Predicting the Popularity of Songs

Chebat, Deeb, DeSanctis, Odayappan, Odayar

December 2022

1 Introduction and Background info

Predicting song popularity is a topic that has been studied extensivley in the past. In particular, A Stanford Research Project attempted to predict the popularity of a song. They favored an approach through Natural Language Processing (NLP), which we found very convoluted, further motivating our excitement behind the project idea. (Pham et al.) In this paper, we wish to create our own model the acts as a prediction algorithm in order to see how predictable the music industry is with respect to a top streaming platform, Spotify. Within our model, we used a variety of different methods: forward selection, lasso regression, random forest, and AdaBoost. Before we introduce our models, we will first go into depth about how each one of these works.

To begin, let's introduce forward selection which is used to find the most important features of a dataset. Forward selection is a greedy algorithm, which takes the optimal choice at each step in order find the absolute optimum. The algorithm works by finding the feature that improves the model best, selects it as the first, and adds it to the model. Then, the next variable that provides the maximum improvement is selected and added. This process is repeated until further improvements cannot be made or a pre-defined stopping criterion is met.(Zach, 2022)

Next, let's cover lasso regression. Lasso regression is a regularization method that uses shrinkage to increase the accuracy of a model. It does so using an L1 penalty that penalizes the model against complexity, thereby reducing the variance. As a result, lasso is an important method that can be used to prevent overfitting and make our model more accurate when predicting new data.(Kumar, 2022)

Another technique we used in our models were random forests, which is an ensemble method that utilizes multiple decision trees to make a guess about a data point's prediction. For each tree, only a subset of the data is utilized, which allows the method to find patterns across the entire set, and prevents overfitting. Then, an average or majority vote is used to provide an estimated guess for the point.(Yiu, 2019)

The final method we implemented was AdaBoost, which is another ensemble method that uses a number of decision tree weak learners to progressively train and improve, allowing for a strong model to arise. The model adjusts each tree sequentially, changing weights in response to errors, before ultimately using the series of created trees to create a prediction by giving each tree a weight based off its strength.(Kurama, 2019) Our understanding of AdaBoost was solidified by an analysis provided by the Institute of Electronics and Electrical Engineers (Solomatine and Shrestha, 2005).

Aside from providing a short introduction to the models, we also wanted to use this section to outline the motivation behind doing this project. When we spotted this option, we immediately became intrigued by the idea of being able to predict the success of a song. At first, we wanted to predict the number of streams a song would reach. However, due to restrictions in the Spotify API, we did not have access to the streams data. Instead, we had to focus on a new question, centered around the popularity data provided to us. We revised our question to how popular will a song be, which was much more qualitative

in nature than the streams questions and allowed us to use the data available via the Spotify API. The data we collected covered various decades, which naturally led us to want to view the data in light of a time series in order to see how well the data could be predicted using previous decades. Alongside that, we recognized that the patterns could pertain more to particular genres than all songs overall, and thus involved a prediction with genre.

Ultimately, this project was a very educational and exciting experience in which we not only gained furhter insight into the various methods we had learned in class, but also got to apply them to a concept that would have seemed impossible for us to do just 3 months ago.

2 Data Collection

Our data collection was performed in two steps. The first involved finding a pre-collected dataset of various songs separated by decade. We were able to find such a set on Kaggle under the title "The Spotify Hit Predictor Dataset (1960-2019)", available at **https://www.kaggle.com/datasets/theoverman/the-spotify-hit-predictor-dataset**.
This initial dataset contained many useful metrics for song measurement. More specifically, it contained the following:

Table 1: Kaggle Database Description

Feature	Description	Type
track	Name of traack	string
artist	Name of artist	string
uri	Song ID for Spotify API	string
danceability	Combined factors to rank "excitment"	real on [0.0, 1.0]
key	Key of song	Positive Integer, -1 if none found
loudness	Avereage db measurement	Real on [-60, 0]
mode	Minor or major	binary (0, 1)
speechiness	Presence of human speech	Real on [0.0, 1.0]
acousticness	Confidence level that song is acoustic	Real on [0.0, 1.0]
instrumentalness	Confidence level that there are no lyrics	Real on [0.0, 1.0]
liveness	Confidence level that audience is in track	Real on [0.0, 1.0]
valence	Measure of human positiveness	Real on [0.0, 1.0]
tempo	Estimated BPM	Positive real
duration_ms	Length of song in ms	Positive integer
time_signature	Beats per measure	Positive integer
chorus_hit	Estimate of when chorus starts	Positive integer
sections	Number of sections in a song	Positive integer
target	Custom classification of whether or not a song was a hit given its performance in a given genre and time frame	Binary [0, 1]

While this data is extensive, it fails to include the popularity ranking or song genres. In response, we made a custom script that connects to the Spotify API and makes calls for each song. These calls gather the song's popularity and the genres of the song's artist. In the end this adds two more features:

Table 2: Script Features

Feature	Description	Type
popularity	Spotify API's measure of popularity	Positive integner in [0, 100]
genres	List of possible genres the song could belong to	List of strings

Please email us or send us a Canvas message if you are unable to view any code throughout this document

3 Preliminary EDA, Cleaning, & Visualizations

Initially, our dataframe contained 41,106 songs and 17 quantitative features for each one. However, the Spotify API repeated some songs when exporting them by decade. To fix this, we removed duplicate songs across decades, leaving 40,003 songs remaining. In total, there are 11,904 unique artists in the dataset. The number of songs by decade is displayed in Table 3 below.

Table 3: Number of Songs by Decade

	No. of Songs
1960	5730
1970	5456
1980	8516
1990	7564
2000	6125
2010	6612
Total Songs	40,003

The first step in our EDA was to explore how the covariates are related (or not related) to our measure of popularity. Given the size of our dataset, scatterplots are too noisy and crowded to be a valid option. As a result, we chose to make binned scatterplots that display the average value of the predictor for each value of popularity. In addition, we show the standard errors for each point.

Figure 1:

Predictors Plotted Against Popularity

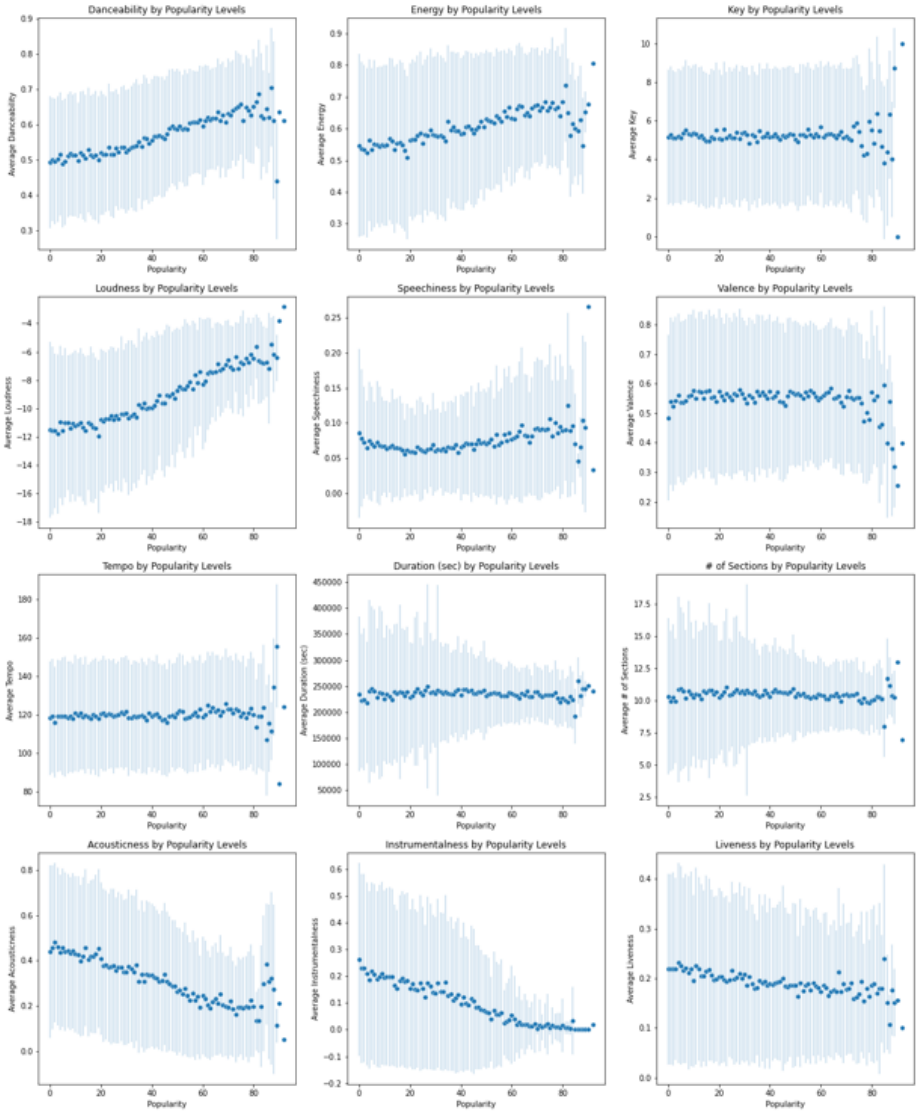
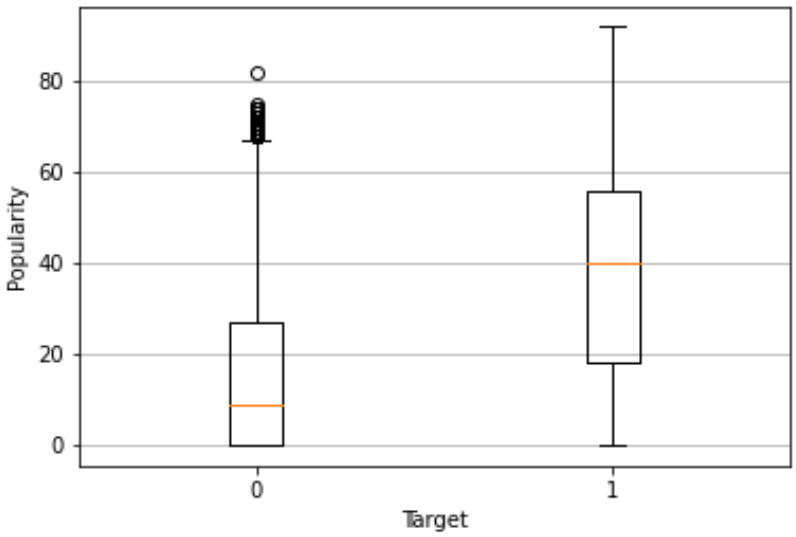


Figure 1 shows that Danceability, Loudness, and Energy seem to have a positive linear relationship to popularity. As these metrics increase, songs, on average, tend to be more popular. On the other hand, Acousticness, Instrumentalness, and Liveness are negatively related to popularity. For many graphs, there is volatile end behavior, which can likely be attributed to the fact that there are only a few songs with very high popularity levels. As a result, the small sample size may be leading to increased noise in the data. Similarly, the standard error bars are wide for all the graphs, also suggesting that the data is quite noisy and one feature alone will not be able to accurately predict popularity.

Additionally, our dataset includes a binary variable called "Target" that takes the value of 1 if a song has been featured in the decade's weekly list of Hot-100 tracks at least once. Although this metric is strongly correlated with popularity, we will likely not include it in our model since our model attempts to understand song attributes lead to popularity, but the target variable seems to simply be a byproduct of a song's popularity. A boxplot showing the relationship between Target and Popularity is shown below in Figure 2.

Figure 2:

Relationship Between Target and Popularity



4 Quantitative Exploration and Modeling

We begin our modeling with a set of arguably naive models that treat the popularity variable as a quantitative measure from zero to one hundred, inclusive. However, these models do not account for the fact that popularity is bounded. Thus, it may estimate popularity to be below zero or above one hundred for a given set of predictors. This problem will be addressed in the next section.

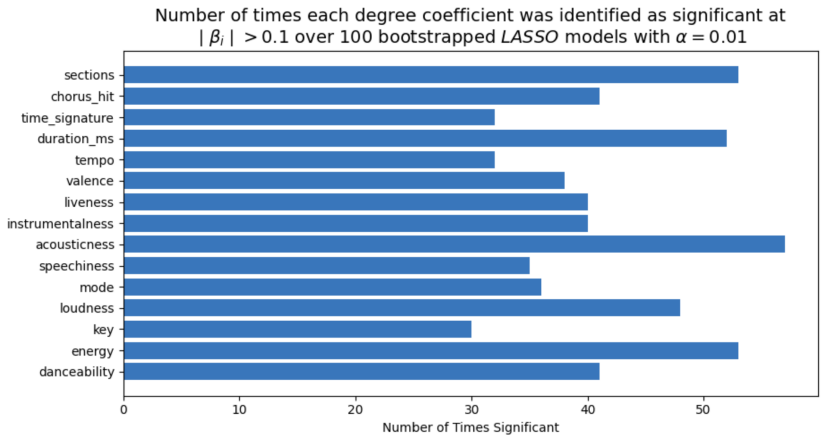
We wanted to first understand how the features are linearly related to popularity. However, including all combinations of features would be 2^j models with j being the number of predictors. Instead, we chose to use a forward selection model. We built a script in the attached code file that implements this approach. We start by looping through all of the predictors and fitting a linear regression for each. Then, we picked the one that had the lowest validation mean squared error and called it Model 1 (M_1). Following the same process, we looped through all the other variables and added them one by one to M_1 and picked the new two-variable model with the lowest validation mean squared error (M_2). After repeating this for all the features, we had a list of models $\{M_1....M_{14}\}$. These models are shown in the table below.

Table 4: Forward Selection Multivariate Model

New Variable	Val. MSE	No. of Vars
loudness	485.0064694	1
danceability	468.5588813	2
valence	459.6249305	3
instrumentalness	450.0324565	4
acousticness	444.200251	5
liveness	438.8054133	6
energy	435.3666393	7
speechiness	434.4266104	8
tempo	434.1300263	9
time_signature	433.9107343	10
sections	433.791851	11
duration_ms	433.788294	12
mode	433.7862114	13
key	433.7871966	14

Table 4 shows each model in $\{M_1....M_{14}\}$. To clarify, each row does not represent a linear single-variable model but rather each row represents a multi-variate model with all the variables from the rows above. The variables that are initially the most important in the forward selection (loudness, danceability...) also show predictive trends in the binscatter plots from the EDA in Figure 1. The best forward selection model is the one with all the variables except key. However, the models are very close to one another suggesting that adding each additional variable is not adding much predictive power.

Next, we used lasso feature selection to understand which features are most important. We first standardized the data so that they are all of a similar magnitude. Then, we bootstrap (sample with replacement) the data 100 times and fit a lasso model each time. Each time, we tracked the variables whose coefficients have absolute values above 0.1. The following figure shows the results of the feature selection.



Looking at the x-axis, there is surprisingly not as much variation as one would expect. In a dataset where some variables are highly predictive and some are not, we would expect to see some features be significant in a vast majority of the 100 models and some significant in virtually none of the models. This is somewhat in line with the forward selection model where we did not see significant variation in

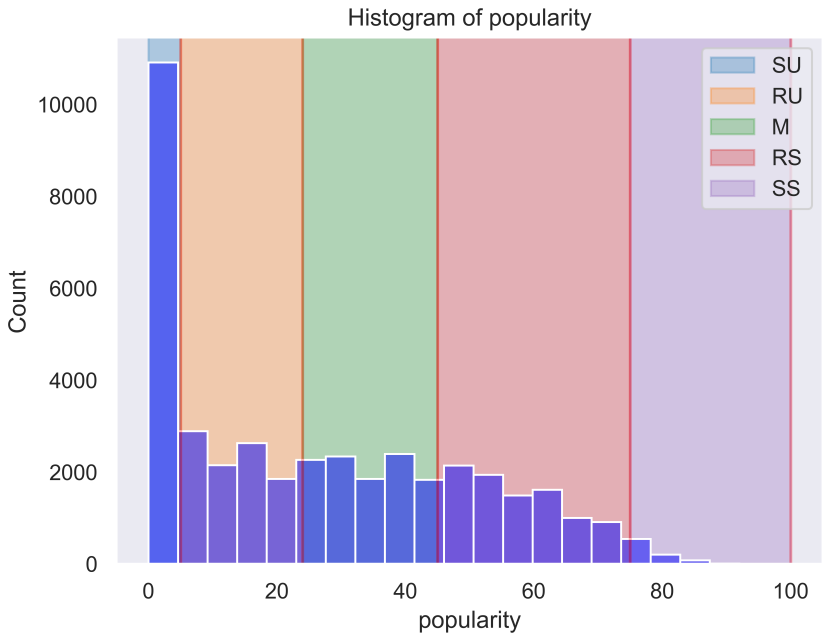
validation MSE. Based on looking at the bar plot, we decided on using a threshold of 45 to select the most important features. This resulted in a multilinear model with energy, loudness, acousticness, duration, and sections. Using the non-standardized data for comparison purposes, this model had a validation MSE of 460.661 - which is significantly worse than the forward selection best model. Thus, we conclude that our best multivariate linear regression model is the forward selection model identified in Table 4. Next, we will run more sophisticated models taking into account the bounded nature of popularity.

5 Categorical Exploration and Modeling

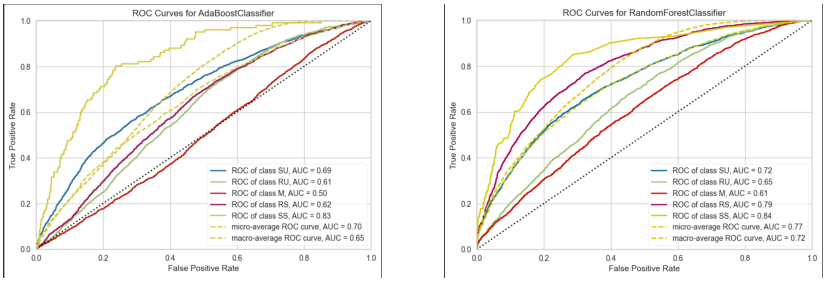
Given the wide range of popularity values, it may be beneficial to instead transform popularity to a categorical variable. Taking a look at the distribution of popularity, you can see that songs with very low scores are quite common, while the top songs are quite rare. Therefore, we can set these two to be their own category and apply a uniform separation to the rest. Here, label the categories by two letters "R/S" for relatively/significantly, and "U/S" for "unsucessful/sucessful." M is then the "moderate" category.

Figure 3:

Popularity Split



As we are dealing with a multi-class regression with many data, it's best to narrow the modeling down to AdaBoost and RandomForest (we choose RanomdForest over bagging, since we will need a large number of estimators given the large data set). Using AdaBoost Cross validation, we can compare the affect of learning rates and estimators across models.

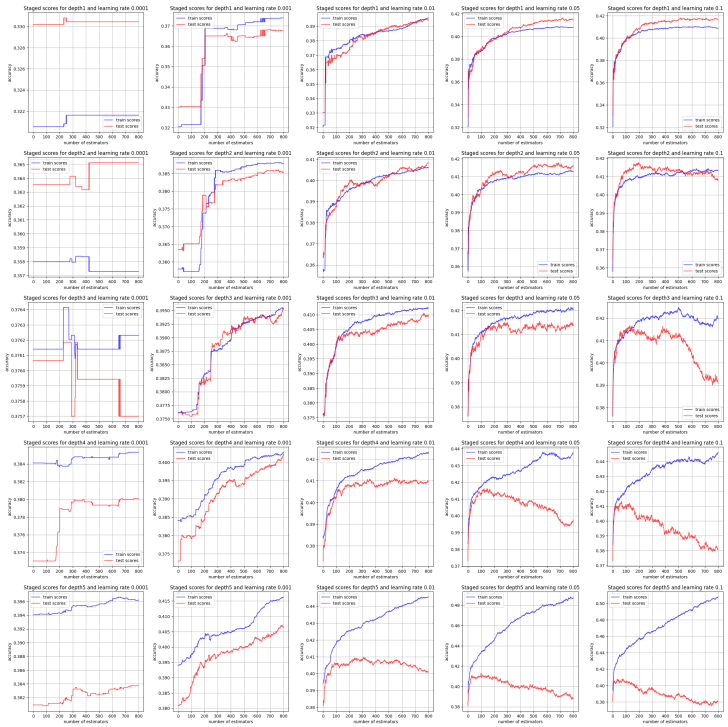


(a) AdaBoost **(b)** RandomForest

Figure 5: ROC curves comparison

Figure 4:

AdaBoost CV

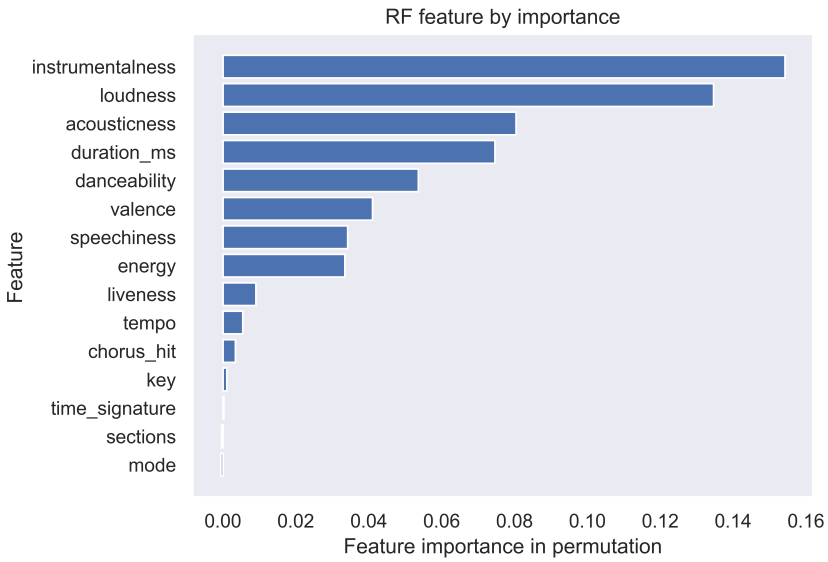


A script running over these results determines the best result at max depth 3, learning rate 0.05, and 290 estimators. This gives around 0.41 accuracy. A RandomForest trained on the balance of this data performs slightly better at 0.42 accuracy with 400 estimators and a max depth of 30. Upon comparing the ROC curves of both models, the RandomForest model seems to produce better results, having higher AUC scores on average and not over-fitting to certain outputs like AdaBoost does.

With the selection of Random forest, we may also inspect the permutation importance of the features to determine what the greatest predictors are (more specifically, what gives the random forest classifier the greatest increase in accuracy). From what is available, it seems that features more sepcific to the song tend to score higher than general metadata (with the exception of duration). Aspects such as instrumentalness, loudness, and danceability seem to add more to a song's success than its key, mode, or tempo. Logically, this makes sense, or all major pop songs would be written in the same key!

Figure 6:

Random Forest Feature Permutation



Strengths Splitting into categories gives the model a better overall accuracy when it comes to predicting labels. Furthermore, it does so at over twice random odds with $p << 0.01$. Random Forest permutations gives us some semblance of an idea on how important certain characteristics are to a song's success and, even though model running took substantial computational power, the models themselves are quick to run given song information.

Weaknesses and Limitations An accuracy of 0.42 is not admirable. Given more time (and more computational power), we would like to rerun the random forest on subsets of predictors (or even more) to limit co-linearity and increase the available attributes. The breakdown of categories is also subjective in this regard despite the attempt to reflect how the data is represented in our data set.

6 Time-Series EDA and Analysis

We were interested in exploring how the popularity of various features of songs changed across the decades included in our data set (1960s to 2010s). We utilized two of the models explored above: AdaBoost and Random Forest. From the best model found above, we used 290 estimators and a depth of 3 for AdaBoost. For random forest, we used 400 estimators and a maximum depth of 30. A table displaying the accuracy scores for each of these two models by decade can be found below. In analyzing the table, we can see that the range of accuracy scores for both the AdaBoost and Random forest models tends to fluctuate in the range of 0.40 to 0.43, which is not a high accuracy score as mentioned in the above paragraph which discusses potential future directions.

Figure 7:

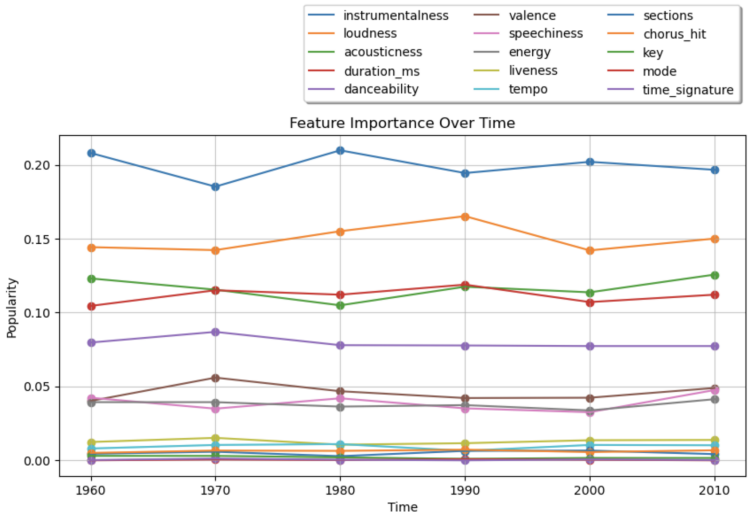
Accuracy Scores By Decade for AdaBoost and Random Forest Models

	Decade	Accuracy Scores Using AdaBoost	Accuracy Scores Using Random Forest
0	1960s	0.400876	0.416687
1	1970s	0.402578	0.423984
2	1980s	0.398930	0.418511
3	1990s	0.403916	0.426295
4	2000s	0.403430	0.422890
5	2010s	0.405984	0.430431

We started by computing the permutation importance in order to better understand how randomly shuffling each predictor variable would impact the model accuracy. Then, we found the permutation importance, which directly measures variable importance by observing the effect on model accuracy of randomly shuffling each predictor variable. Following this step, we looked at feature importance to better understand how the popularity of features of songs changed across the decades. The results are depicted via a line graph as found below:

Figure 8:

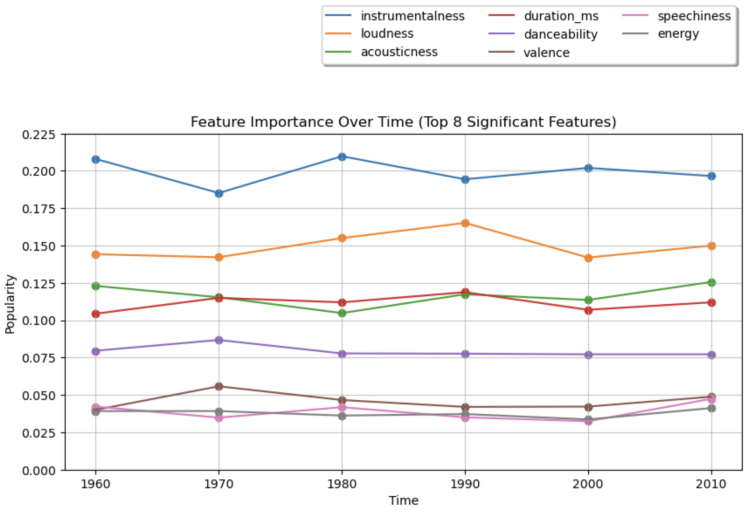
Changes in Feature Popularity over Time



We can see from this figure, that there are many features which exhibit very little change in popularity over time including liveness, tempo, sections, chorus hit, key, mode, and time signature. When removing these features, we chose to include 8 features which seemed to depict more significant changes in popularity across the decades. The results are depicted below:

Figure 9:

Changes in Feature Popularity over Time: Top 8 Features



We can see from this figure that the popularity of instrumentalness and loudness seem to change and fluctuate from decade to decade rather significantly. Similarly, we can see the fluctuations in popularity of the other features over time as well.

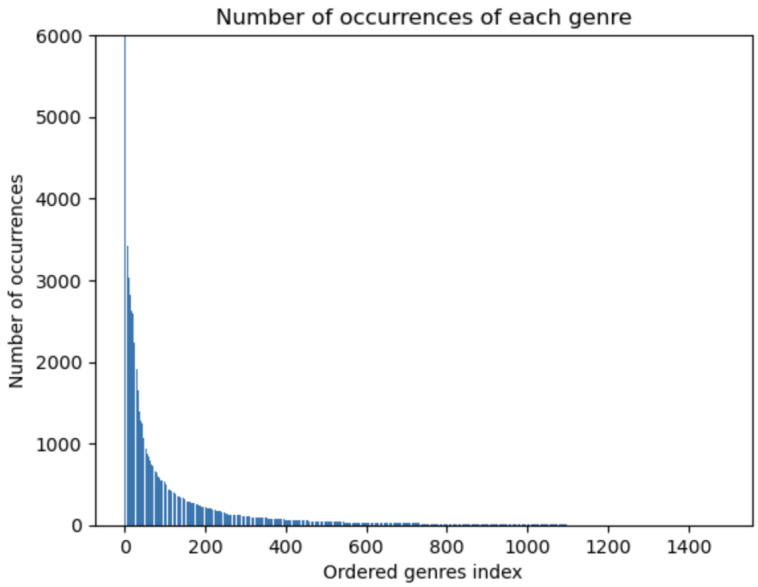
7 Prediction with Genre

Thinking about how different genres' listeners tend to have different listening habits, we wanted to see how adding genre dummies would affect the model. We used the genre column in the database to extract the genres of each song. A lot of genres tend to be a mix of two genres, "Canadian Pop", for example. Therefore, we have separated genres into words and analyzed the frequency of occurrence for each genre.

Here's a plot of the number of occurrences of each genre:

Figure 10:

Occurrences of Genre Indices



We can see that the vast majority of genres are relatively rare. Therefore, we have only used the top 50 genres to prevent overfitting.

Given our results from the previous model, we have decided to use AdaBoost with a base estimator depth limit of 4, a learning rate of 0.01 and 530 estimators. We have got improved accuracy scores; for the categorical prediction, we got an accuracy score of 0.44.
In the future, we hope to be able to use cross-validation to find the optimal hyperparameters for the model with genre dummies and inspect how the random forest scores compare to the current ones.

8 Conclusion and Implications

From these models, we found the, given the features used, popularity can be predicted better than random chance with statistical significance, but it is still a very difficult feature to model. Our best accuracy and correlation coefficients stayed within a 0.3-0.45 range for the most part and, even with intense cross validation and scaling, we were unable to increase it.

Still, this does not mean that our modeling did not reveal anything. We found that categorical modeling may help in the effort to predict popularity by shrinking the outputs to a 5-classification. Furthermore, genre and time seem to have some bias in how popularity trends over time and how Spotify uses popularity as a measuring metric.

Going forward, more research should be conducted into the effect of genre and time on popularity jointly, as well as the best method to split popularity categorically without abstracting too much. If done correctly, we believe a better accuracy may be achieved on a given test set. Furthermore, we suggest using the song-analysis API offered by Spotify. Due to time constraint and a large dataset, we were unable to run this analysis on enough songs, but this joined with removing some less important features (such as mode) may indeed pave the way to a better model.

9 References

1. Kumar, Dinesh. "A Complete Understanding of Lasso Regression." Great Learning Blog: Free Resources What Matters to Shape Your Career!, 31 Oct. 2022, https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/#:~:text=Lasso%20regression%20is%20a%20regularization,i.e.%20models%20with%20fewer%20parameters).

2. Kurama, Vihar. "A Guide to Understanding AdaBoost." Paperspace Blog, Paperspace Blog, 9 Apr. 2021, https://blog.paperspace.com/AdaBoost-optimizer/.

3. Pham, et al. Predicting Song Popularity - Stanford University. http://cs229.stanford.edu/proj2015/140_report.pdf.

4. Solomatine, D.P., and D.L. Shrestha. "AdaBoost.RT: A Boosting Algorithm for Regression Problems | IEEE ..." IEEE Xplore, 17 Jan. 2005, https://ieeexplore.ieee.org/document/1380102/.

5. Yiu, Tony. "Understanding Random Forest." Medium, Towards Data Science, 29 Sept. 2021, https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

6. Zach. "What Is Forward Selection? (Definition & Example)." Statology, 13 May 2022, https://www.statology.org/forward-selection/.

7. Ansari, Farooq. "The Spotify Hit Predictor Dataset (1960-2019)." Kaggle, 25 Apr. 2020, https://www.kaggle.com/datasets/theoverman/the-spotify-hit-predictor-dataset.