Predicting the Popularity of Songs - CS221

Maika Isogawa (misogawa@stanford.edu)
Sam Masling (smasling@stanford.edu)
Monica Pan (jpan5@stanford.edu)
Stanford University

1 Overview

The music industry has become one of the central hubs for fame, fortune, and artistic commentary. The success of a musical artist and their work depends heavily on how popular their songs are, or how widely their songs are heard. With the development of modern streaming services like Spotify, YouTube, and more, the ability of songs to go viral has changed. But what makes a song popular? Does the success of a song depend on the lyrics? The artist? Or perhaps the musicality of the song plays a larger role than both? This study aims to use various machine learning methods to discover what components of a song contribute its popularity and success.

The success of a song is heavily dependent on an individual's subjective tastes. To streamline what is categorized as a 'successful' song, we use the Billboard Hot 100 [1]. This ranking is the music industry standard record chart in the United States. The rankings are determined by sales, radio play, and online streaming.

The code for the entire project can be found in our GitHub repository [2]

2 Task Definition

The goal of our project is to classify whether or not a given song will make it onto the Billboard Hot 100 chart. Essentially, we aim to predict the popularity of a song. Using historical data from decades of past songs, we aim to develop models that utilize both text and audio features of a song to generate a prediction. We designed a simple binary prediction: '1' if the song makes it to the Billboard Hot 100, and '0' if it does not.

The evaluation metric for the success of our project will be classification accuracy, an F1 score, and a confusion matrix. We have implemented three main methods: a binary linear classifier, logistic regression, and SVM.

3 Infrastructure

3.1 Data Sets

To properly implement our algorithms, we cleaned, compiled, and restructured two large data sets.

The first data set was compiled through the usage of the Billboard API[3]. The Billboard API was used to access the title and artist of every song that has made the weekly Top 100 since 2000 through present day. We ended up with a list of over 7000 unique songs that had made the weekly Billboard Hot 100 list. The songs in this data set were exclusively songs that made it to the Billboard Hot 100.

Below are two example data points from this data set:

- Artist: Ed Sheeran, Song Title: Perfect.
- Artist: Fergie, Song Title: Fergalicious.

The second data set is from Kaggle[4], an online community that allows users to find and publish a variety of data sets. This data set is a collection of over 55,000 unique songs, the majority of which did not make it onto the Billboard Hot 100. This data set included the artist, song name, a link to the song, and the song lyrics.

Below are two example data points from this data set:

- artist: ABBA, song: As Good As New, link: /a/abba/as+good+as+new_20003033.html, text: I'll never know why I had to go Why I had to put up such a lousy rotten show Boy, I was tough, packing all my stuff Saying I don't need you anymore, I've had enough And now, look at me standin...
- artist: Aerosmith, song: My Fist Your Face, link: /a/aerosmith/my+fist+your+face_20004249.html, text: Wake up baby, what you in for Start the day upon your knees What you pissin' in the wind for You musta snorted too much bleas East house pinball wizard Full tilt bozo plague Second floor t...

3.2 Data Pre-processing

In order to format the data in a way that would be useful for our project, we put our data through heavy pre-processing. We combined the two data sets into one large data set. Our master data set includes a label of whether or not the song made it to the Billboard Hot 100, in addition to a variety of text and audio features. To create our master data set, we used a variety of techniques to gather data that each data set was lacking.

The first data set only included the artist and song title. To gather the rest of the data, we utilized the Genius API [6] to scrape the lyrics for the songs from the first data set. The second data included most of the text data that was necessary for our project. We removed the link to the song, and labeled whether or not each song made it to the Billboard Hot 100.

3.3 Data Cleaning

As we were compiling our master data set, many sections of the data were left empty or unusable. Much of these issues came from the Spotify and Genius APIs themselves. Thus, we had to be careful when reviewing our data set, and re-scraped data when necessary.

To balance our data, we took an equal amount of songs that made it onto the Hot 100, and songs that did not. In addition, to account for differences in era and genre, we attempted to mitigate superfluous imbalances by selecting an equal amount of songs from similar eras and genres. For the Genius API, and the Spotify API [5] which we discuss later, we implemented a multithreading tool that allowed us to scrape data far more efficiently, saving us hours of dead time.

3.4 Features

For our basic features, we used the lyrics of the song (text features). We extracted word features from the lyrics and created a feature vector.

For more advanced features, we utilized the Spotify API to gather metadata from each track. Our advanced features include: duration of song, key, acousticness, danceability, energy, instrumentalness, liveliness, loudness, speechiness, tempo, valence (musical features). These advanced features will give us insight into the audio qualities of the track, so that our classification will be more robust compared to the text data alone. A full table describing the audio features that are available through Spotify's API can be found in the appendix [9]. The descriptions are officially defined by Spotify.

3.5 Example Input

Below are two examples of our data. Each song has both text and audio data, and our models were trained on this data.

artist	title	text	Top100	duration	key	acousticness	danceability	energy	instrumentalness	liveliness	loudness	speechiness	tempo	valence
ABBA	Dancing Queen	You can dance,	yes	230693	9	0.382	0.539	0.884	0.00166	0.76	-6.53	0.0403	100.812	0.752
Drake	Come Thru	We had the type	no	236360	3	0.159	0.558	0.468	0	0.586	-5.812	0.173	81.977	0.405

4 Approach

In order to build our system, we want to understand the complex relationships between different aspects of a song and its popularity. Given that our data include many important features, such as lyrics and audio, it is hard to identify the real trends and patterns. We want the system to be able to make decisions based on them. A machine learning model serves the purpose really well. The weight of each feature would be improved as each song gets trained. After learning, the model would be able to predict the classification result based on the trained weights of the feature vector.

There are many pros and cons using machine learning models. Machine learning is really good at reviewing a large amount of data and looking for hidden patterns that human cannot easily see between thousands of word and audio features. It is also a highly adaptive method as we are continuously improving our data set over time. However, the machine learning results are very susceptible to error. With the automation, machine learning models cannot identify inaccurate data or implementation mistakes by themselves. Therefore, some diagnosis and correction process are necessary in building the system.

4.1 Baseline and Oracle

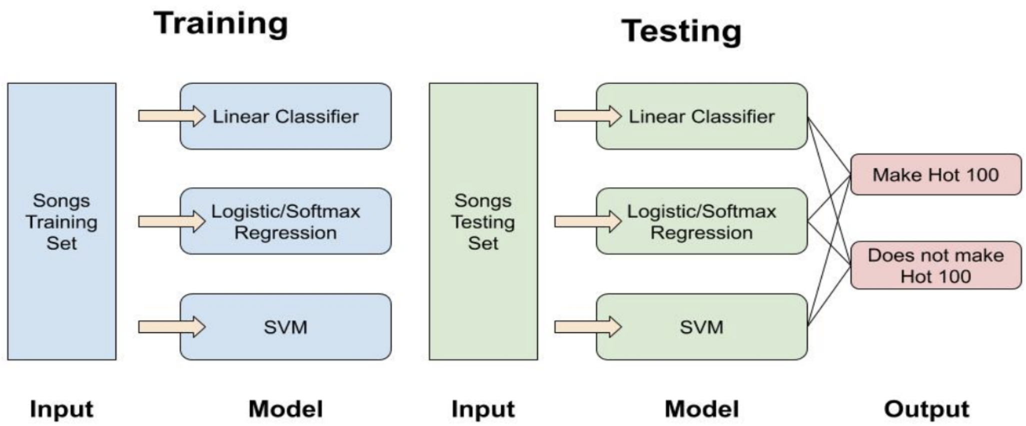
The baseline of the project is training a linear classifier only with the lyrics of songs. The weights of the feature vector was trained based on the frequency of each words. The baseline algorithm is expected to give some statistical truth about the relationships between the lyrics and the popularity of songs. As a result, the baseline algorithm works pretty well: the overall accuracy is about 80% while the training error is as low as 12%. Therefore, it can be seen that training on lyrics of songs is a promising approach.

The oracle is having a professional music producer listen to 50 songs randomly selected from our data set and determine if they will be popular. As a result, the music producer guessed 48 songs correctly out of 50 songs, giving a high accuracy of 96%. It is nearly impossible for us to implement an algorithm that analyzes rhythms or process lyrics like a human. However, we believe that it is important to measure the audio features of songs since the popularity of music itself heavily influences the success of songs nowadays.

4.2 Models

We explored several effective algorithms in three machine learning models. We utilized stochastic gradient descent in the binary linear classifier, regularized logistic regression, and support vector machine (SVM). A few implementation details are shown in the section below.

To separate our full data set: the training set consisted of 80% of the total data, and the testing set consisted of 20% of the total data. The training and testing sets are selected using the train_test_split method from Scikit-learn library. Each model uses the same feature extractor. The feature vector is presented as a sparse matrix efficiently. The feature extractor extracts each word from the lyrics of the songs and maps it to its frequency.



4.2.1 Binary Linear Classifier

We drew inspiration from the Sentiment Analysis assignment. We used a linear predictor for the binary classification. The model calculates the score of the given inputs, defined as the weighted combination of features and predicts positive if the score is equal to or larger than 0. In order to get the optimized weight for each feature, we used stochastic gradient descent to minimize the hinge loss.

4.2.2 Logistic Regression

We implemented a regularized logistic regression using the Scikit-learn LinearRegression class. In order to solve the binary problem, we used the liblinear solver. The method DictVectorizer is used to transform the feature vector to valid sparse matrix inputs.

4.2.3 SVM

Implemented using the Scikit-learn SVM class. We explored four kernel SVM: linear, polynomial, Gaussian, and Sigmoid. The fit method was used to train, and the predict method was run on the testing set to predict the classification.

4.3 Challenges

One challenge was building the master data set. Not only were there technical challenges, but there were conceptual challenges as well. On the technical side, we had problems with making multiple request to the Genius and Spotify servers. As we were running a large data set, we had to make thousands of requests. Not only did this take a long time, but we also had issues with disconnection, barring by the website, and other credential issues. Many of these issues were likely on the company's end, and we were able to work around them by utilizing multiple techniques such as multithreading, and using multiple account credentials. Conceptually, since the success of a song is not binary, it was difficult to classify a song as 'not successful' for the purpose of our project. A popular song that barely missed the Billboard Hot 100 would be classified as not-successful. With a binary classifier, we had no way of representing the range in successes, even within the songs that did make it to the Billboard hot 100.

Another challenge we encountered was that the models with text features only gave much better results than with both text and audio features. This seemed counter-intuitive because we expected that feeding the classifiers with more information would certainly improve the results. It was possibly caused by the noise in the data and the fact that modern pop songs put more emphasis on the music instead of lyrics. We continuously improved the data set by cleaning out some unnecessary data points. The final data set only contained songs from 2000 to 2019.

5 Literature Review

Prediction tasks are common in machine learning. Thus, there is already a lot of literature on predicting the popularity of songs, or predicting hit songs. These projects ask questions similar to ours: are there certain characteristics for hit songs? What has the largest influence on a song's success? Can old songs predict the popularity of new songs? Music is likely a common area of interest, as it adds another layer of complexity beyond text and natural language alone.

Here we briefly review two projects who's goals are very similar to ours.

The first project is "Song Popularity Predictor" by Mohamed Nasreldin, Stephen Ma, Eric Dailey, and Phuc Dang [7]. Like we did, this project utilized Spotify's API to gather metadata on tracks. They also defined the "success" of a song by whether or not it made it on to the Billboard Hot 100 chart. This project used models like logistic regression and SVM as we did, but they also looked at models like KNN, and a decision tree. The most accurate model predicted popular songs with a 68% accuracy.

The second is the article, "Predicting Hit Songs with Machine Learning," by Minna Reiman and Philippa Ornell [8]. This project theorized that hit songs had features in common that made them appealing to a majority of people. They also utilized Spotify's API. The other models used for this project were K-Nearest Neighbors and Gaussian Naive Bayes. Their most accurate model was their Gaussian Naive Bayes model, with 60.17% accuracy. This project explored the changes in certain music features over time. This is further analysis of the data and results that would be great to include in a more rigorous approach of this problem.

Compared to projects similar to ours, our main differential is our data set and our balance of text vs audio features. Our data set is well balanced, and it is also modernized to contain songs from a similar age. These are components that may have led to a higher accuracy rate with our models.

6 Error Analysis

The main metrics for our classification task are accuracy, an F1 score, and a confusion matrix. For each model, there are results for basic text features and results for both text and audio features. Our final results for each of the models are found below.

Linear Classifier Results

Hinge Loss

Text Features

	Negative (Predicted)	Positive (Predicted)
Negative (Actual)	567	424
Positive (Actual)	59	1466

Accuracy: 0.808029
F1 Score: 0.858565
Train Error: 0.124814

Text and Audio Features

	Negative (Predicted)	Positive (Predicted)
Negative (Actual)	877	144
Positive (Actual)	166	1359

Accuracy: 0.888712
F1 Score: 0.906604
Train Error: 0.073239

Logistic Regression Results

Text Features

	Negative (Predicted)	Positive (Predicted)
Negative (Actual)	850	141
Positive (Actual)	178	1347

Accuracy: 0.873211
F1 Score: 0.894125

Text and Audio Features

	Negative (Predicted)	Positive (Predicted)
Negative (Actual)	871	120
Positive (Actual)	165	1360

Accuracy: 0.886725
F1 Score: 0.905158

SVM RESULTS

Text Features

	Linear		Polynomial		Gaussian		Sigmoid	
	Negative (Predicted)	Positive (Predicted)	Negative (Predicted)	Positive (Predicted)	Negative (Predicted)	Positive (Predicted)	Negative (Predicted)	Positive (Predicted)
Negative (Actual)	808	183	0	991	885	106	885	106
Positive (Actual)	209	1316	0	1525	247	1278	247	1278

	Linear	Polynomial	Gaussian	Sigmoid
Accuracy	0.844197	0.606121	0.859698	0.859698
F1 Score	0.870370	0.754764	0.878652	0.878652

SVM RESULTS

Text and Audio Features

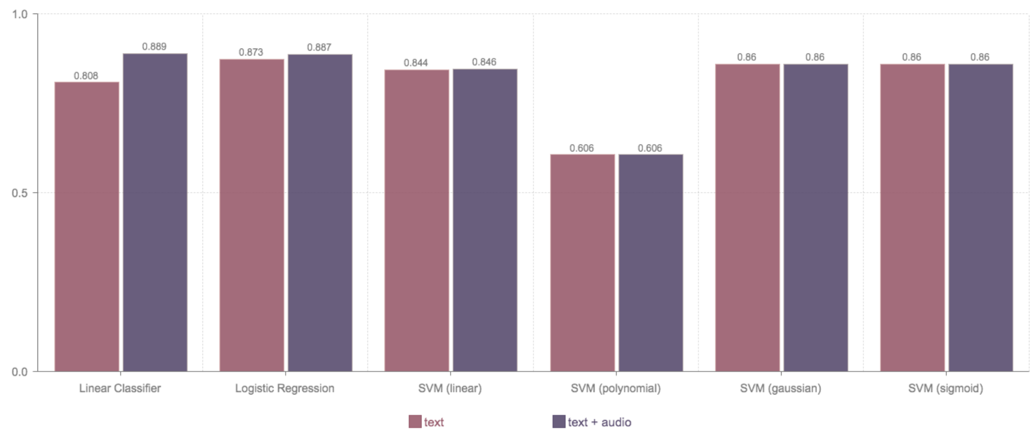
	Linear		Polynomial		Gaussian		Sigmoid	
	Negative (Predicted)	Positive (Predicted)	Negative (Predicted)	Positive (Predicted)	Negative (Predicted)	Positive (Predicted)	Negative (Predicted)	Positive (Predicted)
Negative (Actual)	812	179	0	991	838	153	838	153
Positive (Actual)	208	1317	0	1525	200	1325	200	1325

	Linear	Polynomial	Gaussian	Sigmoid
Accuracy	0.846184	0.606121	0.859698	0.859698
F1 Score	0.871897	0.754764	0.882451	0.882451

For text features alone, the binary linear classifier, logistic regression, and most of SVC models performed equally well. Among them, the logistic regression performed the best with an 87% accuracy and an F1 score of 0.89. The polynomial SVC model did not perform as well as the rest of models but still had an accuracy of 61%.

For the text and audio features, all classifiers produced slightly more accurate results, and performed better than with text features alone. The most significant improvement is in the results of the bi-linear classifier. The accuracy increased from 81% to 89% and the F1 score from 0.86 to 0.91. The increases in the results of other models are within 1%. One of the possible reasons why the results changed very little is that the feature vector mostly consisted of word features. Since we only have 13 audio features but thousands of unique words in the feature vector, the predictions were heavily based on lyrics itself.

Below is a comparison of the prediction accuracy of all of the models with text features and with text and audio features.



7 Conclusion and Next Steps

To predict the popularity of a song, we built and explored several machine learning models: bi-linear classifier, logistic regression and SVM. We explored different aspects of songs, lyrics and many audio features in order to understand which feature affects the popularity the most. Our final results showed that focusing on both lyrics and audio features generally gave us a better classifier than if we only had text features. The results were expected because the musical features of a song should impact the popularity of it. We also found that the bi-linear classifier is the most successful among all models we tested. The high accuracy and F1 score showed that it is a very promising prediction tool of the success of a song. It also yielded a clear statistical relationship between the performance of the model and different features.

As the literature review and the multitude of API tools exemplify, music and music success are popular topics for research. Music seems to be able to convey meaning and feeling, similar to words, but without the semantics. This phenomenon lends itself well to machine learning techniques, as models may reveal things about music that we may not know or yet understand. Our project could lend itself to be developed into a more sophisticated one, but nonetheless, the techniques we learned in this project will be useful to apply in various problems that we may face in the future.

8 Appendix

[1] https://www.billboard.com/charts/hot-100

[2]https://github.com/smasling/FinalCS221Project

[3]https://github.com/guoguo12/billboard-charts

[4]https://www.kaggle.com/mousehead/songlyrics

[5]https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

[6]https://docs.genius.com/

[7]https://towardsdatascience.com/song-popularity-predictor-1ef69735e380

[8]https://pdfs.semanticscholar.org/e6cc/edb50d2c2b01bca108cb090943e86fb58135.pdf

[9]

Feature	Type	Description
Danceability	float	Describes how suitable a track is for dancing. This value is based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is the least danceable and a value of 1.0 is the most danceable.
Energy	float	A value representing a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud and noisy. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy. Energy is described as a value between 0.0 and 1.0.
Key	int	The key the track is in. Integers map to pitches using standard Pitch Class notation. (`https://en.wikipedia.org/wiki/Pitch_class`)
Loudness	float	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). This value usually ranges between -60 and 0 dB.
Mode	int	Describes the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by the value 1 and minor is represented by the value 0.
Speechiness	float	Detects the presence of spoken words in a track. The more exclusively speech like the recording is, the closer the value is to 1.0. If the speechiness ranges between 0.66 and 1.0, the track is probably made entirely of spoken words (such as audio books, poetry etc.). Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered. Values below 0.33 most likely represent music and other non-speech-like tracks.
Acousticness	float	A confidence measure between 0.0 and 1.0 of how acoustic a track is.
Instrumentalness	float	Predicts whether a track contains no vocals. Sounds like "*Ooh*" and "*Aah*" are treated as instrumental in this context, while rap or spoken words are clearly "vocal". The attribute ranges between 0.0 and 1.0 and the closer to 1.0 the value is; the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks.
Liveness	float	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track was performed live.
Valence	float	Describes the musical positiveness conveyed by a track. The value ranges between 0.0 and 1.0. Tracks with high valence sound more positive (happy, cheerful etc.), while tracks with low valence sound more negative (sad, depressed etc.).
Tempo	float	The overall estimated tempo of a track in beats per minute (BPM). Tempo is the speed or pace of a given piece and derives directly from the average beat duration.
Duration	int	The duration of a track in milliseconds.
Time signature	int	An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).