Beta Records: Finding the Trends in the Top 100 Songs

Phillip Bagley, Abhishek Birhade, Hyundae Kang,
Beau Ritter, and Alex Tadros
Department of Mathematics and Statistics,
Georgia State University, Atlanta, GA, USA

May 3, 2019

1. Abstract

The motive of this study is to create a machine learning algorithm that can predict the success of a given song on Billboard charts. Using python API tools we ammased a data of roughly 80,000 songs, of which over 3000 made it to billboards Hot-100 chart. We used the sae tools to collect our audio features for each song and used python to deploy a logistic regression. Based on this information our model can predict if a song can make it to the aforementioned chart with an accuracy of 60%.

2. Introduction and Background Research

With the emergence of trap and rap as the major genres of this decade it is apparent that a great new song by popular artist in this genre is likely to make it to the Hot-100 chart. This leaves a lot of room for other artists. and begs the question, how does a new song become popular? At Beta Records, we set out to answer these questions. Empowered with the tools of statistics and computer programming, we were determined to investigate if there is a statistical trend that we can trace. Our objective for this research was to build a machine learning model that can classify/identify if a can make it to the Hot-100 billboard chart.

Our initial discussions for this research revealed that the study that most resembeled our objective was "Hitpredict: Predicting Hit Songs Using Spotify Data" by Elena Georgieva, Marcella Suta, and Nicholas Burton [1]. At first we were impressed with their results. Their model seemed to be able to predict the popularity of a song with a high amount of accuracy. However, upon further research we noticed a few flaws in their approach. Namely, their dataset was fairly small with 4000 songs. We noticed they had not mentioned how many of these songs made it to the billboard charts. We concluded that the small size of this dataset would lead to some selection biases that can not accurately reflect the large state of the mainstream music industry. We also noticed that release dates for the songs they selected varied a fair amount. We concluded that such varying dates will have a good contribution to the results if one is trying to predict changes in the industry. We are of the opinion that songs of today reflect the historical transformation of music and popular genres. Furthermore, we

do not think songs of our parents generation are similar in structured and melody to songs of this day and age. Thus, we decided to restrict our dataset to songs that have been released in the past decade. We proceeded to expand upon the work of the Hitpredict team. Other studies have attempted to predict Top 10 Dance Hits [2] and to classify songs based on lyrics [3]. Such studies seemed restrictive in their aproach and objective to us.

3. Methods

3.1 Data Collection

We proceeded to collect data from Hot-100 Billboard chartsand Spotify. Manually scraping songs would have been quite hectic so we decided to use a popular scraper for python called Billboard designed by Allen Guo[4] and Spotipy which is a lightweight python API [5]. Data from billboard consisted of some statistics such as weeks on chart and such, we decided to drop these features deeming them unecessary. After this we scraped spotify using spotipy for about 10,000 songs from each year, going back a decade to the year 2009. The next step was to clean our data and look for corrupt and duplicate entries in our data set. This left us with a fairly sizeable dataset of roughly 78000 songs. Our feature set consists of fourteen fields, of these 9 were continuous (danceability, energy, speechiness, acousticness, instrumentalness, liveness, valence, loudness, and tempo) and 5 were discrete (key, mode, duration, time signature). Of the continuous featuresx 7 had variables between [0,1] while loudness and tempo had larger ranges. These features are determined by audio analysis software created and deployed by Spotify.

Following this, the final phase was to merge the two data sets. We decided to do a simple VLOOKUP in excel and a pandas lookup in python. For most songs with a single artist this was sufficient. However, upon close inspection we noticed that collaborations were note being identified. We tackled this by manually looking up all collaborations in the dataset by splitting the dataset amongst us. Finally we flagged songs in our original dataset as 0 or 1, 0 for a failure and 1 for sucess to make it to the chart. Due to the way Spotipy works, making individual search queries for each billboard song to obtain its features would not have been a very fruitful method and would have time consuming. Each request return a list of songs of the same name by different artists, including remixes and covers, as well as songs that were released on different continents by the same record label. As a consequence, we had to work with billboards songs that were already in our dataset, which was roughly 3300. This left us with about 74,700 songs as Not Hot.

3.2 Data Exploration

Our next step was to attempt to determine if there existed statistically significant difference between the means of each feature of the Hot and Not Hot datasets. Since we had such a large dataset, using a t-test and using the reported p-value would be unwise, as it is possible that even an incredibly slight difference in mean could still be reported with very strong confidence Instead, we elected to use the t-test to calculate a 95% confidence interval for where we believe the true difference in means lies. We then divided the bounds of these confidence intervals by the range of each feature to create a magnitude interval. This magnitude interval is how large of a percentage difference we expect between the means of the two populations. For example, a feature with a magnitude interval of 30–35% would indicate

that the mean of the Hot songs for that given feature is expected to be 30-35% higher than the mean of the Not Hot songs for that feature. This allows us to see not only statistical significance, but also if that significance is truly meaningful.

Feature	Confidence Interval, $\mu_1 - \mu_2$ ($\alpha = .05$)	Magnitude Interval
Danceability	(0.0768, 0.0792)	7.778 – 8.024%
Energy	(0.0413, 0.0446)	4.131 – 4.462%
Loudness	(1.789, 1.855)	2.871 – 2.977%
Speeciness	(0.012, 0.0136)	1.197 – 1.359%
Acousticness	(-0.231, -0.226)	22.707 – 23.145%
Instrumentalness	(-0.136, -0.132)	-13.200 – -13.610%
Liveness	(-0.107, -0.105)	-10.468 – -10.707%
Valence	(-0.078, -0.0746)	-7.463 – -7.817%
Tempo	(-55.581, -56.158)	-25.281 – -25.472%
Duration	(-10666.596, -9153.214)	-3.910 – -4.556%

Table 1: The convidence and magnitude intervals the difference of means; μ_1 = Hot mean, μ_2 = Not Hot mean.

As we can see by Table 1, even though the difference in means is statistically significant for each variable (no interval contains 0), many of these values have low expected magnitude. This means that the actual difference in means is likely too small to be actually worthwhile and it is possible that the differences come from issues in sampling or random assortment, which could be detrimental to the machine learning process. We experimented with dropping features with values that had an expected magnitude of less than 5%, but this wound up decreasing our programs accuracy instead of increasing it, likely because removing these entries lowered the number of test features below the an appropriate level. Optimally, we would try to locate and determine additional features outside of the Spotify API that displayed higher magnitude. However, due to time constraints and data limitations, we decided our best course of action would to be to stick with the variables we had and hope that the results we had generated were still meaningful.

3.3 Machine Learning

When deciding on which ML algorithm to use for the predictive stage of the process, we decided on focusing on using logistic regression for a number of reasons. First, the assumptions necessary for logistic regression to be effective (all variables must be independent, the dependent variable assumes only a binary value of 0 or 1, does not require a strictly linear relationship between dependent and independent variables, ect.). Secondly, the study done by Burton, Georgieva, and Suta found that logistic regression was one of the two most effective algorithms in classifying their data [1]. Even though we were using a different data set than they were, we concluded that, since our problem was similar, following their method would likely be effective. Finally (and honestly), building a logistic regression algorithm seemed significantly more accessible when compared to the other options we considered. As it was our first ML project as a group, we had to be realistic and work within our capabilities.

The logistic regression algorithm uses Newton's Method for gradient descent to optimize the weights. The final equation of a logistic regression model will resemble

$$Log(\frac{Y}{1-Y}) = b_0 + b_1X_1 + ... + b_nX_n$$

where $Log(\frac{Y}{1-Y})$ represents likelihood (our dependent variable), b_i represent weights (b_0 is the bias/error weight), and X_i represent the features of a given sample. When preparing our data to be used in the algorithm, we generated dummy variables to translate the discrete integer values of Key and Tone to binary values. Additionally, to balance our dataset, we used SMOTE (Synthetic Minority Over-sampling Technique) to generate synthetic Hot samples using the true values as a basis.

4. Results

Due to the nature of how the logistic regression algorithm we selected iterates its weights, our program converged on its optimal values after just 8 iterations. The coefficients of each of the 20 real and dummy variables can be found in Table 2. Coefficients with positive values and positively correlated to likelihood, while negative values indicate a negative correlation. Coefficients with high absolute value are deemed to be highly correlated, while those with values close to or equal to 0 are seen as uncorrelated. The algorithm we developed saw danceability, energy, and instrumentalness to be by far the most important predictors to a song's success on the chart.

Feature	Coefficient	Feature	Coefficient
Danceability	2.7804	Key_2.0	-0.0445
Energy	-2.2225	Key_4.0	-0.003
Loudness	0.1862	Key_6.0	0.354
Speeciness	0.5511	Key_8.0	0.3941
Acousticness	-1.8273	Key_9.0	.0018
Instrumentalness	-4.8608	Key_11.0	0.2426
Valence	-1.1947	Mode_1.0	0.2611
Tempo	.0064	Time Signature_1.0	0.2598
Duration	0	Time Signature_3.0	0.3877
Key_1.0	0.3138	Time Signature_4.0	1.0822

Table 2: The coeffecients of each feature in our final algorithm

In measuring the algorithm's accuracy, a test set of 25% of our observations were ran through the algorithm and classified as Hot or Not Hot. A confusion matrix (Figure 1) was generated to determine the accuracy of the program. A final accuracy of 59% was given to our set of coefficients, indicating that these weights have a 59% chance of correctly classifying a given song in our test set as Hot or Not Hot. Interestingly, it seems our program is slightly more accurate at predicting which songs will not make the charts than which songs will.

A final test of algorithm's merits is the ROC curve (Figure 3). The ROC curve measures the stability of a given algorithm by charting how the false positive rate correlates to the

n=31,350	Predicted "Not Hot"	Predicted "Hot"	
Actual: "Not Hot"	TN = 8,499	FP = 7,151	15,650
Actual: "Hot"	FN = 5,690	TP = 10,010	15,700
	14,189	17,161	

Figure 1: The Confusion Matrix

true positive rate as the threshold for acceptance increases. As the curve moves left, the rate of true positives increases, but as does the rate of false positives. The threshold is decided upon at the value which maximizes the true positive rate while minimizing the false positive rate. When comparing effectiveness, an area under the ROC curve closer to 1 represents a more stable and generalizable machine learning algorithm. The value of the area under our ROC curve was 0.59, the same as our accuracy value. While these measurements are closely correlated and will take the same value, they are not the same thing.

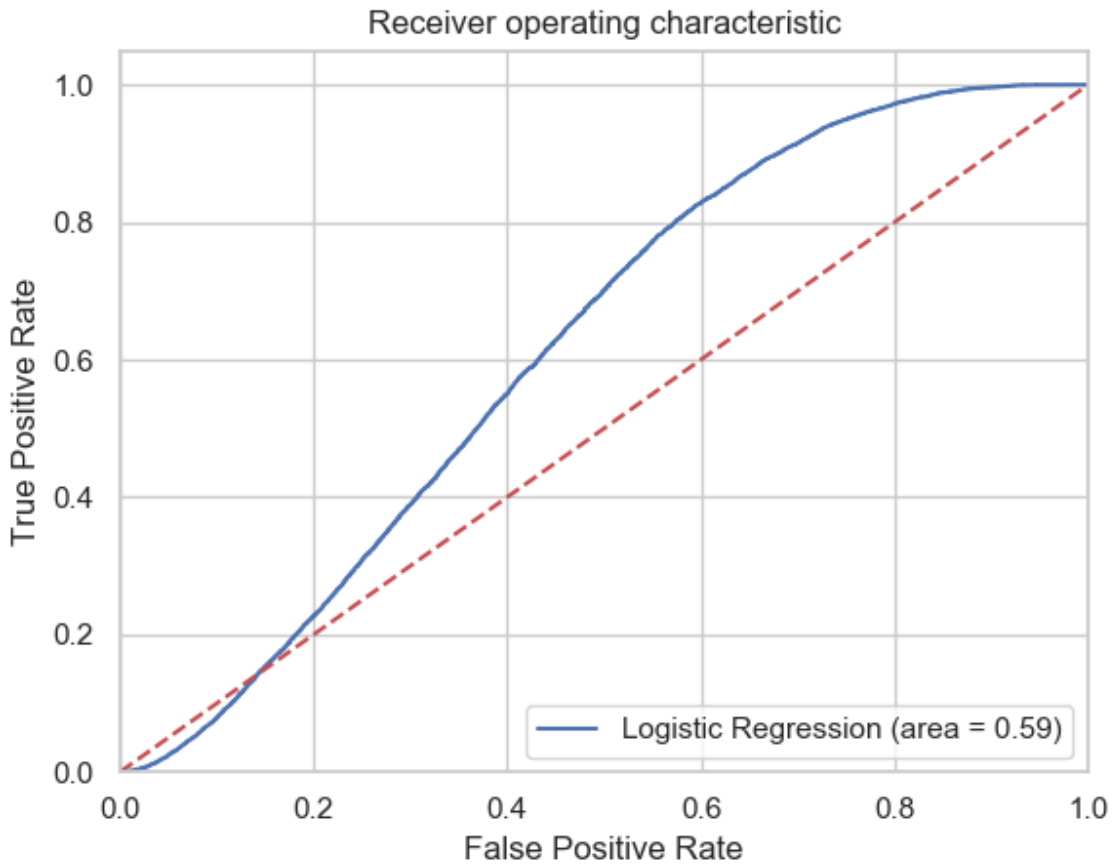

Figure 2: The ROC curve. The blue line represents our algorithm, while the red line is just present for comparison

5. Discussion and Conclusion

This is the first study that considers such a large data set for the purpose of analysis. While our study did not yield as significant results some of the other metioned ones, the size of our data set is what sets us apart from them. That being said, there are many interesting results that our study had produced. We noticed that Hot songs have a lower acousticness value as compared to Not Hot songs. This is a definitive feature that we noticed and by comparing this particular feature to a different dataset in a different time we can argue that trends that make something popular overtime in a particular category have changed.

I would like to expand on my work by including more features in the model. The relationship of math with music has been studied very well and I would like to see how other features of a song affect its popularity. Such features could include the use of overtones and most importantly the genre. The only issue I forsee is how readily such data is available. Furthermore, I would like to compare my algorithm with other ones such as Random Forests and KNN using that new data.

Additionally, I would also like to study how language of lyrics affect the popularity of a song. Since the Hot-100 chart ranks songs by airplay in America, it is easy to deduce that songs written in English are most likely to make it to the chart. That does not totally disqualify songs written in other langauges, for example Daddy Yankees songs feature spanish lyrics and he still makes it to the chart often. Another example would be the Korean Group BTS. It would be extremely intruiging to explore this avenue. Does a foreign artist have to be a well known for their songs to become Hot? Or do they have to collaborate with a well known American artist?

6. References

[1] Georgieva, Elena, Suta, Marcella, and Nicholas Burton. Hitpredict: Predicting Hit Songs Using Sporify Data. Stanford Computer Science 229: Machine Learning, 2018.

[2] Chinoy, S. and Ma, J. .Why Songs of the Summer Sound the Same. New York Times, 2018.

[3] Singhi, Abhishek, and Daniel G. Brown. Hit song detection using lyric features alone. Proceedings ofInternational Society for Music Information Retrieval, 2014.

[4] Billboard. Billboard Charts Scraper. (2019) Retrieved from: https://github.com/guoguo12/billboard-charts

[5] Spotipy. Spotify Python API. (2019) Retrieved from: https://github.com/plamere/spotipy

[6] Filion, Guillaume. The curse of large numbers (Big Data considered harmful). The Grand Locus, 2018. http://blog.thegrandlocus.com/2018/02/the-curse-of-large-numbers-big-data-considered-harmful

[7] Li, Susan. Building a Logistic Regression in Python, Step by Step. Towards Data Sciene, 2017. https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8