Predicting Song Popularity

James Pham
jqpham@stanford.edu
Department of Computer Science
Stanford University

Edric Kyauk
ekyauk@stanford.edu
Department of Computer Science
Stanford University

Edwin Park
edpark@stanford.edu
Department of Computer Science
Stanford University

***Abstract*—Predicting song popularity is particularly important in keeping businesses competitive within a growing music industry. But what exactly makes a song popular? Starting with the Million Song Dataset, a collection of audio features and metadata for approximately one million songs, we evaluated different classification and regression algorithms on their ability to predict popularity and determined the types of features that hold the most predictive power.**

I. Introduction

Music has been an integral part of our culture all throughout human history. In 2012 alone, the U.S. music industry generated $15 billion. Of this $15 billion, the majority of the revenue is generated by popular, mainstream songs. Having a fundamental understanding of what makes a song popular has major implications to businesses that thrive on popular music, namely radio stations, record labels, and digital and physical music marketplaces.

The ability to make accurate predictions of song popularity also has implications for customized music suggestions. Predicting popular songs can be applied to the problem of predicting preferred songs for a given population.

Making predictions of song popularity based on machine learning, often referred to as Hit Song Science, is a problem that has gained a lot of traction within the past 10 years. Many private companies in the music industry are working on this problem, but details about the success of these companies are held private for competitive reasons.

Our project focuses on predicting whether or not a song is popular based on a number of characteristics of the song and its artist. The input for our algorithm is a list of song characteristics consisting of both acoustic features and metadata. We then use a number of machine learning algorithms (SVMs, neural networks, logistic regression, Gaussian discriminant analysis, and linear regression) to output whether or not the song is popular (or in the case of linear regression the score of the song popularity).

II. Related Work

The problem of predicting popularity is one that has been heavily researched. Salganik, Dodds, and Watts conducted an experimental study on popularity that focused heavily on the social influence of popularity. They found that the quality of a song only partially influences whether or not a song becomes popular, and that social influence plays an extremely large role [7]. Therefore, our project aims to use both acoustic features and metadata features to create a more accurate prediction model.

The work by Koenignstein, Shavitt, and Zilberman, which predicts billboard success based on peer-to-peer networks, potentially captures this social influence on song popularity. This group was extremely thorough with their work and used multiple regression and classification algorithms for their predictions [4].

Bertin-Mahieux et al. found that machine learning techniques can be used to apply labels to songs based on acoustic features. They created a model for predicting social tags from acoustic features on a large music database by using AdaBoost and FilterBoost [1]. While this group was extremely thorough with considering the possible models to use and sanitizing their features, using SVMs instead of AdaBoost with FilterBoost may have been a better option.

However, Pachet and Roy investigated the problem of making predictions of song popularity and made the blunt claim that the popularity of a song cannot be learnt by using state-of-the-art machine learning [6]. In order to test the effectiveness of current machine learning algorithms, they test the improvement of their classification models to a generic random classifier. Similarly to our work, Pachet and Roy consider both acoustic features and metadata; however, the study deals with an extremely large number of features (over 600) but does not mention any type of feature selection algorithm. As a result it is extremely likely that their model was subjected to overfitting. Pacet and Roy also considered features commonly used for music analysis which potentially could have affected the success of their results.

However, Ni et al have responded to the above definitive claim with more optimistic results on music popularity prediction, using a Shifting Perceptron algorithm to classify the top 5 hits from the top 30-40 hits (a slightly different problem from the aforementioned study) [5]. However, this study also uses more novel audio features which is a likely factor in their improved results.

III. Dataset and Features

A. Data

We used music data from The Million Song Dataset [8]. The Million Song Dataset is an exhaustive collection of audio features and metadata for one million songs dating to 2011. The audio features include attributes about the music track itself, such as duration, key, year. The metadata uses

more abstract features, such as danceability, energy, or song hotttnesss, generated from The Echo Nest, a music intelligence platform. Our project uses a subset of this data. We extracted 10,000 tracks from the database and of those 10,000 tracks, we removed all tracks that were missing any of features we were considering. This left us with 2,717 tracks. We divided these tracks so that 90% of the tracks was used for training and 10% was used for testing. Below is a subset of the fields for a song in the datset.

Feature	Type	Description
key	int	key the song is in
loudness	float	overall loudness in dB
mode	int	major or minor
mode confidence	float	confidence measure
release	string	album name

B. Feature Extraction

1) Baseline Features: The Million Song Dataset contains a plethora of feature types, ranging from number-type features such as those that measure the general loudness of a song, string-type features such as names of artists and albums, and array-type features such as those that contain pitches across the length of the song. All array-type acoustic features contain measures for segments across the song. We included the mean and variance throughout all of the segments as features.

2) Additional Features: By looking at plots of different features vs. song popularity, we saw that there were some non-linear relationships. By squaring the values, we were able to capture polynomial relationships between some of our features and the popularity. Additionally, to avoid constraining our model under an additive assumption, we also incorporated some interaction terms (e.g. tempo $\times$ major/minor) to capture multiplicative effects.

3) Bag of Words: Our dataset includes many string features, including song name, artist id, and terms that the artist is frequently associated with (genre). In order to capture these in our model, we used a bag of words approach and added the top 900 frequently occurring words as term frequency features in our model.

C. Popularity

The Echo Nest provides social field for a song called song hotttnesss which we will use as our metric of popularity. While the exact calculation of this field is not released, the metric is generally based upon total activity they see for the songs on the thousands of websites that Echo Nest uses. Songs with a hotttnesss value above a threshold were classified as popular; in our case, we defined songs as popular if they were in the top 25% of song hotttnesss. We set this threshold to be 0.623, since this was the 75th percentile for our dataset.

IV. Methods

A. Feature Selection

In total, our final dataset consists of 977 features, many of which were not relevant to predicting popularity. As a result, using all features suffered from overfitting. In order narrow down the number of features and find the most relevant features, we conducted several feature selection algorithms.

1) Forward Stepwise Selection: Forward selection greedily chooses the best combination of features by starting with an empty subset of features, then incrementally adding a feature to the model that was selected through evaluation of the feature subset through cross-validation. This step is repeated until the generalization error is minimized and the best subset of features is reported.

2) Backward Stepwise Selection: Backward stepwise selection works similarly to forward stepwise selection; however, instead of starting with an empty subset of features, it begins by evaluating the use of all features and incrementally removes features until the model is optimized.

3) l_1 Regularization: l_1 Regularization is a shrinkage method that regularizes the coefficient estimates by shrinking the coefficients towards zero. Regularization often improves the fit because reducing coefficient estimates can significantly reduce their variance, thus decreasing the effect of overfitting.

$$\min_{\theta} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \sum_{j=1}^{n} |\theta_j|$$

By increasing the tuning parameter λ, the shrinkage penalty term effectively serves to force some coefficient estimates to become *exactly equal* to zero. l_1 regularization has an advantage over l_2 regularization in that the models obtained by l_1 regularization are easily interpretable: setting some coefficient estimates to zero is essentially performing feature selection. Tuning λ has a bias-variance trade-off: increasing λ decreases the flexibility of the fit but also increases bias.

B. Classification

1) Logistic Regression: We used logistic regression with l_1 regularization. Logistic regression selects the parameter θ that maximizes the likelihood function:

$$L(\theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) = \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1-h_\theta(x^{(i)})^{1-y^{(i)}}$$

where

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

2) Linear Discriminant Analysis (LDA): We used LDA, which is a specific method of Gaussian Discriminant Analysis, to build our classifier. LDA assumes that samples have come from a multivariate Gaussian distribution with a specific mean vector and a covariance matrix that applies to all classes. LDA is similar to logistic regression in that they both produce linear boundaries and thus similar results, but LDA tends to perform better when the Gaussian assumptions are met.

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\Big(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\Big)$$

3) Quadratic Discriminant Analysis (QDA): Similar to LDA, QDA is another specific method of Gaussian Discriminant Analysis that assumes that samples come from a multivariate Gaussian distribution with a specific mean vector. However, unlike LDA, it assumes that each class has its own covariance matrix. By estimating multiple covariance matrices, QDA allows for more flexible fit by allowing non-linear boundaries.

$$p(x;\mu,\Sigma_k) = \frac{1}{(2\pi)^{n/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma_k^{-1}(x-\mu)\right)$$

4) Support Vector Machines (SVM): We also leveraged support vector machines to classify our data. To create an optimal margin classifier, a decision boundary or separating hyperplane can be calculated that maximizes the distance of the nearest points of any class to the decision boundary. For SVMs to perform efficiently in high-dimensional spaces, SVMs leverage the kernel trick, which allows for computation without having to explicitly represent the high-dimensional feature vectors.

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha^{(i)} - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)}, x^{(j)}\rangle$$

$$s.t.\ 0 \le \alpha_i \le C,\ i = 1, ..., m$$

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

We used a Gaussian radial basis function (rbf) kernel to create a non-linear classifier by transforming the feature space.

$$K(x,z) = \exp\left(\frac{||x-z||^2}{2\sigma^2}\right)$$

5) Multilayer Perceptron: We also used a multilayer perceptron classification algorithm, an example of a neural network model. The MLP consists of a directed graph, where possibly different dimensional layers of nodes are all fully connected. The first layer of nodes embody the original set of features, and the final layer of nodes represents higher level features influenced by multiple original features. The weights of these nodes are learned through backpropagation. To minimize the error of the output, we use stochastic gradient descent to find the change of each nodes weight and the activation function $y(v_i) = tanh(v_i)$ to map the weighted inputs to the output of each neuron.

$$E(n) = \frac{1}{2}\sum_{j} e_j^2(n)$$

$$\Delta w_{ji}(n) = -\eta\frac{\delta E(n)}{\delta v_J(n)} y_i(n)$$

C. Regression

Seeing our classification results, we noticed that classification approaches lose valuable information about the value of the song popularity itself due to the binary conversion. As a result, we also use regression to predict the values of popularity. For regression, we fitted models using a standard multiple linear regression, and applied feature selection methods to achieve the best coefficient estimates for regression. For feature selection, we evaluated each model of size 1 ... m features using cross-validated mean squared errors.

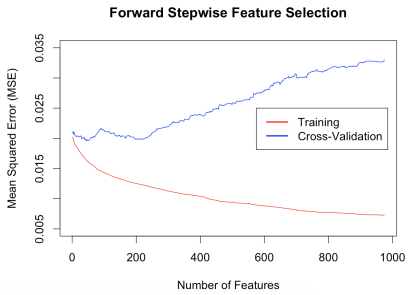

Fig. 1. Forward Stepwise Feature Selection

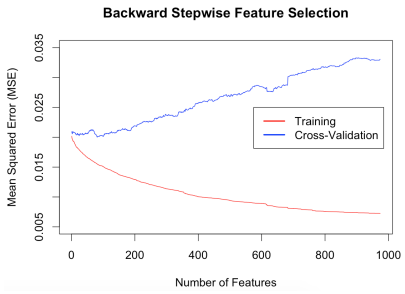

Fig. 2. Backward Stepwise Feature Selection

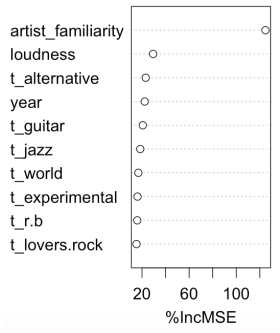

Fig. 3. Most Important Variables

V. Experimental Results

A. Feature Selection

For both forward stepwise selection and backward stepwise selection methods, we determined the optimal number of features in our fitted model by seeing which set of features gives the smallest cross-validation error. From a model with 976 features, forward stepwise selection chose a model of 45 features (see Fig. 1), whereas backward stepwise selection chose a model of 81 features (see Fig. 2). In both feature selection methods, significant features included artist_familiarity, loudness, year, and tag words such as ”alternative”, ”guitar”, and ”jazz” (see Fig. 3).

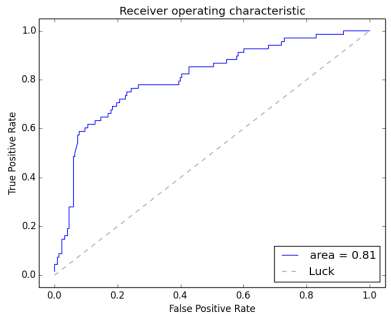

Fig. 4. ROC Curve for SVM with Gaussian Kernel

Model	AUC
SVM (Linear Kernel)	0.79
SVM (RBF Kernel)	0.81
Logistic Regression (LR)	0.69
LDA	0.71
QDA	0.64
Multilayer Perceptron (MLP)	0.79

Fig. 5. AUC for Classification Models

Model	P	R	F1	Train	Test
SVM (Linear)	0.500	0.706	0.585	0.830	0.762
SVM (RBF)	0.495	0.750	0.597	0.827	0.759
Logistic Reg	0.567	0.500	0.531	0.861	0.790
LDA	0.610	0.529	0.567	0.858	0.807
QDA	0.359	0.618	0.454	0.756	0.647
MLP	0.588	0.441	0.504	0.889	0.793

Fig. 6. Classification Results: {train, test} accuracy

B. Classification

1) Metrics: Since we set our threshold such that 75% of the songs in our dataset are classified as not popular and 25% are classified as popular, 75% accuracy can be achieved by predicting all 0s. Therefore, we concluded that accuracy alone would not be a good measure of how well our model can classify. In order to capture this, we also considered the precision, recall, F1, and AUC scores of our models.

Precision, recall, and F1 score are used to capture how well our model does in the task of classification. Precision measures the portion of examples that were classified as popular that are truly popular while recall measures the portion of examples that are truly popular that our model classified as popular. F1 score acts as the weighted average between these two values.

The area under the receiver operator characteristic curve is a metric used to evaluate the performance of a binary classifier by taking the area under a curve created by plotting TPR vs. FPR at different probability thresholds. The AUC represents the probability that the classifier ranks a random positive example higher than a random negative one.

2) Tuning Parameters: With respect to the kernels used by our SVM models, we optimized two parameters: C and γ. C determines the trade-offs between misclassification and simplicity of the decision surface. γ determines the weight of a single training example. We used 10-fold CV to tune the C and γ parameters. For the linear kernel, C and γ were chosen to be 1 and $\frac{1}{n}$ respectively, and for the RBF kernel, C and γ were chosen to be 44 and .0001 respectively.

Additionally, we used 10-fold CV to tune two parameters for the MLP; we chose $\tanh(x)$ to be the activation function for the hidden layers of nodes, and we chose stochastic gradient descent to be the algorithm for node weight optimization.

3) Results: See Fig. 4, 5, and 6.

4) Discussion: The algorithm with the highest F1 score was the SVM using a Gaussian kernel. Considering the fact that the SVM with a linear kernel had a lower F1 score than the Gaussian kernel, our data is likely non-linear. As a result, SVMs using a Gaussian kernel would naturally perform better than the other models that assume linearity. Additionally, higher F1 scores were achieved using the rbf kernel over the linear kernel suggesting nonlinear relationships between the audio/metadata features and popularity.

However, there were no significant differences in the performance among all our our tested models; F1 scores all ranged between 0.5 and 0.6 and the accuracies all ranged between 0.75 and 0.80. This could be due to the fact that all of our classification algorithms assume the data is linearly separable, and the data was mostly likely not linearly separable. The differences in the performance of these models can be attributed to the tradeoffs made in the algorithms. For example, SVMs are primarily influenced by the data points closest to the margin, while logistic regression and other models are influenced by all data points. SVMs also perform better in problems with a high number of dimensions.

We chose our hotttnesss threshold to consider the top 25% of songs to be popular (songs with a hotttnesss larger than 0.623). We initially chose a threshold that considered the top 50% of songs to be popular and were receiving fairly low metrics across the board, including precision, recall, F1 score, and accuracy. We hypothesize that this happened because popular songs (which we predict to have defining characteristics) in reality are only a small subset of all songs. Therefore, the top percentile of popular songs that had these defining characteristics were confounded by the more songs we labeled as popular. As a result we changed our threshold to classify the top 5% as popular. This increased the accuracy, but it was the result of our algorithms predicting that all songs were not popular, a fact that was reflected in recalls of 0. A likely cause of this result is a lack of popular training examples. By using the top 25% of songs as popular, we are able to have songs with defining characteristics in addition to enough popular examples to properly train our models.

C. Regression

1) Metrics: In the regression setting, we use mean squared error (MSE) to evaluate how well our model predicts popularity. The average error is obtained by taking the square root of the MSE, which approximates the standard error of

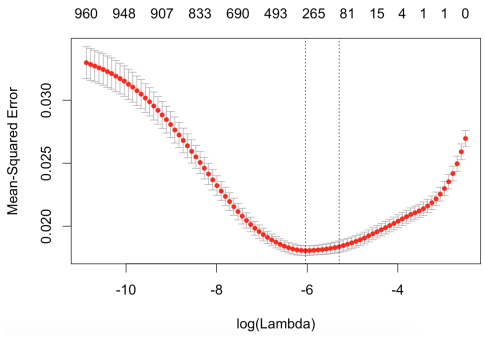

Fig. 7. Cross-validation for optimal λ for Lasso

Model	MSE	Avg Error
Baseline	0.02529	0.1590
Full Model ($n = 976$)	0.03010	0.1735
Selected Model ($n = 45$)	0.01842	0.1357
Lasso ($\lambda = 0.00238$)	0.01802	0.1342

Fig. 8. Regression Results

our predictions. Therefore, the smaller the MSE, the higher confidence we have about our predictions.

2) Tuning Parameters: The λ parameter for l_1-regularization was tuned using 10-fold cross-validation. In addition, the number of features to use for determining forward and backward stepwise feature selection was obtained using 10-fold cross-validation.

3) Results: See Fig. 8.

4) Discussion: According to the results in Fig. 8, seeing that the test error increases from the baseline model (58 features) to the full model (976 features), we can conclude that adding a lot of additional Bag of Words features leads to overfitting to noise in our training set. However, by significantly reducing the number of features to 45 using forward feature selection, we were able to significantly decrease the variance of our fit and consequently reduce overfitting.

Overall, the smallest test error among our models is achieved by using Lasso regression. 10-fold cross-validation is used to choose the optimal value of λ, which turns out to be 0.00238, over a grid of $\log \lambda$ values, (see Fig. 7). Lasso regression gave an output of 21 different features with non-zero coefficient estimates. The features chosen by the Lasso are very similar to the ones produced by forward and backward feature selection.

Our feature vector is generally very sparse due to the large number of BoW binary features, so stepwise selection's greedy approach may not work as well as Lasso's shrinkage method.

VI. Conclusion

Through several different feature selection algorithms, we were able to identify the most influential features in our dataset by taking the intersection among the feature selection algorithms, namely artist familiarity, loudness, year, and a number of genre tags. All of the features can be seen in Fig. 9. We found that the acoustic features arent nearly as predicative

Fig. 9. Features names scaled by the log of coefficients for linear regression

as the metadata features. A likely reason for this is that there is a lot of variation in acoustic features within a single song that make it difficult to extract metrics that represent an entire song. Metadata such as genre tags or year of release are much better at accurately reflecting a trait of a song.

Currently, the features we used divided the range of pitch and timbre into buckets. However, because the original data consists of time series data points with respect to these values, we could add features that represent the transition between certain pitch values. An n-gram type model using sequences of pitches/loudness as features would allow us to investigate if particular pitch intervals have any influence on song popularity.

In addition to using the song hotttnesss metric, we can also create our own metric of popularity, which we can define as the number of downloads on iTunes or the number of plays on Spotify. This would allow us to more accurately capture what we define to be popular and allow us to generalize our findings to commonly understood metrics. Furthermore, we can potentially extend the application of this project to a song recommendation system: training a model on examples with the most number of plays labeled as popular could lead to personal playlist or song recommendations.

References

[1] Bertin-Mahieux, Thierry, et al. "Autotagger: A model for predicting social tags from acoustic features on large music databases." Journal of New Music Research 37.2 (2008): 115-135.

[2] Hastie, Trevor, and Hui Zou. "Regularization and variable selection via the elastic net". *J.R. Statist. Soc.* B, 2005.

[3] James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer Texts, 2014.

[4] Koenigstein, Noam, Yuval Shavitt, and Noa Zilberman. "Predicting billboard success using data-mining in p2p networks." Multimedia, 2009. ISM'09. 11th IEEE International Symposium on. IEEE, 2009.

[5] Ni, Yizhao, et al. "Hit song science once again a science." 4th International Workshop on Machine Learning and Music, Spain. 2011.

[6] Pachet, Franois, and Pierre Roy. "Hit Song Science Is Not Yet a Science." ISMIR. 2008.

[7] Salganik, Matthew J., Peter Sheridan Dodds, and Duncan J. Watts. "Experimental study of inequality and unpredictability in an artificial cultural market." science 311.5762 (2006): 854-856.

[8] 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.