Predicting Movie Revenues

Jeff Kessler (jakessle@)
Sam Lite (slite@)
Daniel Penner (dpenner@)
Rohan Sharma (rohans@)

May 13, 2014

Introduction

We examined methods of predicting overall movie revenue from movie characteristics that are available during production. In particular, we used movie features such as genre, director and actors to predict revenue. We used a few different sets of features to predict movie revenue, as will be discussed in the paper. However, we made sure to only use data readily available during production. In much of the literature that we found online, features only be available after production were also used to predict revenue. While perhaps intellectually interesting, models such as these lack utility to production studios deciding to make a movie. In the models we built, one could make predictions about box office revenue during production, a very useful model. As will be explained, we developed a testing scheme to simulate movie prediction in chronological order; and we considered both regression techniques as well as classification techniques in order to learn more about our data.

Our hypothesis is twofold: (*i*) Using information available during production, we can build a model to predict a movie's box office revenue with reasonable accuracy. (*ii*) Models that take actors and directors' prior success into account will tend to predict revenue more accurately than those that do not.

In order to test these hypotheses, we consider a sequence of generally increasingly sophisticated models. We first identify a baseline error, the mean absolute deviation from average revenue, which gives the average error for simply guessing the mean revenue for each movie is \$46.50M. Then, we estimate two ordinary least squares regression models, as well as the corresponding lasso and ridge regression models. The first model used very basic features about the movies (e.g., budget, MPAA rating), while the second included information about the prior successes of the actors, directors, writers, and production studios involved. For both of these models, we separated our data into training and testing sets with an approximate 80-20 split. Our subsequent testing model mimicked a walk forward through time, in which we predict the revenue of each movie based on all data chronologically prior to it. Using this time-marching scheme, we ran both linear regressions (standard, ridge, and Lasso), and a variety of classification methods to try and correctly place movies in revenue "buckets." We also considered a version of online learning, which we will describe later.

Data

Our first task was to gather data. We used Python/BeautifulSoup to scrape each individual IMDb movie page for the following data on the $6,998$ movies since 1984 with revenue greater than \$400K: title, revenue, budget, runtime, genre, MPAA rating, release date, stars, full cast, directors, production studios, writers, and plot tagline. We omitted observations with missing data, such as budget, runtime and too few stars, or non-standard MPAA ratings (We only kept those movies with ratings of G, PG, PG-13, or R.) We also removed all documentaries from our dataset, conjecturing that both the relevant feature set and consumer behavior differ idiosyncratically for documentaries. We considered movies only from the past thirty years to avoid the introduction of changing consumer tastes and industry characteristics (e.g., changes in active actors) that occur naturally over time. After these data adjustments, we were left with 3696 data points. Finally, to account for inflation, we adjusted revenue and budget figures on our data points using a GDP deflator[1], expressing all quantities in 2009 dollars.

Methodology

We now give an overview of the regression techniques used. As mentioned earlier, our benchmark mean error for this data, as established by just guessing the mean, is \$46.5M.

A Naïve Regression

For our first regression technique, which we shall call "Naïve Regression", we considered the following features: budget, runtime, the MPAA rating, genres and month of release. For mpaa ratings, genres, and months, we assigned each an indicator function for each possible value. The reason why we chose indicators for each rating rather than translating them to categorical variables was so that we did not build any assumptions about the "order" of rankings, months, or genres into our model. We tested this regression by dividing into training and testing in two different ways. First, we randomly set 20% of the data aside for testing (simply using the built in function in R to randomly select 20 percent of the numbers between 1 and 3696.) For the second test, we set aside the chronologically-latest 20% of data (roughly all data after 2009.) Our results for training this regression model and running these two testing

[1]http://www.multpl.com/gdp-deflator/table

schemes are summarized in the tables below (the results are for optimal scaling parameters for ridge and Lasso):

Table 1: Random Test Set Error (millions)

	MLE	Ridge	Lasso
Train	\$33.17	\$33.15	\$32.77
Test	\$35.16	\$35.16	\$34.21

Table 2: Sequential Test Set Error (millions)

	MLE	Ridge	Lasso
Train	\$33.27	\$33.26	\$32.90
Test	\$34.37	\$34.36	\$33.74

Here, we see a noticeable improvement on our benchmark mean error of \$45M. Moreover, we notice that the Lasso performs better than both standard and ridge regression. Additionally, the Lasso selects out a number of the features in our model. By examining which features Lasso removes, we might get better insight into the importance of the remaining features. In particular, certain months (January, February, March, and August), a number of genres (Action, Adventure, Crime, Family, Fantasy, Horror, Musical, Mystery, Romance, Short, and Sport), and the MPAA rating PG-13 were selected out. These results have some reasonable interpretations, as follows.

First, the months that were assigned coefficients of zero can be considered the neutral months of the model. Of the months that remained as features with non-zero coefficients in the Lasso model, April, September and October received negative coefficients while May, June, July, November and December received positive ones. This aligns with our intuition that the holidays and the summer are the highest-grossing movie months. Next, the genres which the Lasso model selected out are in general genres which are revenue-neutral, in that they have a high variability of box office success, and therefore the fact of a movie falling into one of the removed genres tells us little about whether or not it has high revenue. Similarly, the fact that the MPAA rating PG-13 was selected out reflects that it is by far the largest rating class, and thus its movies have a very high variability in revenue. In contrast, the MPAA rating R received a negative coefficient, indicating that movies of this class have lower revenues, on average.

While this regression technique certainly produced interesting results, we hoped to incorporate the actors, directors, writers and production studios involved into our prediction model. In order to do this, we needed to establish a numerical value for each of these entities. One possibility we considered was to represent each unique actor, director, production studio and writer by an indicator function, indicating whether or not a certain entity takes part in a movie. However, this would have resulted in a model with far more features than we have data points. In particular, there are a total of 19,978 unique actors, directors, studios, and writers, while (after filtering for irrelevant data points, as described earlier) we only have a dataset of 3696 movies. Thus such a model would drastically underfit to most of the features, and would assign overly simplistic coefficients to most of the rare features (that is, actors, directors, studios or writers that do not appear often). In theory, if our data set were much larger without increasing the number of actors, directors, studios and writers, this model would perhaps work well and be worth exploring further. Instead, we set out to try to use the average revenue that actors/directors/production studios/writers have made in other movies to predict revenue for the current movie.

At this point, we pause to make a methodological note. In our first regression technique, "Naïve Regression," we first randomly withheld 20% of our movies from the model fitting process to use for testing. As a result, the model was fit using information from the future to predict revenues of movies made in the past. While this was somewhat informative as a test model, our predictor can only be useful if it exclusively uses data from the past to predict a movie's revenue. That is, for a given movie, we would like to only use data available during the production of that movie to make a prediction about the revenue of that movie. Moving forward, we will only consider data in this form to make predictions.

Regression with Creative Entity Metadata

For our second regression attempt, as with one method in our first regression attempt, we used all movies from before 2009 as our training set and all movies from after 2009 as our testing set, giving the desired quality of only using information available during production. This time, we incorporated what we call "creative entity metadata," which is a representation of the revenue history of each "creative entity" in a movie's production: the production studio, the writers, the directors, and the stars. The idea is that a creative entity's historical movie revenue will inform the revenues of their future movies.

In order to incorporate creative entity metadata, we introduced six new features to our model: star 1 average revenue, star 2 average revenue, star 3 average revenue, director average revenue, writer average revenue and director average revenue (note that IMDb lists each movie as having three stars). Then using our training set of movies, we computed these average revenues for each actor, director, writer and studio that appears, and stored the result in a JSON file. We then ran a regression as before, including the average revenues (as listed above) as features.

Table 3: Regression Two (Error, in millions)

	MLE	Ridge	Lasso
Train	20.10	20.10	19.19
Test	37.12	37.12	36.89

Note that here, as is true above, we report the results for the ridge and Lasso models with optimal scaling parameter. Note that the drastic improvement in the training error can be attributed in some part to the fact that the revenue of movies in the training set (the response variables) were used to calculate the average movie revenues of actors (instances of the features). Because of this, we ought to disregard the training error of this model.

Now note that the test error has actually increased in this model. In order to make sense of why the test error has increased, we point out that many stars of movies made after 2009 might not have appeared in any high-grossing movies in our training set. At first, we initialized these actors' average revenue fields to zero when trying to make predictions in the test set. With this setting, the model performed even worse than above, with average test error of over $40M. Instead, we initialized the actors' average revenues at the median in the training set, with which we obtained the results above. To make matters worse for this model, in some situations an actor may have highly misleading pre-2009 data, instead of no data. For example, the average revenue of Jennifer Lawrence's movies from before 2009 was very low, but she starred in five high-grossing movies in our test set. The model will naturally underperform on data points such as these. (The same point can be made about directors, production studios, and writers though to a lesser degree.)

"March through Time" Testing Scheme

In order to remedy the problem discussed above, we implemented a testing method which we call "March through Time". Essentially, we sort the movies chronologically, and progress through time, recomputing our model's parameters (using the movies released prior to the movie being predicted) each time a new movie is released. In this way, each movie has its own training set consisting of every movie chronologically preceding it.

In implementation, we maintained a dynamic dictionary of all actors, directors, studios, and writers, in which we stored the average revenue of their past movies, as well as the highest single revenue of their past movies. As we march through time in the algorithm, this dictionary is updated with the current movie information. We begin by populating the dictionary with the correct metadata given the (chronologically) first 100 movies in our database, and fitting a linear regression model using these movies. Then starting at $i = 101$, we proceed as follows through time: (1) Fit a linear regression model using movies $1, \ldots, i-1$; (2) Use it to predict the revenue of data point i. Compute and store the prediction's absolute error; (3) Update the running averages of actor, writer, studio, and director past revenues using the actual revenue of movie i, and proceed to movie $i + 1$. (see the diagram in Figure 1 for a visual representation)

We list our results below. The first row's model simply represented each creative entity by its average prior movie revenue, while the second row adds an extra feature for each creative entity: its most recent prior movie revenue. With this extra feature, we attempted to capture some of the momentum and "hype" that might surround an actor at the time of a movie's production and which might contribute to its success. However, this model performed worse than the simpler one, and so we omitted the extra features from the rest of our subsequent models. A topic for further investigation could be to experiment with other possible feature additions to attempt to capture momentum or hype of a creative entity, which we hypothesize contributes substantially to a movie's revenue.

Table 4: March Through Time Testing Error (in millions)

	All data (after first 100)	Last 20% of Data
Without Most Recent Revenue	$34.60	**$32.72**
With Most Recent Revenue	$35.42	$34.39

Note that an important feature of the “march through time” testing scheme is that the model becomes more robust over time, as the size of the training set increases. Thus to evaluate the strength of the model, it makes sense to focus on the error of the later predictions. In this vein, we also consider the average prediction error on the (chronologically) last 20% of movies, shown in the second column of the above table.

Now notice that the testing error has decreased below that of our Naïve Regression technique. This is strong evidence that using a regression technique that incorporates this metadata outperforms the regression techniques that does not. That is, using this much more robust test of our regression model, we see an improved testing error over the naive method.

Note that this testing method is similar in spirit to online learning, in the sense that we obtain new data over time and update our model accordingly. To explore this analogy further, we fit an online linear regression model as well, to see how it performed compared to our previous model. Here we simply fit a linear regression model on the first 100 movies (chronologically), and subsequently updated the coefficients in the direction of the gradient of the prediction error at each new data point. However, this technique did not perform well at all. One possible reason is that our feature values differ dramatically in scale (from movie revenues in the millions to the indicator functions of the genres). The literature suggests that we can center our data’s means at zero and normalize by the standard deviation. However, the fact that our means change accordingly with our updated actor, director, studio, and writer data over time, we could not simply apply the standard normalization technique. Instead, we tried to normalize the feature values by dividing by their average each step (which itself varied over time). However, the results were still rather poor and not worth reporting.

Classification

After performing the above-described regressions, we decided to reformulate the problem in terms of classification. The impetus for considering classification models was twofold. First, classification has interesting applications for this problem in its own right: a movie studio might just be interested in whether or not a movie will be above or below a certain threshold (for instance, perhaps a movie only gets a green light if it is predicted to gross above a threshold). The second impetus for considering classification was to further explore our results for regression. We will see that we gain insight as to where our regression model succeeds and fails by evaluating its performance when restricted to data which is correctly classified by our classification model.

As mentioned earlier, we consider binary classification for whether each movie grosses over $100M. If the revenue of a movie was greater than $100M, we placed the movie in class 1; otherwise, we placed it in class 0. After using the first 100 movies to train our classifier (which we separate as before in our March through Time scheme), the class label distribution of the remaining data points is 3063 with class 0 and 533 with class 1. We also discretized all the creative entity metadata, by simply classifying entity’s revenues in class 0 or 1 (since the entity’s feature values were themselves revenues as well) so that all data in the model is binary.

As a benchmark, if we were to just guess by assigning all movies to class 0, we would correctly identify $3063/3596 = 85.2\%$ of the movie classes in particular, we would guess class 0 with 100% accuracy and class 1 with 0% accuracy). If we were to somehow know the distribution and then guess using this distribution, we would correctly identify class 0 with accuracy $0.852^2 = 73\%$ and we would correctly identify class 1 with $0.148^2 = 2.2\%$ accuracy.

We first ran a Bernoulli Naïve Bayes Classifier using Python’s scikit-learn library. We applied the March through Time testing scheme discussed earlier, as this method provides an excellent approximation of how our prediction models would be used in practice. This classifier predicted class 0 with accuracy of 2655/3063 and predicting class 1 with accuracy of 346/533, a noted improvement over the baseline.

We also attempted three other classification techniques, also using Python’s scikit-learn library: logistic regression, a random forest classifier (RFC), and a support vector classifier (SVC). See table 5 below for a summary of the results, and the figures 2-4 for visualizations of how these methods’ mean errors changed as we marched through time.

We note briefly that the overall prediction error of RFC and SVC is highly misleading, since class 0 is far larger than class 1, and since these methods were heavily slanted towards class 0, their average accuracy appears high. However, Bernoulli NB performed by far the best on class 1, and hence we will stick with this classification method as we proceed in the next section. Note also that an interesting and useful topic for further investigation would be multinomial classification techniques, to try and place movies into revenue buckets.

Table 5: March Through Time Classification Accuracy

	Bernoulli NB	Logistic Regression	RFC	SVC
Prediction Accuracy in Class 0	86.7%	96.0%	96.0%	99.6%
Prediction Accuracy in Class 1	64.9%	36.6%	29.8%	9.0%
Overall Prediction Accuracy(weighted)	83.5%	87.2%	86.2%	86.2%

Informing Regression Results Using Classification

After running these regression and classification techniques, we came up with the following hypothesis: perhaps the mean prediction error in our regression was thrown off by a small number of very bad predictions, masking that the model may have performed substantially better on the majority of movies. We make the guess that, if this is the case, then these bad predictions will be filtered out by the classification method (that is, they will be wrongly classified).

To test this hypothesis, we simultaneously ran the Bernoulli NB classifier and linear regression (with creative entity metadata) in a march-through-time scheme, and saved both the classification predictions and the revenue predictions for each movie as we progressed. This allowed us to analyze the regression error on various equivalence classes under our classification scheme. In particular, we found that the average revenue prediction error for movies which were correctly classified as making below \$100M was \$24,232,047, while the error for movies correctly classified as making over \$100M was \$73,279,894. First, note that the under-\$100M regression error is a substantial improvement over our general average error. This is important, since these movies have smaller actual revenues and thus their prediction errors ought to be below the overall average. Similarly, the error of \$73M for high-grossing movies aligns with the fact that these movies made more money, and thus the proportional error remains about the same.

As before, we consider the fact that our march through time testing scheme is best evaluated at the later stages, where the training set is larger. So we also consider the last 20% (chronologically) of movies that were correctly classified in either class 0 or class 1. These results are drastically lower than the overall average, as we might expect, as summarized in table 6 below.

Table 6: Testing Error on Correctly Classified movies (in millions)

	All Data (after first 100)	Last 20% of Data
Average Error in Class 0	\$24.32	\$20.58
Average Error in Class 1	\$73.28	\$19.28
Overall Average Error (weighted)	\$31.57	**\$20.39**

Note that the average error on the last 20% of data for both classes is substantially improved. In fact, our mean prediction error on correctly classified movies which gross over \$100M is \$20M, which is at most 20% of any of the movie revenues in that class. Thus we see that, when our model makes at least somewhat accurate predictions on blockbuster movies, its predictions are actually quite accurate, on average. Further research ought to focus on methods to predict the mis-classified cases, which we conjecture consist largely of unexpected smashes and big flops. Naturally, predicting these would be a substantially more difficult, but highly useful model.

Conclusion

We hypothesized that we could predict movie revenue from data available before or during production. We also hypothesized that our prediction would improve when using actor data and when accounting for chronology. The results of our analysis showed these hypotheses to be correct. Our first regression using only basic variables available during production predicted movie revenue with reasonable accuracy. Our second regression then tried to incorporate meta data such as average actor revenue. Due to partitioning data into pre-2009 and post-2009, we missed a lot of information and our second regression performed more poorly. In order to better test this method, we implemented a march through time where we essentially mimicked being at the pre-production stage of a movie (and having only the information available at that time) for each point in the dataset and tried to make a prediction using available information. When implementing this testing technique, our training error on the last 20 percent of data decreased. We then tried classification and then when using this classification to inform our regression results we found that on a large set of movies, our regression performed very well.