Predicting Movie Revenue from Pre-Release Data

Benjamin Flora, Thomas Lampo, and Lili Yang

December 12, 2015

1 Introduction

The film industry is predicted to generate 39.1 billion dollars in box office revenue for 2015, with continued growth to nearly 50 billion dollars by 2020[1]. Film production exhibits a great deal of financial risk in an age where blockbuster movies can have budgets in excess of of hundreds of millions of dollars. This risk could be reduced to an extent with the use of quantitative modeling. Since movie scripts and projects are pitched and budgets are set and spent well before any revenue is ever made, we plan to construct a predictive model using features that are generally known before the movie release date.

2 Prior Work

Several CS 229 groups previously generated predictive models for movie revenue using tens to hundreds of features. Prior models include linear and logistic regression, support vector machines, Naive Bayes, and K-means clustering (see Apte, Forsell, & Sidhwa 2011, Ericson & Grodman 2013, Gross, Merwe, & Eimon 2013, Yoo, Kanter, & Cummings 2011, Xue & Chen 2011, Im & Nguyen 2011). Classification models using "revenue bins" generally perform in the range of 50% assignment error or above when bins are log-spaced, while pure linear regression models (even when weighted) generally perform more poorly. In the scientific literature, similar studies have been performed to find features that correlate with revenue. Older studies identified features such as actor and director pay [2], while newer studies discovered that activity on social media such as Twitter and Wikipedia correlate well with revenue[3].

3 Data Acquisition

IMDB is the largest store of movie data on the web but they do not document their API so we decided to use OMDB.com which has movie data from IMDB and box office revenue from Rottentomatoes.com. The OMDB API requires a movie title in the search so we obtained a list of movies from IMDB, which publishes flat files that include all movie titles (http://www.imdb.com/interfaces).

Using this list and the OMDB API, we were able to curate data to develop our training, development, and test sets. Looking at movies released between 2000 and 2015 we were able to pull 3177 movies that also had box office revenue numbers (see Fig. 1 for revenue histogram). Once we pulled the relevant data to create the features discussed below we split the data randomly into 3 sets - S_{train}, S_{dev}, and S_{test}, where S_{dev} and S_{test} each contain approximately 10% of the data and S_{train} the remainder. Specifically, S_{train} contains 2544 movies, S_{dev} contains 323 movies, and S_{test} contains 310 movies.

4 Features

Since our goal is to build a model that can predict the profitability of a movie during development, we are only interested in data and features that would be available or could be projected before release. Specifically, our models are built around the following sets of features.

Genre, MPAA Ratings, Movie Length. It stands to reason that different types of movies have different degrees of commercial success. After all, who doesn't like a nice romantic comedy? We model the movie genre as a set of binary features, one for each genre.

[1] http://www.pwc.com/gx/en/global-entertainment-media-outlook/assets/2015/filmed-entertainment-key-insights-2-global-box-office.pdf

[2] Simonoff & Sparrow. *Chance*. 13:15-24 (2000)

[3] Mestyan, Yasseri, & Kertesz. *PLoS One*. 8:e71226 (2013)

A particular movie can be classified into one or more genres as well as length (in minutes) and its MPAA (Motion Picture Association of America) rating. We look at rating both as a multinomial feature that scales from 1 for "G" to 5 for "NC-17" and as a set of binomial features, one for each of "G", "PG", "PG-13", and "R". (we omit the binary feature for "NC-17" because there are not enough movies rated as such to make the distinction viable.)

Release Date. Important information can also be derived from the release date of the movie. We split the release date of a movie into the year and the month. Intuitively, the year feature could allow our model to account for long term trends such as inflation or trends in movie going rates. Conversely, incorporating months as a set of binary features could give us insight into seasonality. For instance, we'd expect better performance from movies released around Christmas or in the first month of summer.

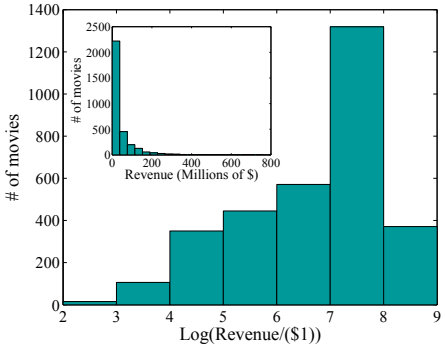

Figure 1. Label data (movie revenue) exhibits a narrow distribution that is better spread and scaled logarithmically. Main plot is a histogram on the Log (base 10) of revenue, while the inset histogram uses just revenue.

Actors, Directors, and Writers. Films with big name actors, directors, and/or writers will often heavily publicize this fact, so it seems intuitive that the star quality of the participants could provide an indication of the success of the movie. However, star quality is itself a nebulous quality that can be difficult to model, so in our baseline model we use as a rough proxy the average number of movies made by the actors, directors, and writers. It seems reasonable that the more famous an actor is, the more movies he or she will have made.

Locality. Where a movie is made can also be important to the success of movie. It is a well known trend that the most profitable movies tend to be US movies out of Hollywood. We attempt to use two different features to model the locality of a movie. The first is whether or not the movie was made in the US, and the second is the number of languages that it has been translated into.

Natural Language Processing Features. Each movie comes with a brief, textual description of the plot of the movie, similar to what a prospective theatergoer might see. We use natural language processing (NLP) to extract two types of features from this text, structural and tonal. This first type of features include the length of the summary, the number of sentences, the average length of a sentence, and the average tf-idf score of the words in the summary. For the second type of features, we ran the plot summaries through the online sentiment analysis tool at http://text-processing.com/demo/sentiment/. For each movie, we distilled a positive score and a negative score, describing respectively how strongly positive and negative the tone of the summary was.

5 Methods and Model Analysis Results

5.1 Linear Regression

We first seek to predict the movie revenue using a continuous variable strategy. We use linear regression for its simplicity of implementation and guaranteed convergence. Several of the previous class projects examining movie revenue performed linear regression on their feature set and generally concluded that linear regression poorly fit the data. In light of this, we chose to also perform a linear regression on the logarithm (base 10) of the revenues. In addition to being incredibly simple to implement in Matlab, this method distributes the data more uniformly and reduces the effect of larger revenue values from biasing the fit parameters (Figure 1).

We use two metrics to compare our results: the Pearson correlation coefficient and the success rate (define as a prediction being one order of magnitude of the actual revenue). The Log linear regression generally performs better than the linear regression, and we find that normalizing our non-binary features by their maximum value results in a minor improvement in the metrics for the test and development sets (Table 1). Normalization prevents features with very large values from biasing the prediction. The training set metrics exhibit the same values with or without normalization, as would be expected. Removing one effective feature at a time, we find that removal of the MPAA ratings from the feature set resulted in the largest decrease in our performance metrics, with reduction in the Pearson correlation of 0.07 and the success rate of 2.8% for the development data (Table 1). We show a scatter plot and a cumulative probability distribution for the best performing model (the log linear regression with normalized features) and find that the outliers are generally movies significantly under-performed their

	Log LR (norm ft.)	Log LR	LR (norm ft.)	LR	Log LR (norm ft., no MPAA)
Correlation (Train)	0.74	0.74	0.67	0.67	0.68
Correlation (Dev)	0.73	0.73	0.67	0.67	0.66
Correlation (Test)	0.72	0.71	0.68	0.68	0.65
Success Rate (Train)	78.3%	78.3%	61.3%	61.3%	73.9%
Success Rate (Dev)	76.8%	75.9%	62.2%	60.7%	74.0%
Success Rate (Test)	78.1%	77.1%	63.6%	60.7%	72.3%

Table 1. Pearson's correlation coefficient and success rate for the training, development, and test sets. LR is linear regression, Log LR indicates a linear regression fit to the logarithm of movie revenue, norm ft. indicates that the non-binary features are normalized by their maximum value, and no MPAA indicates removal of the MPAA rating binary features.

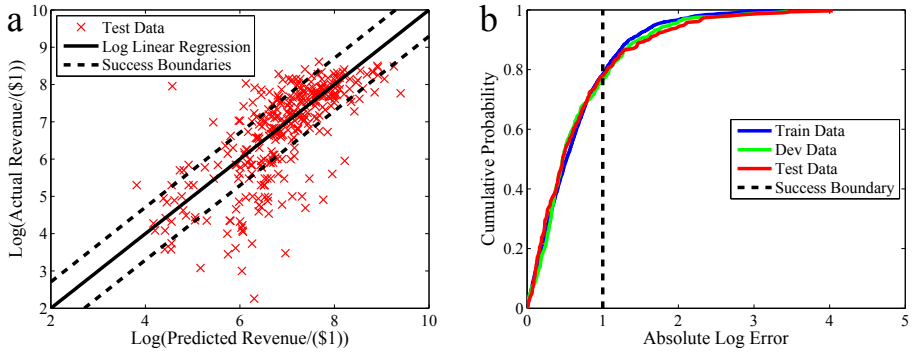

Figure 2. (a) Actual test set revenue vs. revenue predicted from the log linear regression model for the test set with non-binary features normalized to their maximum value. Data points are red x's, the black line shows the log linear regression prediction, and the black dashed line designates our success rate metric (one order of magnitude from the prediction line). (b) Cumulative probability distribution as a function of the absolute Log error for all three data sets. The black dashed line designates our success rate metric.

predicted revenue value (Figure 2). This is likely partially due to the smaller amount of data for smaller revenue movies available relative to movies that made >$1 million.

5.2 Classification Methods

5.2.1 Data Classification

The data was partitioned into buckets by rounding down the Log (base 10) of the revenue. For the training set this gave us buckets from 2 to 8 (or $100 to $100 million). This strategy gave a more uniform distribution than labeling the revenue buckets linearly (Figure 1).

5.2.2 Naive Bayes

Although we know some features are related we wanted to use a classification algorithm for multinomial classification. As stated in class Naive Bayes is simple but effective for many problems sets. We implemented Naive Bayes with pyspark's Mllib [4] implementation of multinomial Naive Bayes with Laplace smoothing (it should be noted that removing Laplace smoothing had a negligible a effect on model performance in this case). The priors are calculated as discussed in class.

We used the development set to investigate the effects of removing one class of features at a time (e.g., the genre binary features) in order to identify the most important features. Through that analysis we found that the NLP features and writer features hurt the model most. Using leave-one-feature-out after removing the NLP and writer features showed that there were no significant gains in either classification or +/-1 classification. Thus removing any additional features would potentially overfit the development set. Through this same analysis we identified the actor feature as the most important to this model. We then examined how well the model classified correctly and how often it was within 1 bucket of being correct. The Naive Bayes model correctly predicted the movie revenue test set buckets 51.6% of the time, and was within 1 bucket 79.0% of the time (Table 3).

[4] https://spark.apache.org/docs/1.5.2/api/python/pyspark.mllib.html

Bucket	2	3	4	5	6	7	8
Unbalanced	0.00%	0.00%	38.39%	18.25%	24.86%	79.75%	55.64%
Balanced	0.00%	7.84%	32.35%	45.83%	31.25%	71.28%	54.00%
Number of Data Points	15	106	350	445	571	1319	371

Table 2. Overall accuracy on the development set for the unbalanced SVM model (where every point is given the same weight) vs the balanced SVM model (where the weight of every training point is inversely proportional to the number of point in the bucket).

		Round Log_{10} (Actual Revenue)						
		2	3	4	5	6	7	8
Round Log_{10} (Predicted Revuene)	2	0% (0)	0% (0)	0% (0)	0% (0)	0% (0)	0.3% (1)	0% (0)
	3	0% (0)	0% (0)	1.9% (6)	0% (0)	0.6% (2)	0.3% (1)	0% (0)
	4	0.3% (1)	0.6% (2)	4.2% (13)	1.9% (6)	2.3% (7)	1.9% (6)	0% (0)
	5	0% (0)	0.3% (1)	1.9% (6)	2.6% (8)	6.8% (21)	1.9% (6)	0.6% (2)
	6	0.3% (1)	0% (0)	0.6% (2)	2.6% (8)	4.5% (14)	11% (34)	0% (0)
	7	0% (0)	0% (0)	0.6% (2)	0.3% (1)	5.2% (16)	31% (96)	1.9% (6)
	8	0% (0)	0% (0)	0% (0)	0% (0)	0.6% (2)	5.5% (17)	7.1% (22)

Figure 3. Confusion matrix for SVM, featuring classification on the development set (323 total movies). Green is classified correctly, orange is classified within 1 and red is misclassified.

5.2.3 Multinomial SVM

We implemented a multinomial SVM model using the python scikit[5] library. We used scikit for two reasons. First mllib did not have a multinomial SVM it only had binary and secondly we wanted to get a feel for multiple python libraries. Using backward search on the set of features, we found that we produced the best results (57.9% accuracy on the development set) with a linear kernel and a set of features that included all features except the release month as a linear value. We also experimented with training models using a Gaussian kernel and a quadratic kernel, but those performed worse in the general, producing scores on the development set of only 47.1% and 54.5% even for the best sets of features. See Table 3 for a summary of of SVM results.

If we dig a little more deeply into the results of the SVM classifier, we see that not all buckets are classified equally well. For instance, our model was able to predict the movies in bucket 7 with 79.8% accuracy, whereas in buckets 2 and 3, we see as low as 0% accuracy on the development set. This is not so surprising when we examine our underlying training data set. 1319 training data points fall into bucket 7, the largest in our data set. In contrast, buckets 2 and 3 are our two smallest, with only 15 and 106 data points respectively. It is very likely that we simply did not have enough data in those buckets to make accurate predictions. Another part of the problem is that with an unbalanced data set, the classifier tends to favor the class with the largest membership. As an experiment, we retrained our best SVM model - this time with each data point in the training set weighed inversely proportional to the size of its bucket. The overall performance of this model 43.3%, which is significantly worse than our original model. However, performance is much more consistent across the different buckets, including buckets with smaller membership (see Table 2).

5.2.4 Multinomial Logistic Regression

An alternative simple way to examine data is through multinomial logistic regression. We implemented this using pyspark Mllib. Using the development set and the leave-one-out removal of features we determined that subtracting features did not affect the model performance. The only parameter that changed the model was the regularization parameter. Fitting for the development set this parameter was set to 0.0005. The results of Multinomial Logistic Regression are similar to our other classification methods as can be seen in Table 3.

6 Conclusions and Future Work

Counter to initial expectations, the performance of the more complex SVM model was not better than that of simpler models such as Naive Bayes and logistic regression (Table 3). On the contrary, SVM with linear kernels did a little worse on the test set (49.4%) in comparison to Naive Bayes (51.6%) and

[5]http://scikit-learn.org/stable/modules/svm.html

	Naïve Bayes Test	Naïve Bayes Dev	SVM - Linear Kernel Test	SVM - Linear Kernel Dev	SVM - Gaussian Kernel Test	SVM - Gaussian Kernel Dev	SVM - Quadratic Kernel Test	SVM - Quadratic Kernel Dev	LogReg Test	LogReg Dev	Classify w/ Average (7) TEST	Classify w/ Average (7) DEV
Correct	51.6%	52.0%	49.4%	57.9%	48.1%	47.1%	46.8%	54.5%	50.9%	57.5%	39.0%	43.0%
Within 1	27.4%	29.4%	39.4%	28.8%	32.6%	35.3%	36.8%	30.7%	34.2%	30.0%	32.2%	28.5%
Within 2+	21.0%	18.6%	11.3%	13.3%	19.4%	17.6%	16.5%	14.9%	14.8%	12.4%	28.7%	28.5%

Table 3. The performance of different classification models (Naive Bayes, SVM, multinomial logistic regression) are compared for the test and development sets

logistic regression (50.9%). SVM with higher order kernels did worse still. However, this result becomes less surprising when we consider the size of data set. Splitting this data into seven buckets leaves the majority of those buckets with fewer than 500 data points. Simpler classification methods that rely on more assumptions about the data, such as Naive Bayes, tend to shine when the data set is very small. It is possible that we could achieve significant improvements in the SVM model if the movie data set were larger. In our current analysis, we found that all three models classified a movie to the same bucket 70% of the time, and within one bucket of one another 85% of the time. Again we attribute this small variance in model performance (which we expected to be larger) to the data set size.

The sparsity of data in certain buckets is a weakness in all of our models. This is evident by the fact that our accuracy on a per bucket level range from 0% in the smallest buckets to nearly 80% in our largest bucket (Table 2 for SVM). This discrepancy holds across all of the classification models.

The movie that was misclassified the highest by all 3 of our algorithms and under performed with the largest log linear regression error was "Meeting Evil". IMDB says this film by Magnolia Pictures made $181 at the box office and stars Samuel Jackson and Luke Owen. Since our actor feature score is simply the number of movies they star in, it is understandable that our models predict this movie to have a high box office revenue. In general, under performing movies had very high director or actor feature numbers, and were essentially box office "flops". It also worth noting that Magnolia Pictures is not a "major" film studio and generally produces smaller budget movies that produce less revenue.

The movie that was misclassified the lowest by all 3 of our classification models and over performed with the largest log linear regression error was "Space Station 3D". This was an unrated 45 minute IMAX documentary film about life on the International Space Station with narration by Tom Cruise. Our training set had few other IMAX films and documentaries usually generate little revenue. Also, this movie was unrated, so it didn't have an MPAA rating feature score, which was the strongest contributing feature for linear regression (removing MPAA rating features from the linear regression models greatly decreases this movie's outlier status). The short run time also contributed by lowering the prediction score in a manner not consistent with the assumed linear trend for regular films that are generally 1.5 to 3 hours long. This movie was also classified much lower than its actual performance despite having Tom Cruise counted for the actor feature. Interestingly, the director Toni Meyers has only directed three films, but all her projects make tens of millions of dollars despite being documentaries.

Some of our outliers could likely be better accounted for by strategically adding features for trends currently unaccounted for (we generally want to avoid unnecessarily adding more features when we've already determined the data set is relatively small for our modeling methods). Most of our misclassified and highest error movies under performed their prediction, and generally were labeled on Rotten Tomatoes as "limited release". Adding a "limited release" feature would likely help better classify and predict revenue for these movies, which generally made less than a million dollars. We could also add another binary feature for whether or not the studio backing the film is a "major" studio, which could better account for films made by smaller studios like Magnolia Pictures. Another strategy may be to try different actor and director features (such as net worth or number of online search results).

In our project we showed that movie revenue forecasting is amenable to predictive modeling by supervised learning methods. By obtaining data for all movies with available online data from 2000 to 2015, we generated the largest modern data set we could to perform both continuous fitting and multiple-label classification methods. In general, we obtained a similar level of predictive power for all our models using our performance metrics, likely due this data set still being relatively too small. We consider our models to be a very coarse estimate of revenue prediction, but we are ultimately skeptical that even a very sophisticated feature development and selection would result in significant improvements. There are many factors that influence movie revenue that are difficult to quantitatively measure and even harder to predict. Instead, these sorts of models would probably work best complementing a human's analysis and intuition, preventing unconscious (or conscious) biases leading to a poor investment of studio resources.