have better performance and then compared models based on their confusion matrices and receiver operating characteristic (ROC) curves. ROC curves show a model's true positive rate on the Y-axis and false positive rate on the X-axis, and the closer to the top left of plot the curve is and the larger the area under the curve, the better the classification results. This is because the model maximizes true positives while minimizing false positives. Comparison of our models were conducted with the metric of the area under the curve (AUC) of these ROC curves.

3.5.1 Logistic Regression. Since the hit or not target variable is binary, logistic regression was the initial method chosen to see if there was linear separation between the two classes. Binary logistic regression combines input values from the training data linearly with coefficients to produce an output prediction of a class. Logistic regression uses maximum-likelihood estimation to calculate the coefficient values from the training data.

3.5.2 Support Vector Machine. A support vector machine is a binary classification model that fits two vectors, called support vectors, to the data in order to form a decision boundary. The data

point is classified based on which side of the decision boundary the point falls. Our idea for creating a support vector machine comes from previous work [4]. We also figured we should try support vector machines since our problem is a binary classification problem. In case our data wasn't linearly separable, we could use nonlinear kernels when creating our support vectors.

3.5.3 Neural Network. The neural network was motivated by prior work, which used a fully connected, single hidden layer model for classification [4]. The structure of our model was to go from 18 initial features to 12 features with a dense layer with Relu, then to condense down to 1 using another dense layer with the sigmoid activation function. With the given data, dense layers made the most sense for model performance.

3.5.4 Gradient Boosting Classifier. Gradient Boosting classifier is an additive classification model for which each new tree is fit on modifications to the original data. Gradient Boosting identifies the weak decision trees by using gradients in its loss function, which measures the model coefficients efficacy at fitting the data. We hypothesized that our data would not be linearly separable, and thus models that can find decision paths for hit or not songs based on the feature set we created would have a better chance of classifying songs correctly.

3.5.5 Random Forest Classifier. Random forests mitigate the over-fitting problem from decision trees by averaging the performance of multiple trees on a smaller number of features. We drew inspiration from previous work when choosing to create a random forest model [4]. Also, since we thought our data wasn't linearly separable, we believed using an ensemble method was good for performing classification. It also allowed us to see which sets of features had more importance when the model had to classify a song as a hit or not.

3.5.6 XGBoost Classifier. XGBoost is a decision tree based model that works by the ensemble method by boosting trees. The premise of XGboost is that it uses gradient descent to continually correct previous mistakes made by the model to improve performance until the performance or parameter limits are reached. This model was chosen after seeing the performance of the Gradient Boosting and Random Forest models as it is a higher performance implementation of decision tree models.

4 EVALUATION

In this section, we discuss the results of each model and compare our models with previous work. We then discuss why some models may have performed better than others.

4.1 Results

4.1.1 SMOTE. Synthetic Minority Oversampling Technique is a data augmentation method to address imbalanced data sets. SMOTE was tested with our data set before we balanced the non-hits to hits in order to see if we could achieve a larger data set size for training a few of our models.

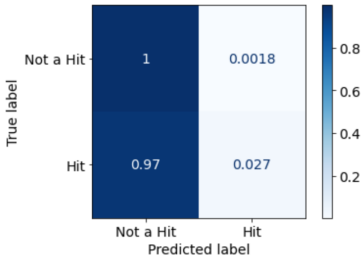

Figure 2: Confusion Matrix for Gradient Boosting SMOTE

After up-sampling our training data to roughly 16,000 hit songs and 16,000 non-hit songs using SMOTE, we ran some of our models (Logistic Regression, Gradient Boosting Classifier, and Neural Network) with this training data. However, our results from using SMOTE indicated over fitting to the majority class even after equalizing the data. In Figure 2, we can see an example of the Gradient Boosting model with SMOTE training data. This is an extreme example of over fitting that we saw when using SMOTE upsampling. With other models run on SMOTE training data, we found that the up-sampled data set either did not significantly improve performance or also caused some kind of over-fit. Due to these results from SMOTE, we decided to balance our data by reducing the number of non-hit songs to equal hit songs and not utilize SMOTE. All results presented in further sections are without SMOTE up-sampling.

4.1.2 Logistic Regression. Logistic regression was used as a baseline method for linear separation of the data. Logistic regression requires normalization of data, so we used Z-score normalization. Recursive feature elimination (RFE) was used in order to choose optimal features. RFE selected 6 audio features and 4 added features (Lyrical Subjectivity, Polarity, Minutes, and Artist Past Performance). Further, we ran Grid Search on the C parameter (inverse of regularization strength). With these features and tuned parameters, logistic regression had a AUC of 0.64.

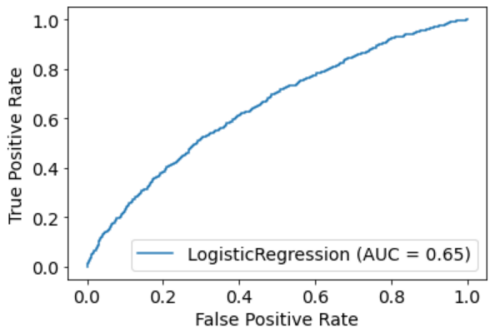

Figure 3: ROC Curve for Logistic Regression

4.1.3 Support Vector Machine. We used Grid Search cross validation to optimize parameters of our SVM. We searched the learning rate (C), the degree for a polynomial kernel (degree), the kernel coefficient for nonlinear kernels (gamma), and the type of kernel (kernel). Each model we tried received all of our training features as input. Our best model used a radial basis function with a learning

rate of 1 and a scaled value for gamma (γ). With these parameters, our best SVM had an AUC of 0.67.

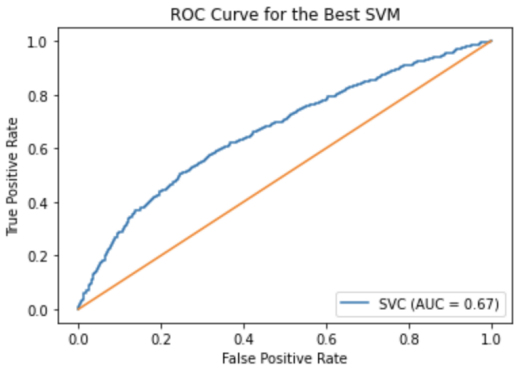

Figure 4: ROC Curve for Support Vector Machine

4.1.4 Fully Connected Neural Network. For neural networks we also used Z-score normalization as it helps with network convergence and performance. The model was compiled with binary cross entropy for this problem and trained on 100 epochs with a batch size of 70. The optimizer chosen was Adam, and learning rate was set to 0.0001. These values were chosen through manual tuning of the model evaluated against the convergence of training and validation accuracy. In Figure 5 we can see the train and validation accuracy increase and taper as it approaches 200 epochs. The AUC of this neural network was 0.64.

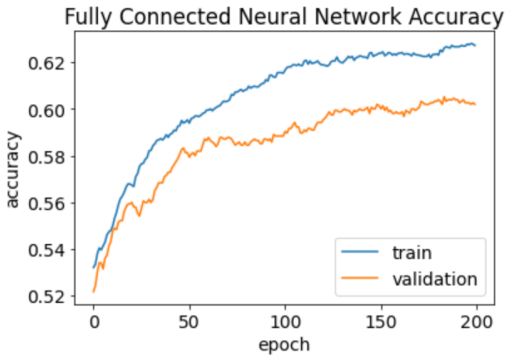

Figure 5: Neural Net Training Accuracy

4.1.5 Gradient Boosting Classifier. In order to optimize performance for the Gradient Boosting Classifier, we used Grid Search three-fold cross validation. The parameters we chose to search were the min samples split (minimum number of samples required to split an internal node), min samples leaf (minimum number of samples to be at a leaf node), max depth (the limit of the number of nodes in the tree), and number of estimators (the number of boosting stages to perform). The gradient boosting classifier with the best parameters had an AUC of 0.73 as shown in Figure 6.

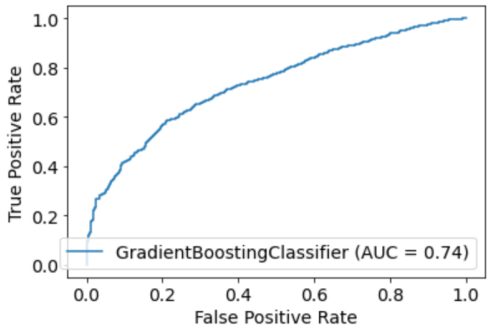

Figure 6: ROC Curve for Gradient Boosting Classifier

The gradient boosting classifier with the best performance also indicated to us what the most important features were: Speechiness, Minutes, Lyrical Polarity, Valence, and Tempo.

4.1.6 Random Forest Classifier. To optimize performance of Random Forest, we used Grid Search cross validation. We searched over whether or not to bootstrap (bootstrap), the maximum depth of a tree (max_depth), the number of features to consider when looking for the best split (max_features), the minimum number of samples required to be at a leaf node (min_samples_leaf), the minimum number of samples required to split an internal node (min_samples_split), and the number of trees to include in the forest (n_estimators). After using this technique to fine-tune the parameters of Random Forest, we found our best forest had 110 trees with maximum $\sqrt{n}$ features per split in a tree, where n is the number of features. The forest also used bootstrapping and split based on 2 samples, and each tree had at least one sample per leaf and no specified maximum depth. Based on Figure 10, this forest had an AUC of 0.72.

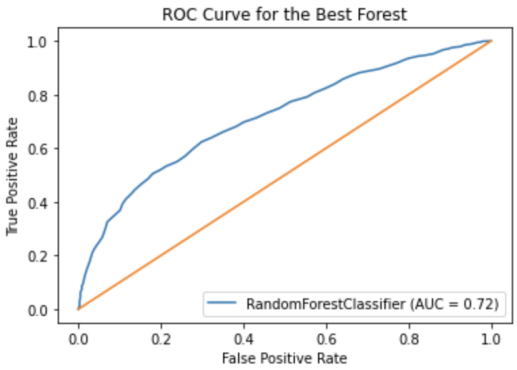

Figure 7: ROC Curve for Random Forest Classifier

4.1.7 XGBoost. After tuning our models and seeing the success of the decision tree based models, we tried an XGBoost Random Forest model. We hyper parameter tuned the max tree depth and child weights of tree partitions to produce the best-performing model. XGBoost Random Forest yielded an AUC of 0.72, which was not significantly better than our gradient boosting model.

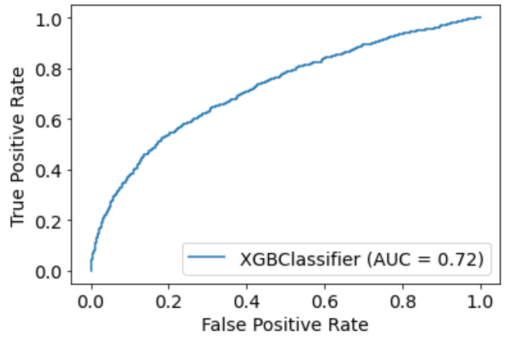

Figure 8: ROC Curve for XGBoost

4.2 Model Comparison

Overall we saw that non-linear algorithms performed much better than linear algorithms on hit song prediction. This is because our data turned out to not be linearly separable. Among the non-linear methods we tried, decision tree based models performed the best, with Random Forest Classifier achieving a top AUC of 72% and Gradient Boosting Classifier achieving a top AUC of 74%.

Refer to the table below for each model's true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and false negative rate (FNR).

Table 2: Confusion Matrix Results

Model	TPR	TNR	FPR	FNR	AUC
Logistic Regression	0.6	0.61	0.39	0.4	0.65
Support Vector Machine	0.57	0.68	0.32	0.43	0.67
Neural Network	0.59	0.61	0.39	0.41	0.64
Gradient Boosting	**0.65**	**0.7**	**0.3**	**0.35**	**0.74**
Random Forest	0.62	0.7	0.3	0.38	0.72
XG Boost	0.65	0.68	0.32	0.35	0.72

From the table, all our models had a higher true negative rate than a true positive rate. Since there were more non-hits than hits and we randomly sampled from non-hits, this could mean that our sampled non-hits were more separable from our dataset than our hits. This would make it more likely to correctly classify a song as a non-hit than a hit, hence the higher TNR.

4.3 Comparison with Prior Work

Throughout our project we have used prior work to motivate modeling decisions and feature engineering. These prior works exist in the same domain of hit song science but have used different data sets as well as have different target variables or desired outcomes. Bal 2018 focused on genre classification and used lyrical sentiment features with audio features. The multinomial logistic regression model had a confusion matrix classification performance of 0.687 with an overall accuracy of .415 [3]. Further, Bal 2018 also implemented undersampling, oversampling, and SMOTE to augment the data used with overall accuracy of 0.19, 0.193, and 0.296 respectively. Although the underlying problem is different, this result aligns with our results from sampling correction using SMOTE. Conclusions from Bal 2018 indicate that lyrical sentiment and audio features together can contribute to higher genre classification scores [3]. We extended this hypothesis to hit song prediction and also found that lyrical sentiment features aided in classification of hit songs with our best performing models using lyrical polarity as a highly weighted feature.

Middlebrook 2019 worked on the same problem we addressed, Billboard hit prediction using Spotify data [4]. This paper used 12,000 hits and 12,000 non-hits, addressing our issue of limited data to create more robust models. Further, the features used by Middlebrook 2019 consisted of all Spotify audio features, and the target variable was also whether a song appeared in Billboard Top 100 charts or not. Conclusions by Middlebrook 2019 indicate that SVM and Random Forest Models performed better than Logistic Regression and Neural Net in classification. This result aligns with the results of our project. However, Middlebrook 2019 uses classification accuracy as their target metric while we used AUC. Middlebrook 2019 concluded with an average classification accuracy of 0.85. Since AUC and classification accuracy cannot be compared one to one, if we calculate our classification accuracy from the confusion matrix results, our classification accuracy of our best models is roughly 0.67, which is lower than Middlebrook 2019 [4].

5 CONCLUSION

Whether a song will or will not make the Billboard Top 100 is not easily predictable. From the results of our models, we could predict if a song would actually be a Billboard Hit only 65% of the time. This model points to speechiness, duration in minutes, lyrical polarity, valence, and tempo being highly influential towards a song's popularity, but these features will change moving into the future as music and music preferences change. This study was limited by the data we were able to collate, and potentially with more data and more top hit charts, model performances and feature importances could increase and change respectively.

Given more time, we would recreate the artist past performance feature and look more closely into how previous work calculated this feature [4]. We would also try to extract the lyrics for more songs and refine how we determine whether a song was a hit or not. This could be done by starting with a larger data set and converting each song name and artist name into their own unique format, which would make it easier to determine song and artist equality across tables. Another approach we could take is to use SMOTE on our data set and refine it so it yields higher AUC for our models. This would give us more data on which to train, which would allow our models to better generalize.

We faced several limitations in our project. One of which was the presence of repeated songs that were re-released in future years. This made it difficult to determine hit songs because if that song was a hit when it first released, it may not have been a hit in the re-released year. Also, the audio features were slightly different, which means the audio features might be based somewhat on the time period. Another limitation was the presence of remixes in our data, especially if the original version of the song was a hit. We assumed that the remixes were hits as well, which gave a higher value to that song artist's past performance feature. Finally, we had difficulty obtaining data sets with different features that matched on songs. For instance, a lot of songs in one data set that had song lyrics did not appear in our Spotify 160k+ data set. This made it

difficult to join features in both data sets, and we did not have access to the API that could generate all the lyrics for us. Consequently, we worked with a smaller data set, which impacted our results.

REFERENCES

[1] Vincent Cheung. *The Science of a Billboard Hit Song https://www.mpg.de/14119325/1108-nepf-132884-the-science-of-a-billboard-hit-song* Max Planck Institute for Human Cognitive and Brain Sciences, Leipzig

[2] Ni Yizhao, Raul Santos-Rodriguez, Matt Mcviar, Tijl De Bie. *Hit song science once again a science. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.412.9732rep=rep1type=pdf* 4th International Workshop on Machine Learning and Music, Spain, 2011.

[3] Anneloes Bal. *Music genre classification using audio features and lyrics sentiment features with multinomial logistic regression* `http://arno.uvt.nl/show.cgi?fid=147004` Masters Thesis Tilburg University, 2018

[4] Kai Middlebrook, Kian Sheik. *Song Hit Prediction: Predicting Billboard Hits Using Spotify Data. https://arxiv.org/pdf/1908.08609.pdf* Preprint, University of San Francisco, 2019

[5] Yamac Eren Ay. *Spotify Dataset 1921-2020, 160k+ tracks https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks*

A APPENDIX

The repository for the work discussed in this paper is accessible here.