Hit or Flop: Box Office Prediction for Feature Films

December 13, 2013

Dan Cocuzzo
dcocuzzo@stanford.edu

Stephen Wu
shw@stanford.edu

This project is an investigation of the effectiveness of various machine learning techniques on the prediction of success for feature films, as measured by their gross earnings and average user rating. Data for approximately 5,000 feature films was used to train two distinct models, and overall classification results and performance analysis are reported here. The code for this project may be found at `https://github.com/shwu/imdb-project.git`.

Introduction

Feature films are a multibillion-dollar industry. Given the sheer number of films produced as well as the level of scrutiny to which they are exposed, it may be possible to predict the success of an unreleased film based on publicly-available data. A large amount of data representing feature films, maintained by the Internet Movie Database (IMDb), was extracted and prepared for use in training several machine learning algorithms. The goal of this project is to build a system that can closely predict average user rating and degree of profitability of a given movie by learning from historical movie data. Since there is a strong correlation between a film's budget and the gross US earnings, predicting raw gross earnings is not particularly indicative of a film's success. Instead, we transform the gross earnings of a film to a multiple of it's budget, which is a much more meaningful indicator of a film's success.

Data

Movie data is drawn from the Internet Movie Database (IMDb), found at `http://www.imdb.com/`.

Access

IMDb makes its data publicly available for research purposes from which a local, offline database can be populated. FTP servers sponsored and maintained by IMDb contain stores of flat-format `.list` files that contain the same information found online through IMDb's web interface. For this project data access and preparation was facilitated by the help of two existing software systems: `sqlite` and `imdbpy`. `sqlite` is a widely-used SQL implementation supporting all standard SQL constructs and can be used to query all information found in the database in a high-level, declarative manner. `imdbpy` is a freely-available Python package designed for use with the IMDb database that implements various functions to search through and obtain data. Python scripts were devloped to automatically pull the required feature data from the local `sqlite` database.

Pruning

The full database contains nearly 3 million titles, of which roughly 700,000 are feature films. Many of the titles found in the database contain incomplete information or are inappropriate for the scope of this investigation. Thus, in an attempt to both decrease training time and increase the accuracy of the prediction, the full title list was pruned using a series of SQL queries. The criteria by which IMDb titles were omitted are as follows:

- Titles which are not movies (e.g. TV, videogames, etc).
- Adult films
- Films missing budget data in US dollars
- Films missing gross earnings data in US dollars
- Films missing user rating data
- Films not released in the United States

After pruning the entire database of nearly 3 million entries, only 4260 titles remain (less than 0.002% of the original database). While this quantity is a tiny fraction of the overall database, the pruning constraints enforced are justifiable for the purposes of this prediction system; the pruned titles include films which are not released in major theater circuits, films we cannot generate labels for, and films not released in the US.

Note that gross earnings reported in US dollars (our focus here) correspond to earnings from US theaters only, and therefore the financial-based metrics for film success are strictly an indicator domestic performance.

Features

Currently, the following features are drawn from each training film:

- cast list
- director(s)
- producer(s)
- composer(s)
- cinematographer(s)
- distributor(s)
- genre(s)
- MPAA rating
- budget
- release month
- runtime

Several prediction models were implemented to learn from these features and make predictions on the success of films drawn from a test set. The prediction comes in the form of two primary success metrics.

Ratings prediction. Many of the movies listed on IMDb contain an average user-rating on a scale of 0 to 10 which corresponds to public opinion of that movie. The rating values exposed by the IMDb are rounded to a single decimal point; in this analysis we have rounded rating values to the nearest integer. Thus, the system predicts rating rounded to the nearest integer, turning a regression problem into a classification problem with 11 effective classes representing the integers 0-10 inclusive.

Gross earnings prediction. Since the magnitude of gross earnings is not necessarily representative of movie success, especially if the movie had a large budget, the system predicts gross earnings of movies as a multiple of their budget, or *bmult*. We label the bmult of each of movie into the following bins: $[0-0.5), [0.5-1), [1-2), [2-\infty)$. As with rating predictions, this significantly simplifies the models by reducing the space of output classes. Note that bmult is a rough approximation of a film's of return on investment, where bmult greater than 1 corresponds to a profitable movie.

Methods

Naïve Bayes

Under the naïve Bayes assumption, a movie can be represented as an independent combination of its associated personas and attributes, as denoted by the features described in the Introduction. In this case, we have

$$P(rating = r|movie) \propto P(movie|rating = r)P(rating = r)$$

where $P(movie|rating = r)$ is the product of the individual conditional probabilities for each persona (e.g. $P(actor_i|rating = r), P(director_i|rating = r)$, etc.) and attribute (e.g. $P(genre_j|rating = r), P(MPAA|rating = r)$, etc.) associated with the given movie. Laplace smoothing with $\alpha = 1$ was also used to account for sparsity in the dataset to avoid zero-valued probability estimates in the empirical training data. This is a required procedure since it is unreasonable to assume that each actor in the database possesses a filmography containing all possible ratings and all possible bmult bins. Furthermore the cast list for each film is limited to a tunable parameter, `MAX_ACTORS`, since actors with more screen time are more likely to contribute to a film's success. Using a larger value for `MAX_ACTORS` may improve prediction performance due to the addition of information at the cost of increasing model complexity. The appropriate choice of the value of `MAX_ACTORS` is discussed further below in the section titled Parameter Tuning.

Training. The multi-variate Bernoulli event model is used; that is, the assumption is that all actors, directors, genres, etc. are drawn independently from the same distribution(s). Training this naïve Bayes model involves calculating the probability that each persona or attribute exists in an arbitrary movie, conditioned on the particular output bin we wish to find the likelihood for.

Prediction. Classification is performed by caculating the posterior probabilities of the output bins for each test movie. That is, the maximum-likelihood prediction of $P(rating|movie)$ and $P(bmult|movie)$ is used to classify a given test movie.

Support Vector Machine

The goal of the SVM formulation is to generate separating hyperplanes in the feature space which allow us to classify input movies. In class, we typically saw examples of SVM for binary classification. In this case, multiclass prediction is done using the one-versus-one method, where a binary classifier is generated for every pair of possible output labels, and the predictor outputs the class selected by the greatest number of classifiers.

We choose the radial basis function kernel, since it is an oft-recommended default kernel for nonlinear models. The RBF kernel is defined as:

$$K(\mathbf{x}, \mathbf{z}) = \exp(\gamma \|\mathbf{x} - \mathbf{z}\|_2^2)$$

where γ is a free parameter.

Training. `libsvm` is a library which implements various SVM classifiers with user-definable parameters. We use Python to interface with this library to train predictors.

Feature vectors and output labels are generated for each movie in the training set. Unfortunately, the pruned dataset contains over 37,000 different personas (actors, directors, producers, etc.). Since we base our features on inclusion (binary features) this would lead to a vector with a length of nearly 40,000 features. Additionally, performing normal forwards/backwards searches on this feature set would have required a prohibitive amount of time and processing power due to the number of training examples in conjunction with the overhead of accessing a single movie from the local database.

The IMDb database assigns unique IDs to each persona; thus we might consider features of the form "director ID" or "actor #3 ID". However, there is no logical ordering of these personas in the database, and certainly no predictable relation between a person's ID and his/her proclivity towards working on a successful film. Thus, the SVM formulation considers a limited feature vector which includes budget, genre, MPAA rating, runtime, and release, all of which are enumerable except for genre. These feature vectors are normalized and used to fit the SVM model. See Parameter Tuning for a discussion of the parameters used in this model.

Prediction. Prediction is performed by using the built-in `libsvm` prediction function, applied to the previously-fitted model.

Parameter Tuning

Naïve Bayes

Cast list length. The `MAX_ACTORS` parameter determines how far the model looks down the cast list in order to train predict on an example. It is reasonable to assume that the most important actors (i.e., the ones which receive the most screentime and publicity) tend to appear high on the cast list. Increasing this parameter improves accuracy locally, but in the long term will lead to increased computational and storage complexity since we must store conditional probabilities for a greater number of actors. Additionally, the naïve Bayes classifier is not cognizant of the cast list ordering since it predicts based only on inclusion. This means that extremely high values for this parameter will cause minor cast members to heavily sway the prediction. Several values of `MAX_ACTORS` were used to train and test on the dataset (see Table 1); 10 was chosen as it yielded an acceptable balance of prediction accuracy and complexity.

Output bin boundaries. Output bins for budget-multiple were selected such that the distribution of movies within the bins is relatively uniform; see Figure 5 for the resulting distribution. This decreases the chance that a large budget-multiple prior dominates the prediction. For example, using bins of size 0.25 leads to a relatively high prior for the "0-0.25" budget-multiple bin (i.e. films that tank tend to tank badly) causing most predictions to be placed into this bin. While this performs respectably in terms of error rate, this result is not very enlightening and leads to a high false negative rate for strong movies.

Since the ratings prediction contains more bins, we did not carry out this procedure for that model, as the resulting gains would both be smaller and also render the output less readable (e.g., it may not make sense to have a "3.5-4" rating bin). Note that this means good movies tend to be underrated by the prediction system, and bad movies tend to be overrated. We deemed this acceptable, as particularly strong or weak movies will still stand out in the predicted ratings.

Support Vector Machine

There are two parameters to be chosen: C, the SVM penalty parameter, and γ, the RBF kernel parameter. A grid-search was performed on these parameters, varying them independently and exponentially from 2^{-2} to 2^{7}. From this, $(C_{rating}, \gamma_{rating})$ and $(C_{bmult}, \gamma_{bmult})$ are chosen to minimize prediction error. See Figure 1 and Figure 2 for the results of this sweep. The values $(C = 0.5, \gamma = 0.5)$ were found to work well for both predictive models.

Results

The predictors performed moderately well on the test data. Qualitatively, many of the rating and bmult predictions were exactly correct, while ones that were incorrect were "close" in the sense that the predicted value-bin was typically adjacent to the actual value-bin. For instance, an incorrectly predicted movie with a true average-user rating of 7 is most often a rating of 6. Figures 7, 8, 9, and 10 show the confusion matrices of rating and bmult prediction results to illustrate the distribution of misclassifications for rating and bmult. We not only report absolute error (correct or incorrect classification) for each test sample, but also the error as a measure of absolute distance from the true value. The absolute distance error of rating and bmult predictions are a valuable indicator of our system's performance, especially when misclassified test samples are 'close' the to true rating or bmult bin.

Note that this is similar to the approach taken by typical regression problems, in which mean absolute error or mean squared error is often the quality metric of choice. Indeed, since our classification bins have a natural ordering for both average user rating and bmult gross earnings, such a metric is likely to be more accurate that a simple measure of error. If not for the issues described earlier in constructing a meaningful SVM feature vector, it is likely that a regressive approach may have yielded similar results.

Naïve Bayes

The distribution of absolute distance error of average user rating predictions is shown in Figure 3, and distribution of absolute distance error of budget-multiple gross earnings predictions for is shown in Figure 4. Both figures report error for a 70%/30% holdout test. These figures also report the priors for both problems, though conditional probabilities for the personas/attributes are omitted for brevity.

A 10-fold cross-validation was performed across the entire set of 4,260 movie titles, and the test error rates for rating and bmult predictions along with random prediction performance are shown in Figure 11. A summary of testing and tuned parameters is provided in Table 2.

Support Vector Machine

Alongside Naïve Bayes model performance, Figures 4 and 6 show the distributions of absolute distance error of average user rating predictions and budget-multiple gross earnings predictions respectively for a 70%/30% holdout test.

10-fold cross-validation was also performed for SVM, the results of which are shown in Figure 12.

Discussion

A few general points to take away from the results of this experiment:

The results of naïve Bayes parameter tuning suggest that varying the value of `MAX_ACTORS` has little impact on prediction performance. Intuitively, it would seem that using information about a greater number of actors per movie for training would improve prediction performance. However, the vast majority of actors collected from the training set appear in one or very few movies, so often the actor probabilities conditioned on rating and bmult contribute negligible information.

The confusion matrices reveal that movies are overwhelming predicted to have a rating of 6 or 7. Though the system is still able to distinguish between good and bad movies, the range is extremely compressed. Fixing the rating bins in the same way as bmult bins may alleviate this problem. SVM performed slightly better at this task, despite having a more limited feature vector.

Both classifiers are excellent at finding highly unprofitable movies. The error for movies earning back less than half of their budget was under 20% in both cases. However, Table 11 reveals that the naïve Bayes model is superior for these two particular prediction tasks given the constraints imposed. We posit that SVM performance may improve dramatically after performing sufficient feature selection/extraction and training with a larger sample set; this would be a suitable path for futher exploration on this project.

Tables and Figures

Table 1
Naïve Bayes parameter selection

`MAX_ACTORS`	Rating error	BMult error
0	0.604	0.536
2	0.604	0.523
4	0.611	0.523
6	0.608	0.523
8	0.601	0.521
10	0.600	0.521

Figure 1. SVM parameter selection for rating prediction

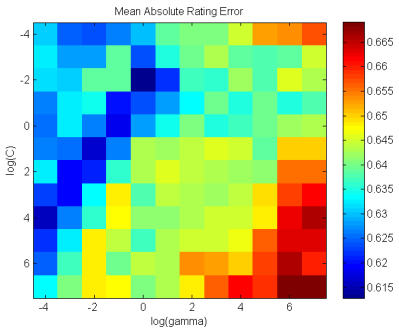


Figure 2. SVM parameter selection for bmult prediction

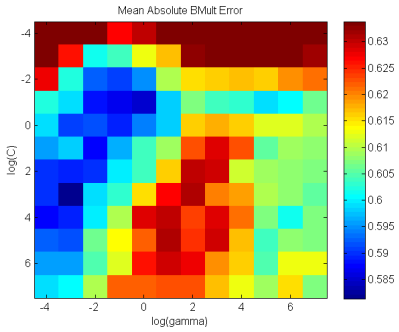


Figure 3. Distribution of Movie Ratings

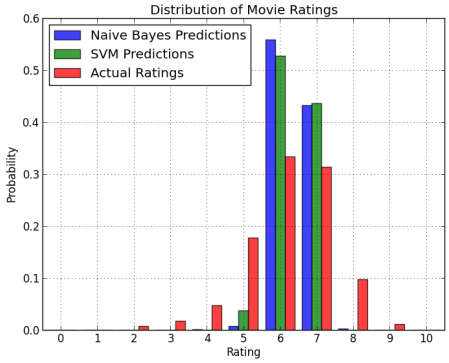


Figure 4. Distribution of Test |Error| in Rating Predictions

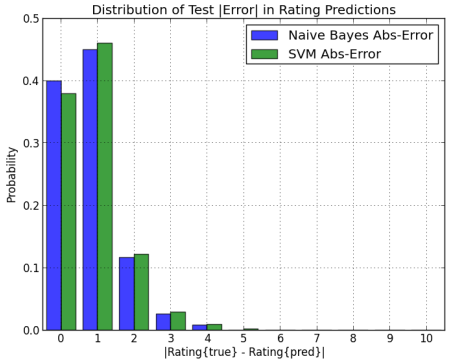


Figure 5. Distribution of Movie Budget-Multiple Gross Earnings

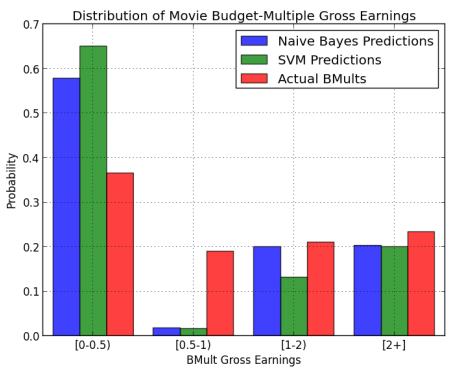


Figure 6. Distribution of Test |Error| in BMult Predictions

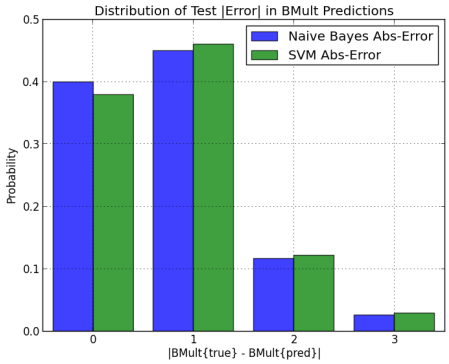


Figure 7. Confusion matrix for NB rating prediction

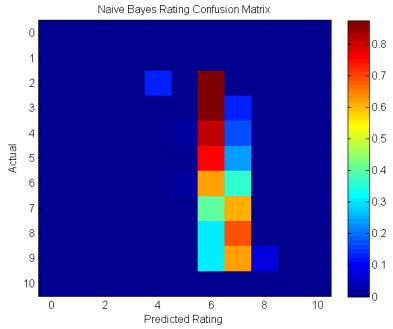


Figure 8. Confusion matrix for SVM rating prediction

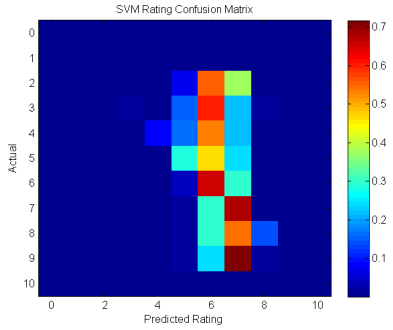


Figure 9. Confusion matrix for NB bmult prediction

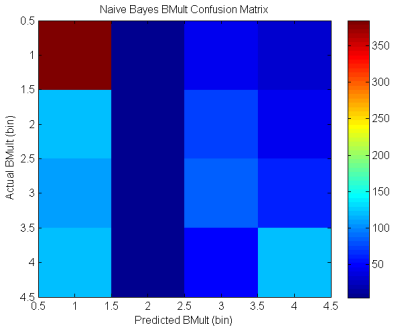


Figure 10. Confusion matrix for SVM bmult prediction

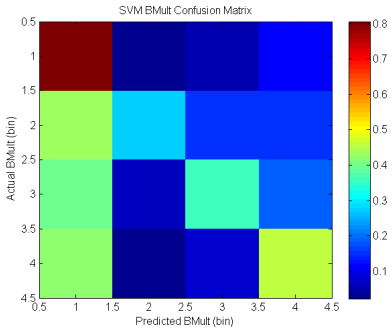
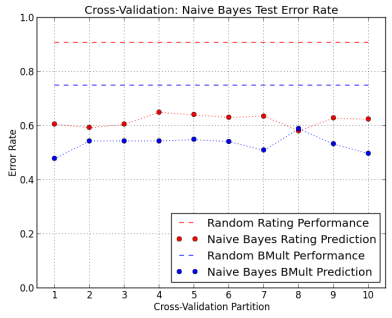


Figure 11. Cross Validation: Naïve Bayes Test Error Rate



/

Figure 12. Cross Validation: SVM Test Error Rate

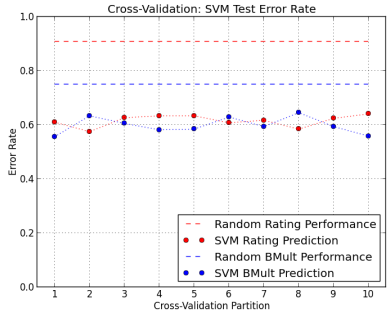


Table 2
Example test results, 70/30 holdout validation

	Naïve Bayes	SVM
`MAX_ACTORS`	10	n/a
C	n/a	0.5
γ	n/a	0.5
Rating error rate	0.600	0.621
Mean $\|ratingerror\|$	0.792	0.832
BMult error rate	0.531	0.587
Mean $\|bmulterror\|$	0.928	1.122