HITPREDICT: PREDICTING HIT SONGS USING SPOTIFY DATA

PIYUSH RAJ

ABSTRACT

In the current study, we approached the Hit Song Science problem, aiming to predict which songs will become Billboard Hot 100 hits. We collated a dataset of approximately 4,000 hit and non-hit songs and extracted each songs audio features from the Spotify Web API. We were able to predict the Billboard success of a song with approximately 75% accuracy on the validation set, using five machine-learning algorithms. The most successful algorithms were Logistic Regression and a Neural Network with one hidden layer.

1. INTRODUCTION

The Billboard Hot 100 Chart [1] remains one of the definitive ways to measure the success of a popular song. We investigated using machine learning techniques to predict whether or not a song will become a Billboard Hot 100 hit, based on its audio features. The input to each algorithm is a series of audio features of a track. We use the algorithm to output a binary prediction of whether or not the song will feature on the Billboard Hot 100.

This research is relevant to musicians and music labels. Not only will it help determine how best to produce songs to maximize their potential for becoming a hit, it could also help decide which songs could give the greatest return for investment on advertising and publicity. Furthermore, it would help artists and music labels determine which songs are unlikely to become Billboard Hot 100 hits.

2. RELATED WORK

The initial idea for this research project stemmed from a New York Times article that used the Spotify audio features to illustrate the similarity of summer songs [3]. Music technology companies such as The Echo Nest, ChartMetric, and Next Big Sound have been using data analytics to help artists and labels predict and track a song's success for almost a decade. This problem is referred to as Hit Song Science (HSS) in the Music Information Retrieval (MIR) field.

Machine learning is a popular research and industry tool to approach the HSS question. Researchers have used Convolutional Neural Networks [10] and K-Means Clustering [6] to predict pop hits. Both of these studies were engaging and successful, but focused more heavily on the signal-processing involved in audio analysis.

Another group of researchers used Support Vector Machines (SVM) to predict top 10 Dance Hits [4]. By narrowing the scope of the study to only dance music, researchers were able to present a more focused work. Another study attempted to classify songs based on lyric content [7]. While they successfully classified many hits, they also returned many false positives and concluded that analyzing lyrics is an ineffective approach to this problem.

3. METHODS

3.1 Dataset and Features

A dataset of 10,000 random songs was collected from the Million Songs Dataset (MSD) [9], a free dataset maintained by labROSA at Columbia University and EchoNest. This was narrowed down to songs released between 1990 and 2018. Next, we collected a dataset of all unique songs that were featured on the Billboard Hot 100 between 1990-2018, using the Billboard API library [2]. The datasets provided the artist name and song title, as well as other miscellaneous features. To balance the dataset between positive (hits) and negative (non-hits) examples, we removed two thirds of the songs collected from the Billboard Hot 100. Finally, we removed overlapping songs to form a dataset of approximately 4,000 songs.

Tracks were labeled 1 or 0: 1 indicating that the song was featured in the Billboard Hot 100 (between 1991-2010) and 0 indicating otherwise. Next, we used the Spotify API to extract audio features for these songs [8]. The Spotify API provides users with 13 audio features, of which we chose nine for our analysis: Danceability, Energy, Speechiness, Acousticness, Instrumentalness, Liveness, Valence, Loudness, and Tempo. The first seven features are represented as values between 0 and 1 by Spotify. Loudness is measured in decibels and tempo refers to the speed of the song in beats per minute.

To account for artist recognisability, we defined an additional metric: the artist score. Each song was assigned an artist score of 1 if the artist had a previous Billboard Hot 100 hit, and 0 otherwise. We looked back to 1986 for this metric. There is some inherent inaccuracy in this measure. If an artist had a hit song before 1986, but not after, they were given an artist score of 0.

3.2 Algorithms

To predict a song's success, we used six different machine-learning algorithms: Expectation Maximization (EM), Logistic Regression (LR), Gaussian Discriminant Analysis (GDA), Support Vector Machines (SVM), Decision Trees (DT), and Neural Networks (NN). We focused mainly on the accuracy of results, but we report the precision and recall as well. False positive predictions may be costly if a music label invests in a song that is unlikely to become a hit.

For an initial identification of clusters in the data, we used the EM algorithm assuming no labelled data, then compared the clusters to the actual labels. This algorithm creates clusters of the data, according to a specified probability distribution. In each iteration, the parameters of each cluster are calculated, and the probability of each data point being in each cluster is calculated. We used a Gaussian distribution with the following update rule.

$$w_j^{(i)} = \frac{P(z^i = j)P(z^i = j)}{\sum_{k=1}^{K} Pz^i = kP(z^i = k)} \quad (1)$$

$$\vartheta := argmax_{\vartheta} \sum_{i=1}^{m} \sum_{j=1}^{K} w_j^{(i)} log \frac{P(x^i, z^i; \vartheta)}{w_j^{(i)}} \quad (2)$$

We then used the semi-supervised EM algorithm with the labels of a randomly selected 20 percent of the examples. This algorithm incorporates the known labels into the calculation of parameters as above.

For each supervised learning algorithm, we split the data into training and validation examples using a 75/25 split. An additional test set was not needed. We tested the accuracy against both the training and validation labels. LR and GDA both fit a decision boundary to the data. LR uses Newtons Method to maximise the logarithmic likelihood on the training set, with the following algorithm.

$$H_{a,b} = 1/m \sum_{i=1}^{m} x_a^{(i)} x_b^{(i)} \sigma(\vartheta^T x^{(i)})(1 - \sigma(\vartheta^T x^{(i)})) \quad (3)$$

$$\nabla_a l(\vartheta) = 1/m \sum_{i=1}^{m} x_a^{(i)} y^{(i)} - x_a^{(i)} \sigma(\vartheta^T x^{(i)}) \quad (4)$$

$$\vartheta := \vartheta - H^{-1} \nabla l(\vartheta) \quad (5)$$

GDA fits a probability distribution to positive and negative examples, and calculates the decision boundary that maximizes the logarithmic likelihood on the training set, using the following equations.

$$P(x^{(i)}; \vartheta) = \frac{1}{1 + exp(-\vartheta^T x^{(i)})} \quad (6)$$

$$\vartheta := argmax_{\vartheta} log \prod_{i=1}^{m} P(x^{(i)}, y^{(i)}; \vartheta) =$$

$$argmax_{\vartheta} log \prod_{i=1}^{m} P(x^{(i)} | y^{(i)}; \mu_j, \Sigma) P(y^{(i)}; \psi) \quad (7)$$

We then used SVM, which creates a decision boundary based on the data points closest to the decision boundaries, creating support vectors. We maximize the Lagrangian on the training set with respect to values of alpha as follows.

$$\sum_{i=1}^{m} \alpha_i - 1/2 \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)}, x^{(j)} > \quad (8)$$

$$\alpha_i >= 0, i = 1, ..., m \quad (9)$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \quad (10)$$

We used three different kernels (linear, radial basis function (RBF) and polynomial), with notably different results.

DT creates a series of decision boundaries on the training set. Each boundary splits the data into two clusters (within the current cluster) at a value of a feature that minimizes the Gini loss.

$$\frac{|R_1| L(R_1) + |R_2| L(R_2)}{|R_1| + |R_2|}$$

$$L(R_m) = \sum_{k=1}^{K} P_{mk}(1 - p_{mk}) \quad (12)$$

Our final approach in this hit predicting problem was to use a Neural Network. We used a neural network regularization, with one hidden layer of six units and the sigmoid activation function. The L_2 regularization function was applied to the cost function to avoid over-fitting.

$$J(W) = \sum_{i=1}^{N} (y - \hat{y}) + ||\alpha_1 W_1 + \alpha_2 W_2||_2 \quad (13)$$

Where W_1 is the weight matrix mapping the features to the hidden layer and W_2 is the weight matrix mapping the output of the hidden layer to the final output.

4. RESULTS

We used accuracy, precision and recall on the training and validation sets to evaluate the performance of each algorithm (Figure 2). Note that plots in this section show only two features: Danceability and Acousticness.

The EM algorithm gave a poor accuracy of 50.1%, with predictions on data points matching poorly to their actual labels (Figure 1). The semi-supervised EM algorithm also gave a poor accuracy of 46.9%. We concluded that unsupervised learning algorithms are inappropriate for this supervised learning problem.

LR and GDA yielded a reasonable accuracy of 75.9% and 73.7% against the validation data, with similar accuracy against the training data indicating no overfitting. The

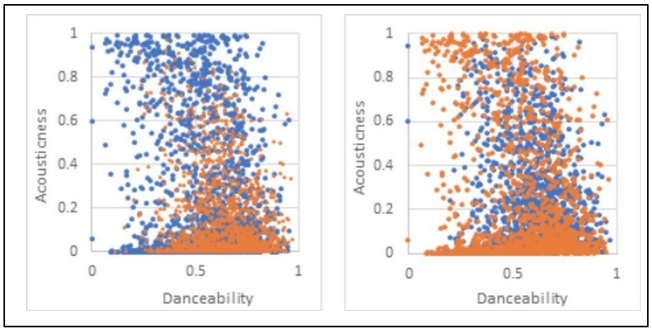

Figure 1. Original data and EM predictions. The accuracy of the predictions is poor.

Algorithm		Accuracy		Precision		Recall	
		Training	Validation	Training	Validation	Training	Validation
LR		0.735	0.759	0.719	0.758	0.575	0.648
GDA		0.735	0.737	0.767	0.767	0.503	0.559
SVM	Linear	0.550	0.729	0.783	0.806	0.436	0.490
	RBF	0.573	0.622	0.749	0.779	0.185	0.173
	Polynomial	0.543	0.738	0.770	0.788	0.499	0.536
DT		1.0	0.687	1.0	0.658	1.0	0.657
NN		0.768	0.765	0.875	0.752	0.600	0.661

Figure 2. Analysis Results

average cross-entropy loss was 1.372. The precision and recall on the validation set were acceptable. The confusion matrix on the validation set shows that there are some false negatives, meaning that songs that could potentially become hits could be unnoticed (Figure 3). Using random forests did not significantly improve the precision or recall. We could potentially increase the precision by collating a larger validation set with more positive examples.

For the SVM, each kernel yielded reasonable accuracy on the training data but poor accuracy on the validation data, indicating significant overfitting.

The DT algorithm can achieve full accuracy on the training data, by creating closely spaced decision boundaries that split the data perfectly. However, this is likely to cause high overfitting, with an accuracy of only 51.5% on the validation set. We used random forests to correct the SVM (linear and polynomial kernels) and DT against overfitting. Four sets of parameters were considered and the accuracy was recorded (Figure 10).

Using 10 trials of 500 random samples was the most successful measure for each algorithm. The accuracy on the training and validation sets were roughly equal, implying that overfitting was reduced significantly. Furthermore, to prevent overfitting of the DT, we experimented with different maximum depths. Using a maximum depth of n (the number of features) gave the optimal result (Figure 6).

The NN gives similar accuracy to LR, but interestingly generates significantly higher precision. This shows the

		Predicted	
		Positive	Negative
Actual	Positive	225	122
	Negative	72	386

Figure 3. LR Confusion Matrix on the Validation Set.

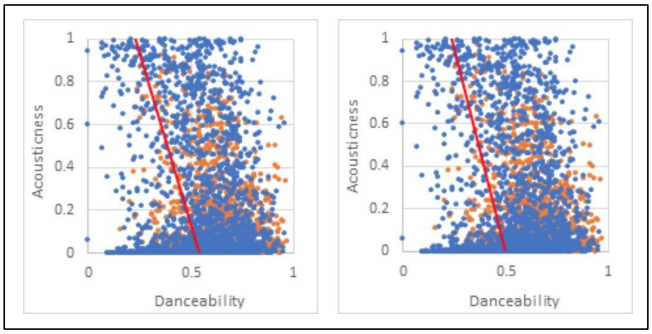

Figure 4. Decision boundaries for LR and GDA Algorithms. Boundaries from the two algorithms were very similar.

Algorithm	Random Forest Size	Trials	Accuracy	
			Training	Validation
SVM Linear	200	10	0.720	0.729
	200	50	0.527	0.544
	500	10	0.720	0.728
	500	50	0.674	0.686
SVM Polynomial	200	10	0.486	0.481
	200	50	0.353	0.349
	500	10	0.582	0.576
	500	50	0.536	0.540
DT	200	10	0.165	0.148
	200	50	0.024	0.027
	500	10	0.210	0.178
	500	50	0.058	0.046

Figure 5. Results from Bagging Using Random Forests.

Max Depth	Accuracy		Precision		Recall	
	Training	Validation	Training	Validation	Training	Validation
n	0.840	0.723	0.808	0.677	0.797	0.695
1.5n	0.925	0.709	0.895	0.654	0.921	0.692
2n	0.990	0.689	0.988	0.654	0.987	0.646
10n	1.0	0.698	1.0	0.654	1.0	0.666
100n	1.0	0.703	1.0	0.649	1.0	0.654
500n	1.0	0.702	1.0	0.653	1.0	0.657
1000n	1.0	0.698	1.0	0.646	1.0	0.631
5000n	1.0	0.704	1.0	0.663	1.0	0.657

Figure 6. Analysis Results for Different Maximum Depth of DT.

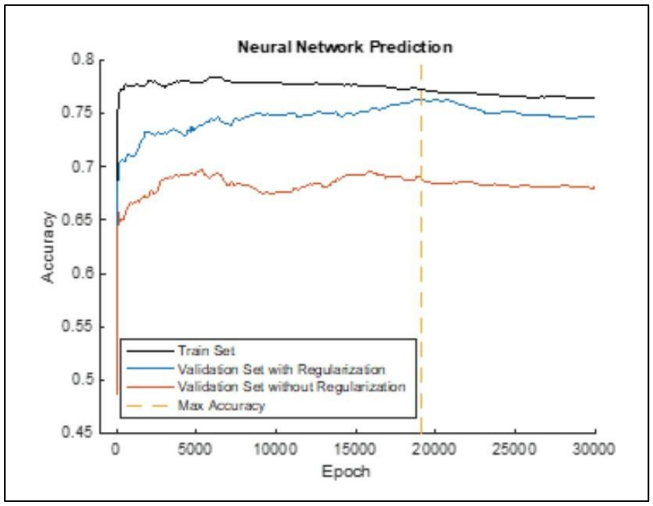

Figure 7. Accuracy of NN with increasing epoch. The peak accuracy on the validation set with regularization was achieved with approximately 19000 epochs.

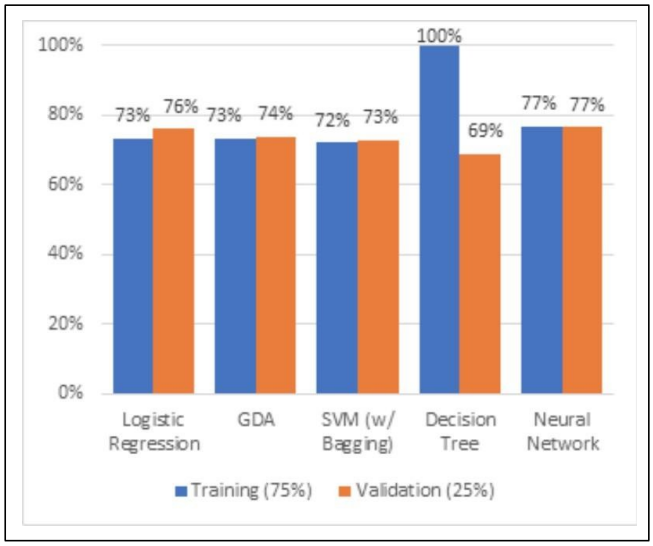

Figure 8. Analysis results. The LR and NN algorithms were the most successful.

robustness of the NN prediction. The regularization prevented overfitting. The optimal number of epochs was investigated in order to achieve the highest accuracy (Figure 7).

5. DISCUSSION

We used LR and NN, the most successful algorithms, for further investigation. We performed error analysis for both algorithms to determine the features with the greatest influence on predictions. Ablative analysis was used, beginning with one feature and subsequently adding the features which provide the greatest improvement in accuracy, until the maximum accuracy has been achieved (features which reduced the accuracy of the prediction were subsequently removed from the analysis). This provides a ranking of the features in terms of their influence on predictions. The artist score proved to be the major feature for

Importance	Logistic Regression		Neural Network	
	Feature	Accuracy	Feature	Accuracy
1	Artist Score	72.9%	Danceability	65.3%
2	Instrumentalness	73.2%	Acousticness	69.6%
3	Danceability	73.2%	Speechiness	73.0%
4	Acousticness	75.3%	Valence	73.4%
5	Speechiness	75.8%	Energy	74.9%
6	Loudness	75.8%	Artist Score	74.6%
7	**Tempo**	**75.9%**	Instrumentalness	75.1%
8	Valence	75.7%	**Tempo**	**76.5%**
9	Energy	74.0%	Liveness	76.4%
10	Liveness	74.3%	Loudness	72.7%

Figure 9. Error analysis for the two strongest- performing algorithms.

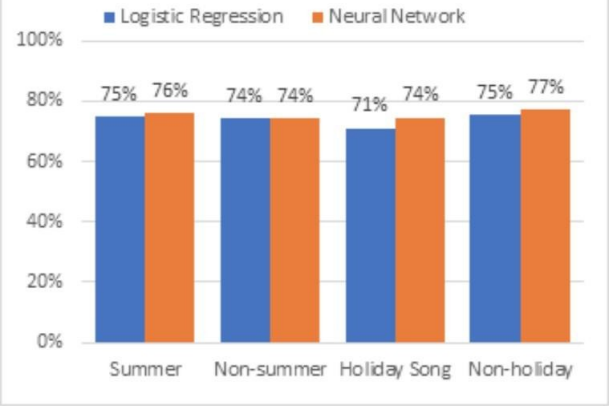

Figure 10. Features of hit songs released in winter vary from features of other songs.

LR and danceability was found to be the prominent feature for NN (Figure 9). The features at the end of the list decreased the accuracy of predictions.

Next, we investigated seasonal effects of the algorithms, focusing on two periods: summer months (June to August) and the holiday period (November to January). By training on songs released outside of the focus period and validating on songs released in the period, we were able to identify whether the general trends in pop music in the period were different. There was no difference observed for songs released in summer, but there was a noticeable reduction in the accuracy when the algorithms were validated on the holiday set. We can conclude that the features of a hit song are different in the holiday period (Figure 10).

We also investigated whether trends in pop music change over time. We divided the data into subsets of five-year periods and split each subset into training and validation sets using an 80/20 split. In most cases, the accuracy on both the training and validation set improved, implying that the features of pop music are somewhat unique to the time period of the songs release. The period from 2000 to 2004 saw a worsening of both the training and validation accuracy compared to that computed over all examples, and the period from 1995 to 1999 saw a decrease in the training accuracy (Figure 11).

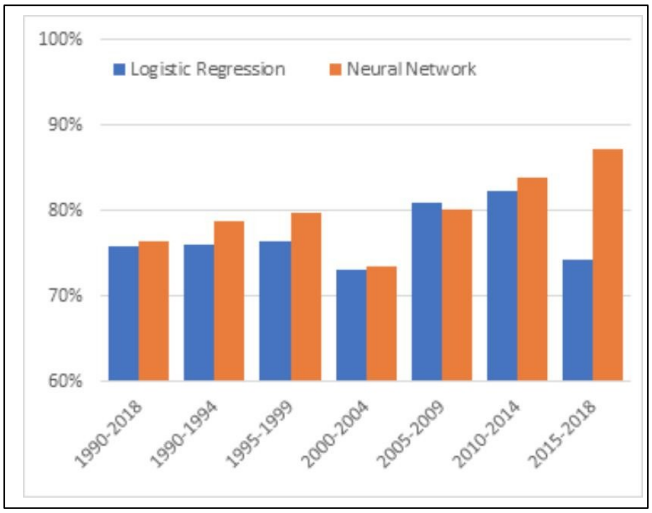

Figure 11. Accuracy on the validation set for specific time periods. Accuracy improves for individual time periods, indicating that hit songs have features unique to their time period.

6. CONCLUSION AND FUTURE WORK

The analysis showed that LR and NN yielded the highest accuracy, precision and recall of the algorithms tested. SVM and DT suffered from overfitting. We would like to use more data to reduce the variability of results. Instead of using 4,000 songs, we hope to include all Billboard Hot 100 hits taken from a longer time period, and a similar number of non-hits from the MSD. Furthermore, we would like to look into additional audio features, such as duration, which was not included in this project but has the potential to predict a songs Billboard success.

7. REFERENCES

[1] Billboard. (2018). Billboard Hot 100 Chart. Retrieved from: https://www.billboard.com/charts/hot-100

[2] Chinoy, S. and Ma, J. (2018). Why Songs of the Summer Sound the Same. Nytimes.com. Retrieved from: https://www.nytimes.com/interactive/2018/08/09/opinion/do-songs- of-the-summer-sound-the-same.html

[3] Guo, A. Python API for Billboard Data. Github.com. Retrieved from: https://pypi.org/project/billboard.py/

[4] Dorien Herremans, David Martens Kenneth Srensen (2014) Dance Hit Song Prediction, Journal of New Music Research, 43:3, 291-302.

[5] Mauch, M., MacCallum, R. M., Levy, M., and Leroi, A. M. (2015). The Evolution of Popular Music: USA 19602010. R. Soc. open sci.

[6] Sander Dieleman and Benjamin Schrauwen, Multi-scale approaches to music audio feature learning, in Proc. Int. Soc. Music Information Retrieval Conf., 2013, pp. 116121.

[7] Singhi, Abhishek, and Daniel G. Brown. ”Hit song detection using lyric features alone.” Proceedings of International Society for Music Information Retrieval (2014).

[8] Spotify Web API. Retrieved from: https://developer.spotify.com/

[9] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

[10] Yu, Lang-Chi, et al. ”Hit Song Prediction for Pop Music by Siamese CNN with Ranking Loss.” arXiv preprint arXiv:1710.10814 (2017).

8. CONTRIBUTIONS

Marcella collected song information from the Million Song Database and developed the neural network. Elena collected data from the Billboard API and coordinated the creation of the poster and final report. Nicholas collected audio features from the Spotify API and developed the unsupervised and supervised learning algorithms.