

A Predictor for Movie Success

Jeffrey Ericson & Jesse Grodman
CS229, Stanford University

1 Introduction

What makes a movie successful? The right combination of talent, chemistry, and timing, all with some good luck. Consider “The Shawshank Redemption” and “Gigli”. The former a flop at the box office, overshadowed at the 1994 Academy Awards by “Forrest Gump”, and ignored for many years - until its rise from a cult hit to one of the most revered movies of all time. The latter was poised for success, with Ben Affleck and Jennifer Lopez co-starring at the height of their popularity, yet the film became a poster child for movie disasters. Though there are many factors that comprise a movie’s success, and it is not always clear how they interact, this paper attempts to determine these factors through the use of machine learning techniques.

We predict five different measures of success, based solely on what we know about a movie before its debut. Many attributes reveal themselves after a movie premieres, but our input features include only that which a producer can influence during planning and production. Our measures of movie success are diverse enough to cover a variety of perspectives, from popular review to critical review to gross profit.

2 Data

2.1 Data Collection

The first step was to collect data. There was no pre-existing dataset with the movies and features we wanted, so we collected data from multiple sources and merged it. Our main sources of data are IMDB, Rotten Tomatoes, and Wikipedia. The mechanism through which each of these sites makes their data available varies - everything from an API (thank you, Rotten Tomatoes) to messy space-delimited text files (we need to talk, IMDB).

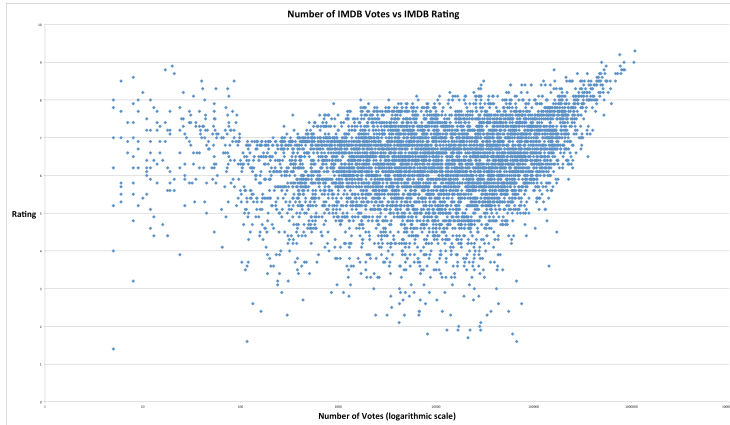
From IMDB, we obtained for each movie: movie title, IMDB rating, plot description, budget, box office gross, and opening weekend gross. From Rotten Tomatoes, we obtained critic score, audience score, runtime, MPAA rating, studio, theater release date, DVD release date, list of genres, abridged list of cast, and abridged list of directors. From Wikipedia, we obtained the number of Academy Awards that actors and directors in each movie had won prior to that movie, and also the number of Best Picture films that actors and directors in each movie had been involved in, also prior to that movie. We collected the same data from the Golden Globes. In order to stay true to our goal of only considering factors known before a movie’s release, we considered only awards that had been received prior.

Using Python and JavaScript to extract, parse, and clean up what we retrieved, we had our data - albeit unmerged. We removed rows from each table that did not have a complete tuple of information (in particular, Rotten Tomatoes often had important attributes missing about less popular movies, like runtime or studio). To merge the files, we keyed on a combination of two fields: normalized movie title and release year. To normalize movie titles, we uppercased all titles and stripped all non-alphanumeric characters. Finally, we loaded each table into a SQLite database and joined all three tables, leaving us with a data set of movie information with 6,590 movies. Our data made sense: the highest rated movies across IMDB and Rotten Tomatoes include “The Dark Knight” and “The Godfather” and the lowest rated movies include “Justin Bieber: Never Say Never” and “From Justin to Kelly”. Alas, maybe the best classifier predicts that if a movie title includes the word “Justin”, it is destined for failure.

Our data set represents many features as bit vectors. Some features have been parsed into other features, such as splitting the release date into bins by month and bins of five years. We also added a feature for whether a movie came out on a popular weekend (like the Fourth of July or Christmas), and included a feature for whether a movie was a sequel, third installment, or later in a series. Our final input feature vector contained 176 features: 6 pertaining to bit vectors for MPAA rating, 1 for movie runtime, 22 pertaining to bit vectors for movie studio, 12 for release month, 4 for popular weekends, 8 for year bins, 1 for budget, 6 for awards (one for each of the following for both Academy and Globe: sum of all award winners among the cast, director, and those involved in a Best Picture), 22 pertaining to bit vectors for genre, 90 pertaining to bit vectors for the most popular words in a movie’s plot description (after removing stop words and words that seemed unrelated to movie content), and 4 bit vectors for the movie’s place in a series.

2.2 Data Cleaning and Improvement

After some preliminary runs of our machine learning algorithms, we had decent results, but felt we could do better if we cleaned up the data. Each IMDB rating comes with both a numeric rating and the number of votes cast. Concerned that less well-known movies with fewer votes would be unreliably rated, we plotted the number of votes vs. the rating, and what we found was interesting.



Movies with more than 100 votes trend towards a higher rating with more votes. However, with fewer than 100 votes, there is little structure to the data. Based on this, we removed all movies with fewer than 100 votes from our dataset.

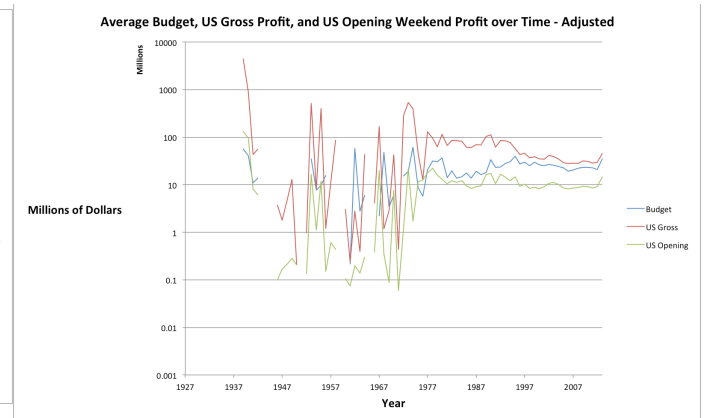
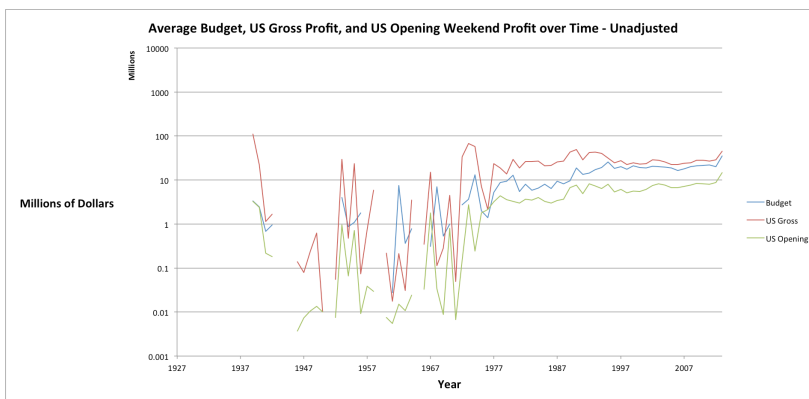
We noted that we had three monetary attributes: budget, US gross profit, and US opening weekend profit. In order to maximize the utility of these fields, we accounted for inflation and population growth. For the budget, we accounted only for inflation, since the population has no direct relation to it.

For US gross profit and US opening weekend profit, we accounted for both inflation and population growth, using the following calculations:

$\text{budget} := \text{budget} * (\text{USD in 2013} / 1 \text{ USD in year of movie})$

$\text{US gross profit} := \text{US gross profit} * (\text{USD in 2013} / 1 \text{ USD in year of movie}) * (\text{US population in 2013} / \text{US population in year of movie})$

$\text{US opening weekend profit} := \text{US opening weekend profit} * (\text{USD in 2013} / 1 \text{ USD in year of movie}) * (\text{US population in 2013} / \text{US population in year of movie})$



Interestingly, after accounting for these factors, all three fields are fairly constant over the last 40 years.

3 Machine Learning Techniques

3.1 Locally Weighted Linear Regression

We chose locally weighted linear regression as a first approach, since it is easy to implement and makes few assumptions. Additionally, it seemed natural that we should weight success of similar movies more heavily. We implemented the algorithm in Matlab, making sure to normalize our input vectors (very important since our input features are on widely different scales: 0-300 for runtime minutes, but millions for budget).

Intending to next run a classification algorithm and desiring to directly compare the performance of the two, we discretized the output space and ran it as a classification algorithm. We did this by measuring the median for each output value and predicting outputs as usual with locally weighted linear regression. We classified the prediction as a success if the predicted and actual values were either both above or both below the median.

We measured the accuracy rates on all five outputs for various values of Tau. We did this using 70/30 hold-out cross validation, training on 70% of our data, and testing on the remaining 30%.

| | T = 0.01 | T = 0.05 | T = 0.1 | T = 0.5 | T = 0.8 | T = 1.5 |
|----------------|----------|----------|---------|---------|---------|---------|
| Critic Score | 0.5793 | 0.5746 | 0.5740 | 0.5693 | 0.5693 | 0.5693 |
| Audience Score | 0.6107 | 0.6096 | 0.5992 | 0.5929 | 0.5929 | 0.5929 |
| US Gross | 0.6604 | 0.6227 | 0.6180 | 0.6128 | 0.6128 | 0.6128 |
| US Opening | 0.5097 | 0.5086 | 0.4971 | 0.4945 | 0.4940 | 0.4940 |
| IMDB Rating | 0.5877 | 0.6070 | 0.5824 | 0.5824 | 0.5824 | 0.5824 |

0.01 was found to be a reasonable value of Tau - below it, performance dropped rapidly. It is interesting to note that the performance decreases as Tau increases; this indicates that there is some merit to the use of locally weighted linear regression, as opposed to standard linear regression. If locally weighted linear regression provided no benefit over linear regression, there would be no decrease in performance, and perhaps even an increase in performance, as we increased Tau, making the algorithm more and more similar to linear regression.

3.2 SVM (Support Vector Machine)

The second machine learning technique we applied was SVM. With SVM, we could use all 176 of our input vectors and then pare down the space by use of filter feature selection, which uses the forward search paradigm to choose a subset of features with which to make predictions. As with locally weighted linear regression, we predicted whether a movie performed better or worse than the median found across our entire dataset for each output feature. We asked the following questions (based on the medians):

1. Does a movie have a Rotten Tomatoes critics' score above 60 (on a scale to 100; Rotten Tomatoes labels a movie either "fresh" with a 60+ score or "rotten" with a score below 60).
2. Does a movie have a Rotten Tomatoes audience score above 64 (the median value, also on a scale to 100).
3. Did a movie gross over \$7.49M in the United States (the median)?
4. Did a movie gross over \$500K in the United States its opening weekend (the median)?
5. Did a movie have an IMDB score of 6.5 or above (the median, on a scale to 10).

For each output feature, we performed a separate filter feature selection. We grouped bit vector features that were from the same category. For example, our dataset has 22 bit vectors to represent a movie's genre categorization (22 total genres). We grouped these 22 features together when performing filter feature selection.

We ran SVM with a linear kernel because it was the only one that both converged and had a classification rate substantially above 0.5. Also, we used L1 regularization, with the "slack penalty" C set to 0.15. This value was a good compromise that caused the algorithm to converge after a reasonable number of iterations without allowing for too many misclassifications. C is rather low, but this is not surprising given the huge variety in movies. It made

sense not to employ a heavy penalty, but instead to fit a decision boundary to the majority of movies and accept the misclassification of movies that do not fit the general trend.

Filter feature selection provided us a list of features ranked by score (which was the test success rate from cross-validation). With ranked features, we then performed hold-out cross-validation (again, 70/30% training/test) for each of the top k features (in practice, we tested k from 1 to 20). We chose k based on which k had the highest test prediction rate (or equivalently, the lowest test error rate). While this method does not provide optimum results, it provides a good heuristic. We arrived at 5 different subsets of input features for predicting each of the 5 output features. For each of the 5 subsets, the optimal value of k was somewhere between 8 and 15 (this was one of the motivations for extending our testing of k to 20).

The 5 subsets had many input features in common: each contained the features pertaining to genre, MPAA rating, runtime, and studio. However, budget was meaningful for critic score, box office gross, and opening weekend gross, but not audience score or IMDB rating (by meaningful, we mean that the smallest cross-validation test error was realized without this input feature). Number of awards won by cast members (both Academy Awards and Golden Globe Awards) was only a meaningful feature for audience score, box office gross, and opening weekend gross. Release month was a meaningful feature for box office gross and opening weekend gross, but not for any of the ratings. Of all the words used in movie plot descriptions, most were not meaningful. Four that were include “life” and “story” which were not meaningful for box office gross or opening weekend gross, but the former was meaningful for all the ratings and the latter was meaningful for all the ratings except the audience score. The other two words were “find” and “must”, which were meaningful only for both box office gross and opening weekend gross.

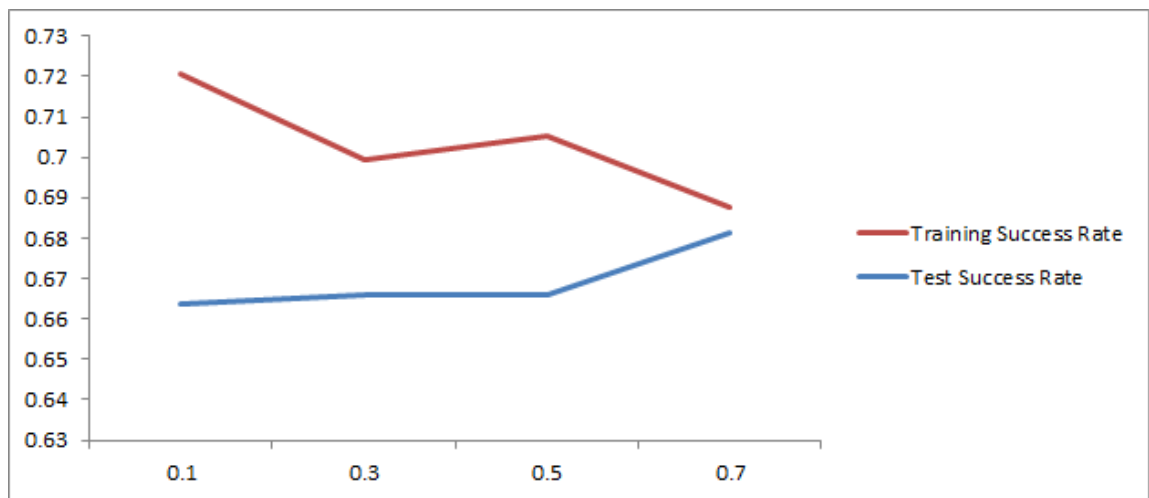
Below are the resulting prediction rates for cross-validation (whether a sample from the test dataset was correctly predicted as being above or below the median for the particular output feature) using the top input feature subsets for each of the 5 output features:

| | Success Rate for Critics Score | Success Rate for Audience Score | Success Rate for US Gross | Success Rate for Open Weekend Gross | Success Rate for IMDB Rating |
|------------------------------------|--------------------------------|---------------------------------|---------------------------|-------------------------------------|------------------------------|
| Feature Set for Critics Score | 0.68237 | 0.68237 | 0.86081 | 0.86761 | 0.71481 |
| Feature Set for Audience Score | 0.6719 | 0.69492 | 0.84406 | 0.85034 | 0.71533 |
| Feature Set for US Gross | 0.6719 | 0.68184 | 0.88488 | 0.87755 | 0.7112 |
| Feature Set for Open Weekend Gross | 0.66405 | 0.68707 | 0.88435 | 0.87703 | 0.7033 |
| Feature Set for IMDB Rating | 0.66039 | 0.6787 | 0.8158 | 0.81423 | 0.70853 |

We achieved roughly a 70% success rate for each of the rating outputs and 88% success rate for each of the monetary gross outputs. Using a group of features was an improvement over using any one feature category, even though some single feature categories were indeed highly correlated with outputs. From our filter feature selection scores, the most correlated feature category for US gross was studio (78%), and for US opening weekend, it was genre (79%). The most correlated variable for ratings was always genre: 65% for critics score, 62% for audience score, and 63% for IMDB rating. Thus, the gain from including multiple features was least realized in predicting critic score, which we would have been nearly as successful doing by just considering genre.

It is interesting to note that the feature set for each output feature did not necessarily achieve the highest success rate for that feature. For example, consider IMDB rating. The success rate was actually highest using the input feature set for audience score. This is not overly surprising, since the rates are close. The feature selection algorithm is greedy, and does not promise to find the best possible set of features. Also, we measured these rates using one pass of 70/30 cross validation, which is likely to introduce some noise.

To measure the balance between bias and variance, we plotted a learning curve where we trained on 10%, 30%, 50%, and 70% of our samples. The plot below uses the input features set found for critic score, and measures the classification success of critic score.



As we see the success rates approach one another at 70%, we can infer that we do not have a problem with high variance. It seems reasonable that we might achieve a success rate above 70%, so perhaps we have high bias, and a better model exists for the data. We noted that having more input features can help lower bias, so this was one of the points at which we both gathered more data and cleaned up our existing data, as we discussed in the data section above. In fact, our initial data collection contained about 1900 movie samples, while our final dataset has over 6,500, with many more features.

The classification rates we achieved with SVM are higher across the board than with locally weighted linear regression. The main reason for this is likely that we were able to use many more input features. As locally weighted linear regression is a regression algorithm, we could not use input features that do not have a natural numeric representation, like genre or MPAA rating.

4 Conclusion

Using SVM, we achieved classification rates well above 0.5 for each of our output features. While the success rates are perhaps not ready for use in financial analysis, our results provide insight into what makes a movie successful. For instance, it was revealing that movies with the words “life” or “story” in the description have higher ratings (we verified that these input features correlated with movie success, not failure) and the words “find” and “must” correlate highly with positive box office gross.

One self-imposed limitation in our approach is that we allowed ourselves only to consider as input features those which are known about a movie pre-release. For example, we could surely have achieved higher success rates if we considered the awards that a movie would go on to win, but this did not seem very insightful. Our dataset was also heavily skewed to movies post 1980, as many before that do not have accurate financial data.

While a near-perfect predictor some day would be a machine learning triumph, it could also lead to an overly algorithmic approach to movie planning. Studios may be less willing to take risks, and outliers are often some of the most exciting products of Hollywood.

References

CS229 Course Notes
www.imdb.com
www.rottentomatoes.com
www.wikipedia.org
www.usinflationcalculator.com