Predicting A Song's Commercial Success Based on Lyrics and Other Metrics

Angela Xue (angelax) and Nick Dupoux (njdupoux)

1 Task Definition

Given quantifiable metrics from a song, such as lyrics, genre, and tempo, we wanted to predict its commercial success. We used a pre-defined and pre-existing rating system (specifically, an aggregate attribute in the data called "hotttness") as a proxy for determining popularity and success.

Our system takes as an input a subset of songs drawn from the Million Song Dataset and their associated metadata, including genre, year, artist, rhythm/tempo, and instrumentation. The desired output for each of these songs is the value of the corresponding aggregate attribute in the dataset called "hotttnesss". This attribute is assigned to songs by the API providers based on a number of metrics, including mentions in the news, play counts, music reviews, radio airtime, and Billboard rankings. We will measure the accuracy of our prediction by looking at the percentage difference between our prediction of "hotttnesss" and the ground-truth value of "hotttnesss".

For our baseline, we trained a linear classifier on a feature set of just the song's artist. For our oracle, we simply used the ground-truth value of the input, i.e. the "hotttnesss" score.

Due to issues with our datasets (see the Datasets section), we didn't have time to fully implement all the approaches we had hoped to explore, but in this paper we will detail the approaches we attempted and discuss the rest in the Future Work section.

2 Datasets

2.1 Echo Nest API

For our project, we started out using data drawn from the Echo Nest API. EchoNest is an organization that offers an array of music data for use in applications and research. We utilized its API to gather information about songs from various genres. We collected a random sample of 100 songs from each of the 29 genres we selected by hand. Then, we filtered out songs that encountered API failures, which ultimately left us with 2184 songs in our Echo Nest dataset. For each song in this dataset, we collected the attributes of tempo, danceability, energy, loudness, key, duration, time signature, speechiness, liveliness, and hotttnesss for each song. We also scraped the associated lyrics for 632 of the songs in this initial dataset to use for later lyric analysis.

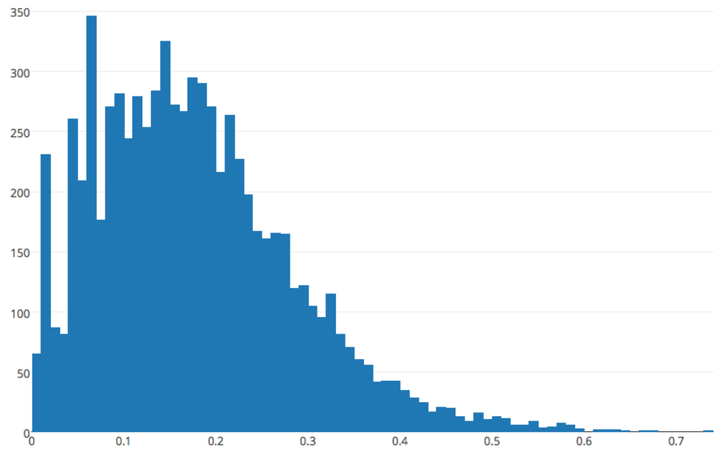
After running linear regression with L1 penalty, standard linear regression, and K-nearest neighbors regression on our initial dataset and getting results that were counterintuitive and made very little sense, we realized the problem was due to the fact that that this dataset had a large number of songs (680, or about 31.1%) with a value of 0 for "hotttnesss", our evaluation metric. We initially thought that a "hotttnesss" score of 0 indicated that a song was not popular at all, so we included those songs because we thought they would provide data on attributes of unsuccessful songs. We were mistaken. In the Echo Nest API, a value of 0 for "hotttnesss" is meant to indicate that the song has no "hotttnesss" score calculated for it at all.

This fact rendered the Echo Nest dataset effectively useless to us because it meant that we had a very strong incorrect data labeling that would overshadow the effects of any other features in the analysis. Discarding all the songs with a "hotttnesss" value of 0 would have reduced the size of our dataset down to 1504 songs, which we thought would be too small.

2.2 Million Song Dataset

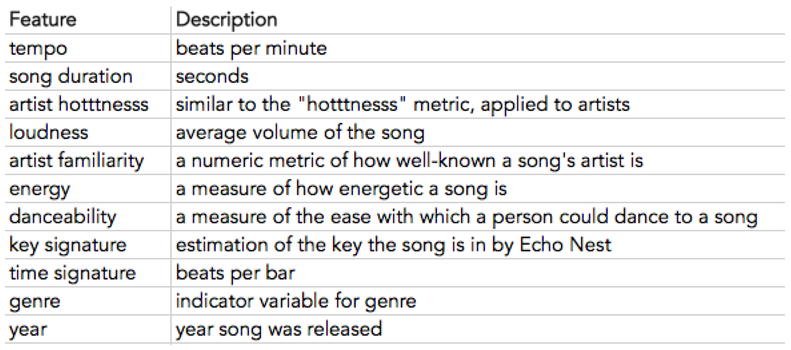
Our second attempt at finding useable data led us to the Million Song Dataset. The Million Song Dataset is a freely available collection of metadata and audio features for a million contemporary popular songs. Among the many raw metadata features included in the Million Song Dataset, we interested ourselves only in 12: genre, duration, artist familiarity, artist hotttnesss, year, tempo, danceability, energy, loudness, key, time signature, and "hotttnesss". The dataset also includes lyrical data for a large portion of the songs in it.

We curated our data based on whether or not associated lyrical data was available for each song. This narrowed down the number of useable songs to roughly 500,000. From there, we further filtered out songs that were missing values for any of the 13 the metadata attributes we were interested in, including songs with a "hotttnesss" value of 0. This brought the size of our dataset down to 9393 songs. The distribution and frequency of "hotttnesss" scores in our dataset can be found below.



3 Approach

Our planned approach to the task of predicting the ”hotttnesss” of a given song had two main parts: feature selection and prediction based on features. Our approach was iterative - we started with a list of features (described in the table below) that we thought would intuitively make sense. We intended to derive new features from existing features if we found that the raw metadata features were insufficient. The focus of this project is not the implementation of L_1 penalized regression, linear regression, or K-nearest neighbors regression, as this these are algorithms that have been perfected, but rather in the construction of meaningful features.



Feature	Description
tempo	beats per minute
song duration	seconds
artist hotttnesss	similar to the "hotttnesss" metric, applied to artists
loudness	average volume of the song
artist familiarity	a numeric metric of how well-known a song's artist is
energy	a measure of how energetic a song is
danceability	a measure of the ease with which a person could dance to a song
key signature	estimation of the key the song is in by Echo Nest
time signature	beats per bar
genre	indicator variable for genre
year	year song was released

Then, we applied a linear regression with L1 penalty to get an idea of which features would be more or less predictive so we could trim down the feature vector. After narrowing down the feature vector, we ran three different regressions (linear, L1 penalized, and k-nearest neighbors) to attempt to predict a song’s ”hotttnesss”.

4 Model and Algorithms

4.1 Model

We are given a dataset of X_i’s, where each X_i represents a song and its associated metadata features. Call the space in which our X_i’s live Ω. Our goal is find a function, $f : \Omega \to \mathbb{R}$, which predicts the ”hotttnesss” score of a song.

To make the problem tractable, we model f as a linear regression function, i.e.

$$f = w^T \phi(x)$$

To specify f, we need to specify w and ϕ. w will be determined by the algorithm we run, while we will have to specify ϕ, the feature map, as part of the model. Note that $\phi : \Omega \to \mathbb{R}^d$ is required for f to be well-defined.

4.2 The Trivial Algorithm: Choosing the Mean

For each song, the predicted ”hotttnesss” value is the mean ”hotttnesss” value in the training set. We consider this trivial algorithm as a much-needed sanity check.

4.3 Linear Regression

Our first and simplest (real) algorithm (and the one we also use for our baseline) for estimating w is ordinary least squares linear regression. Given a weight vector w, a feature map ϕ and a dataset (X_i, Y_i), define the loss of w to be

$$L(w) = \sum_{i=1}^{n} (w^T \phi(X_i) - Y_i)^2$$

Finding the optimal w can be thought of as finding the best linear predictor of Y in the least squares sense at the training points. A huge upside of ordinary least squares is that an exact solution exists, and is given by

$$\hat{w} = \arg\max_{w} L(w) = (X^T X)^{-1} X^T Y$$

4.4 L1 Penalized Linear Regression

We now turn to the algorithm we will use to find the weights w. Ordinary least squares is a good candidate, but for this application, a major goal is to determine which features are useful predictors. For example, we expect our measure of lyrical meaning and song genre to be more important than the song's duration. To this end, we instead use least squares regression with L_1 penalty. This algorithm has the feature that it outputs sparse vectors for w, where the zeros in w correspond to uninformative features. We will use a precoded L_1 penalized regression from the `sklearn` package, but for the sake of completeness, we discuss the basic workings of the algorithm.

Given a weight vector w, a feature map ϕ and a dataset (X_i, Y_i), define the loss of w to be

$$L_\lambda(w) = \sum_{i=1}^{n} (w^T \phi(X_i) - Y_i)^2 + \lambda \|w\|_1$$

where we tune λ to promote sparsity in the optimal w – larger λ corresponds to sparser outputs. For the output of our regression, we seek

$$\hat{w} = \arg\max_{w} L_\lambda(w)$$

As of now we tune λ by hand.

4.5 K-Nearest Neighbors Regression

As an alternative scheme for prediction, we employ k-nearest neighbor regression. The idea is similar to k-nearest neighbor classification. We are given a training dataset of labeled examples (X_i, Y_i), where in our case each X_i is a raw music file and Y_i its corresponding "hotttnesss". Using the same feature map as before, $\phi : \Omega \to \mathbb{R}^d$, we transform the dataset into $(\phi(X_i), Y_i)$.

Suppose we are now given a new song S whose "hotttnesss" we wish to predict. We first transform S with our feature map ϕ, and then find the k closest songs in our training set, where we use Euclidean distance as follows.

$$d(X_i, X_j) = \|\phi(X_i) - \phi(X_j)\|_2$$

Just as in k-nearest neighbor classification, we use this list of k nearest neighbors to predict song S's hotness. For the sake of concreteness, let L be the list of the indices of the nearest neighbors of S. Our first scheme assigns uniform weight to each neighbor to do prediction. More precisely,

$$f(S) = \frac{1}{K} \sum_{i=1}^{K} Y_{L_i}$$

In essence, we take an unweighted average. For an alternative scheme, we weight each neighbor by the inverse of its distance from X, as follows

$$f(S) = \sum_{i=1}^{K} p_i Y_{L_i} \qquad \text{where} \qquad p_i = \frac{d(S, X_{L_i})^{-1}}{\sum_{j=1}^{K} d(S, X_{L_j})^{-1}}$$

This has the effect of placing more weight on nearer neighbors.

k-nearest neighbors directly captures the intuitive notion that similar songs should have similar success, where we measure similarity in terms of our feature map ϕ. If we have a sufficiently dense training set, since k-NN is a local method, we expect a very smooth extrapolation from the training set. One potential problem with k-NN is we may have a feature with very low prediction capability but high range which dominates the neighbor selection. This can be avoided by first selecting features with L_1 regression, and mitigated by properly normalizing the feature vector so no one feature dominates the selection. As before with L_1 regression, the crux of the work for implementing a successful k-NN algorithm is in constructing our feature map ϕ, with a good set of features, which result in a dense training set, k-NN regression enjoys the good properties we mentioned above, particularly low bias since k-NN is a local method.

5 Results

5.1 Feature Selection

We ran an L1 penalized linear regression on the training dataset with $\alpha = 0.5$ the following feature vector:

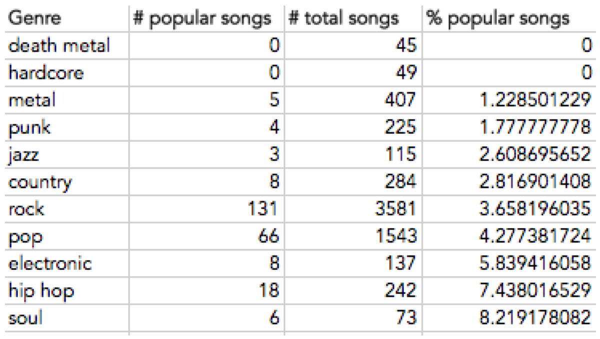
{duration, artist_familiarity, artist_hotttnesss, year, tempo, danceability, energy, loudness, key, time_signature, genre}

and got the resulting weight vector:

{4.17429950e-05, 0.00000000e+00, 0.00000000e+00, -0.00000000e+00, -0.00000000e+00, 0.00000000e+00, -0.00000000e+00, -0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00}.

This suggests that the only feature that seems to be even somewhat predictive is duration, which we found to be somewhat unsettling. The strange result prompted us to investigate further.

We had intuitively expected genre to be a reasonable predictor, so we decided to look more deeply into why we had gotten a weight of 0 for it. For each of the genres with more than 40 songs (40 chosen to ensure our results are statistically meaningful), we calculated the percentage of popular songs in our entire dataset. For the purposes of this analysis, we consider a song popular if it has a ”hotttnesss” score of 0.4 or above. If genre is not useful in predicting the ”hotttnesss” of songs, we would expect the percentage of popular songs to be independent of genre. However, as the table of selected genres below shows, this is not the case.

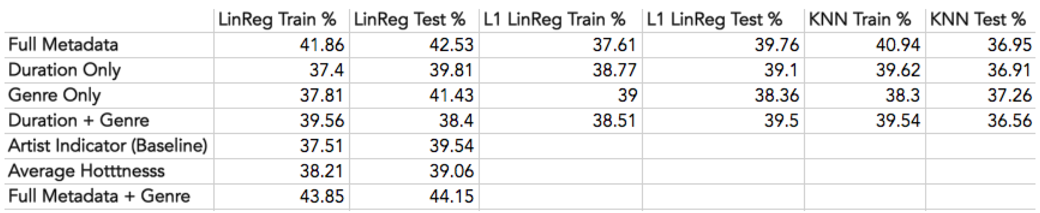


Genre	# popular songs	# total songs	% popular songs
death metal	0	45	0
hardcore	0	49	0
metal	5	407	1.228501229
punk	4	225	1.777777778
jazz	3	115	2.608695652
country	8	284	2.816901408
rock	131	3581	3.658196035
pop	66	1543	4.277381724
electronic	8	137	5.839416058
hip hop	18	242	7.438016529
soul	6	73	8.219178082

Some genres, despite having over 40 songs did not feature a single ?popular? song. Let us compare country and hip hop. Both have around 300 songs in the database, yet hip hop has 7.4% popular songs while country has only 2.8%. Metal has only 1.2% with over 400 songs. We believe our data set to be unbiased (starting with 500,000 songs for which we found lyrics), so this implies that genre is in fact a useful feature in predicting ”hotttnesss”, but our algorithms have not been able to utilize this feature.

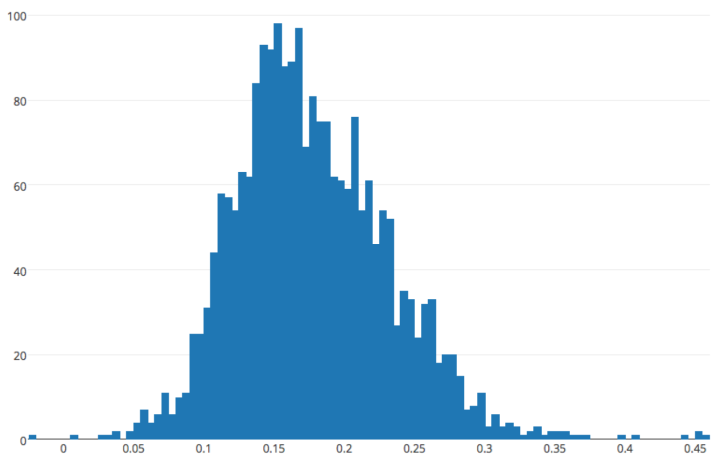
5.2 Prediction

Below, we have included a table of the accuracy rates of each of our algorithms on each of the feature vectors we considered.



	LinReg Train %	LinReg Test %	L1 LinReg Train %	L1 LinReg Test %	KNN Train %	KNN Test %
Full Metadata	41.86	42.53	37.61	39.76	40.94	36.95
Duration Only	37.4	39.81	38.77	39.1	39.62	36.91
Genre Only	37.81	41.43	39	38.36	38.3	37.26
Duration + Genre	39.56	38.4	38.51	39.5	39.54	36.56
Artist Indicator (Baseline)	37.51	39.54				
Average Hotttnesss	38.21	39.06				
Full Metadata + Genre	43.85	44.15				

The histogram below shows the distribution of the predicted ”hotttnesss” scores from running linear regression on our Metadata + Genre feature vector.



6 Discussion and Conclusions

The most notable result is that none of the feature vector + model combinations performed notably better than the trivial algorithm (defined in Model and Algorithms, above). Therefore, we cannot say with any great certainty that our feature vectors or models make any particular progress towards solving the problem of predicting "hotttnesss".

However, the results of the best-performing combination of features and model (Full Metadata + Genre) suggest that some of our features may be weakly indicative of "hotttnesss". From the histogram in the results section, we see that Full Metadata + Genre is heavily biased against labeling a song popular. This combination makes very few high "hotness" predictions, especially compared to the actual distribution of "hotttnesss" scores in the data (see the histogram in section 3.2). We believe the implication of this is that none of the features or attributes that make a song exceptionally popular are in our data. For all other algorithms, we actually did not predict any songs to be popular (i.e. have a "hottttnesss" score above 0.4). This could be due to a number of factors, including the limitations inherent in the linearity of our model and the poor predictive power of our features.

Ultimately, we believe that the features we considered are simply not very good at predicting the "hotttnesss" of a song.

7 Future Work

More work could be done in to adding to and refining our features. Most significantly, we would like to to incorporate the themes and meanings of songs into our model. We would do this by classifying songs with Nave Bayes, and then using PCA to narrow down the different theme labels to only the top several most significant.

In fact, to derive meaningful features from the lyrical data, we would certainly have to do some natural language processing. Our preliminary effort would be to train a naive Bayes classifier to classify songs by subject matter. Then we could use this classifier on test examples to guess subject matter. As a more advanced step, we would use topic modeling. This would allow us to organically pick song topics, and judge song success based on them. We would use this more complicated metric to replace the indicator variables in the feature map referring to subject matter of the song.

Another approach to deriving features from the lyrics of the songs we thought of is to perform sentiment analysis on the lyrics themselves, similar to how analysis is done on tweets from Twitter. The sentiment analysis would analyze the mood of the songs (e.g. happy, neutral, sad, or angry) and could possibly be paired with tempo / time signature / key data to get an even better estimate. Both the sentiment and topic of songs are similar conceptually to pre-defined genres for the song, so they would likely prove indicative of success just as genre is.

8 References

[1] EchoNest. *EchoNest API.* Oct. 2014 URL: http://developer.echonest.com/docs/v4

[2] 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.