Movie Success Predicator and Two Brand-new Bagging Algorithms

Zhongtian Qiu
Department of Computer Science
University of Toronto
Toronto, ON, M5S 2E4
zqiu@cs.toronto.edu

Qiannan Gao
Department of Computer Science
University of Toronto
Toronto, ON, M5S 2E4
ameliagao@cs.toronto.edu

Abstract

Machine learning can provide scientific supports for decision-making. Without data analysis, it is hard to make judgment on the successfulness of investment based on intuition, with fewer sample data, and huge number of factors to consider. This paper takes movie rating as an example to analyze and compare various machine learning methods performance on high dimensional data with much fewer samples. The data used is crawled from the publicly available website IMDb. In addition of the traditional methods, the author proposed weighted bagging, and BFS-bagging to optimize the performance. The paper also examines the data with chi-square and find out the most important features affecting the movie general rating.

1 Introduction

Movie rating is a popular topic in machine learning applications. The general rating of a movie indicates how successful the movie is. Machine learning predicts the movie rating given the information of the budget, cast, and so on. There are thousands of actors, actress and directors. The combination of cast is much more. With comprehensive information of movie, the general user rating for it can be predicted based on historical dataset with machine learning techniques. This can be used in the prediction of coming movies' popularities, and supports the casting selection and investor decision.

The Internet Movie Database (IMDb) provides the information about movies, such as the cast, directors, genre, budget, and etc. It is also a platform for user to rate the movie, scaling from one to ten. The average rating of the general public is shown as a decimal on the movie introduction page. Box Office Mojo is a website specializing in the budget and gross of movies. The data used in this paper is crawled from IMDb and the-numer.com.

Various machine learning methods are tried out on the data, including k-nearest neighbors, decision trees, support vector machine, random forest classifier, bagging, and boosting. The authors compare these machine learning methodologies in movie rating and present the optimization procedures to increase the accuracy of the predictor.

2 Literature Review

Most papers related with movie rating prediction focus on the user customized rating, among which, most of them analyzed the review of users. Pearson compared the random forest and support vector machine's performance on user based rating. He found out that to some extends, random forest is a better predictor than support vector machine. Marovic made experiments on both user and movie's data, with content-based methods, and collaborative methods. He concluded that the probabilistic latent semantic analysis outperforms all others. Here, we mainly review the paper that also rates the movie's general rating.

Armstrong and Yoon proposed Kernel Regression and Model Tree to predict the mean rating. Kernel Regression defined the distance with the mixture of nominal and numerical attributes and predicted the rating with weighted k-nearest neighbors. Model tree performed the decision tree first, and applied linear regression on the leave nodes. The error rate of the kernel regression is 14.11%. The error rate of model tree is 14.40%. However, the data includes the Box Office Gross. The prediction result is highly related with the Box Office Gross, but, by the time the Box Office Gross is available, the rating should also be available.

Ensemble methods can greatly improve the performance without invent new learning algorithms. There are successful trials on the variations of bagging. Yu extended Adaboost to weighted bagging and tested on the real data. It measured the importance of sampling using likelihood or error. Horthorn and Lausen combined linear discriminant analysis and classification tree. This so called double-bagging algorithm suits the data with small training samples, because it gets rid of test set.

Absorbing the papers above, we tested all the base classifiers mentioned in these papers, and created some variation of bagging in our proposed methods.

3 Data

3.1 Data Collection

The first step was to collect data. We tried to acquire data from IMDb and the-numbers.com by doing web-crawling work. Thanks to the Python framework BeautifulSoup, we obtained over 20,000(including duplicates) and 5000 movies' budget and box office information respectively. It's a pity that IMDb uses some special techniques to prevent people from scrapping box office and budget data from their website therefore we have to search for another source, which is the-numbers.com, to find corresponding budgets and box offices.

From IMDB, what I acquired from each movie are: movie title, TV series, starting year, genre, directors, writers, actors, description, duration, number of votes via www.imdb.com/random/title. From the-numbers.com, what I acquired from each movie are: movie title, year, month, production budget, domestic gross, worldwide gross.

Among all the features we obtained, number of votes, domestic gross, worldwide gross are actually "posterior" features, which means those variables could be deemed as another perspective as rating. Therefore, we firstly tried not to use such variables.

3.2 Data Preprocessing

The data includes title, rating, genre, directors, actors, budget, duration, year, month, and number of voting users. The categorical data, such as genres, is processed and stored as binary values. For example, if a movie falls into Comedy and Drama, the values are set to one for these two dimensions. The value are set to zero for all other genre dimensions for this movie. The first five directors and actors of the movie listed in the IMDb are included in the data dimension.

The rating in IMDb is a decimal with fractional part. In binary class classification, the two classes are separated by average rating of all samples. The movie, whose rating is larger than the average rating, is classified as class one. Otherwise, it is classified as class zero. In multi-class classification, the movie is classified into eight classes, from three to ten. The closest integer of the rating is the class the movie belongs to. There is no movie in our sample pool belonging to class one and two. Therefore, we only have eight classes.

4 Machine Learning Techniques

4.1 General Approaches

The accuracy of SVM is not as good as others. The data has very high dimensionality. The number of features is much larger than the number of samples available. It is easier to find

the hyper-plane to linearly separate the two classes. However, the classification might not be appropriate for general cases. This is the reason that it does not fit well in the validation set.

Here is a sum-up for the performance of each classifier for binary classification.

Table 1: General approaches for different classifiers

bagging	SVM	adaboost	decision tree	forest	gradient boost	KNN
0.6521	0.5838	0.6956	0.6273	0.6397	0.6770	0.6645

Decision tree performs better than SVM. Decision tree's decision boundary is parallel to axis. It is more suitable for categorical data. Our data has a lot of categorical data, which benefits the decision tree method. Decision tree is a weak learner in general, so the accuracy is nor much higher than SVM.

Random forest combines bagging and random selection of features based on decision tree. It selects features that have stronger correlation with the rating. Because the number of features is very large, especially compared to the sample size, it is hard to select features that truly reflect the correlation with the ratings. So the random forest performs a bit better than decision tree, but still not very good.

K-nearest neighbor works the best in the basic classifiers. It uses Euclidean distance function. Although the dimension is huge, the categorical values are small. The budget value is the major contribution to the distance. Since the budget is highly related with the rating, the k-nearest neighbor classifier works well in the movie rating data.

Bagging, adaboost and gradient boost work well on the data. All of them take decision tree as base classifier. Decision tree is a weak learner, which is lightly better than guessing. And the data is diverse. Therefore, the bagging and boosting all improve the accuracy. Gradient boost is also boosting, but with the least square as its error function. Bagging only reduce the variance, while, boosting reduce both variance and bias. Therefore, boosting works better than bagging.

In general, ensemble methods work best among all tried methodologies. SVM is not a good fit when dimension is much more larger than the number of features.

4.2 New Bagging Algorithm

4.2.1 Weighted Bagging for Binary Class

Besides tradition approaches, we also tried to invented something new by observing the characteristic of our data. We realized two things in our previous experiment. 1) The misclassification points in our general classifiers are distributed randomly. 2) All the base classifiers we used are generally weak learners, none of which plays a good performance that could reach above 70% accuracy.

Below is one picture that I plotted to imply that the misclassification points are randomly distributed and therefore it reminds me perhaps I could combine them to a new package like bagging. You can see from the picture that most of the points are below 4, which means most of the points could be voted correctly if we can combine them together.

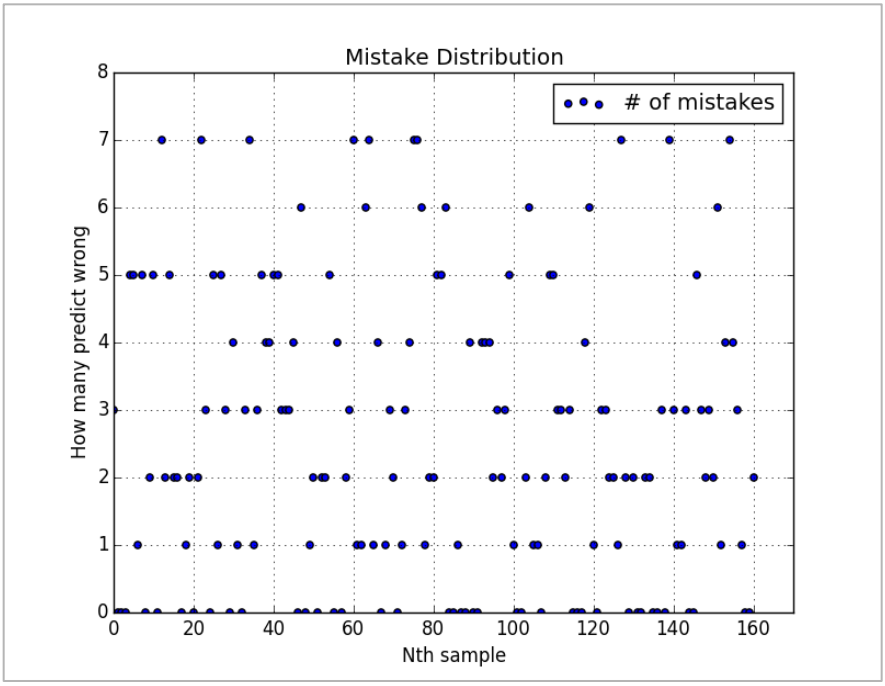

Figure 1: Number of Mistakes for each sample

Based on such situation, I start to think of bagging those base classifier together by writing a big bagging such that:

$$y^M bag(x) = \frac{1}{M}\sum_{m=1}^{M} y_m(x)$$

However, such model has a problem that if we have two base classifiers. For one data point, one estimates 1 while the other estimates 0 and it turns out the average will be 0.5 and hard to tell which classifier is better at this point. In this case, we have to always ensure we have odd number of base classifiers to vote.

Therefore, we come up another idea that we use the accuracy of each base classifier represent one's weight.

$$y^M bag(x) = \frac{\sum_{m=1}^{M} a_m \cdot y_m(x)}{\sum_m a_m}$$

This function applies the idea of softmax into a binary bagging problem. We wish the most accurate classifier has the largest weight vote for the final decision. Moreover, I wrote the code to let the whole "voting" process do over and over again so that we could have three advantages: 1) The low-correction-rate classifier's weight will become inevitably smaller and smaller. 2) Avoid the fluctuation that occasionally good classifier scores low. 3) Don't have to set odd number classifiers to vote. I plotted a picture to see the weight changes during 10 iterations for 7 different base classifiers.

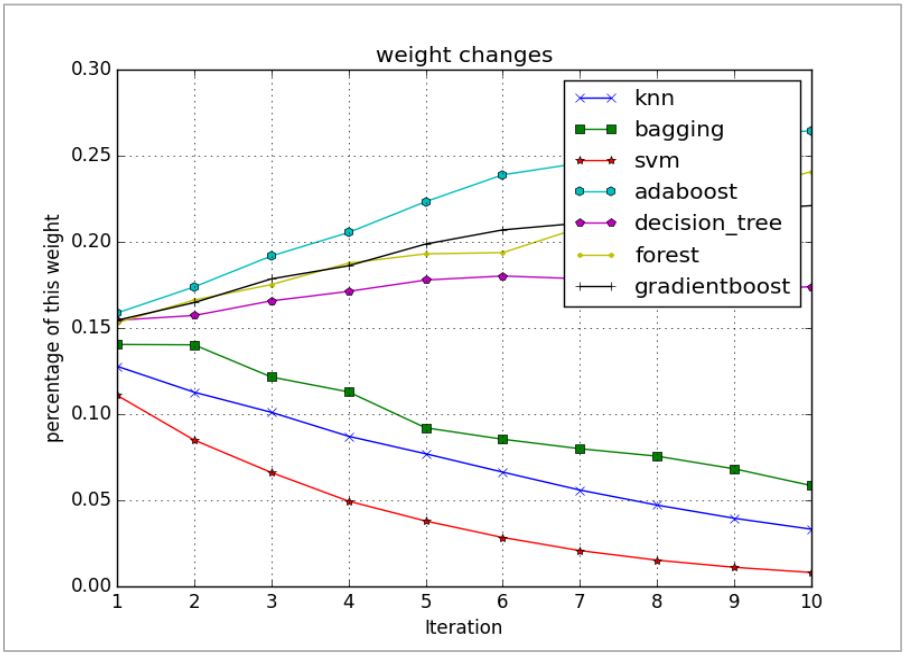

Figure 2: New Weights for base classifiers after 10 rounds

You can see from above. These good classifiers, e.g. forest, keep going up and its weight get bigger and bigger.

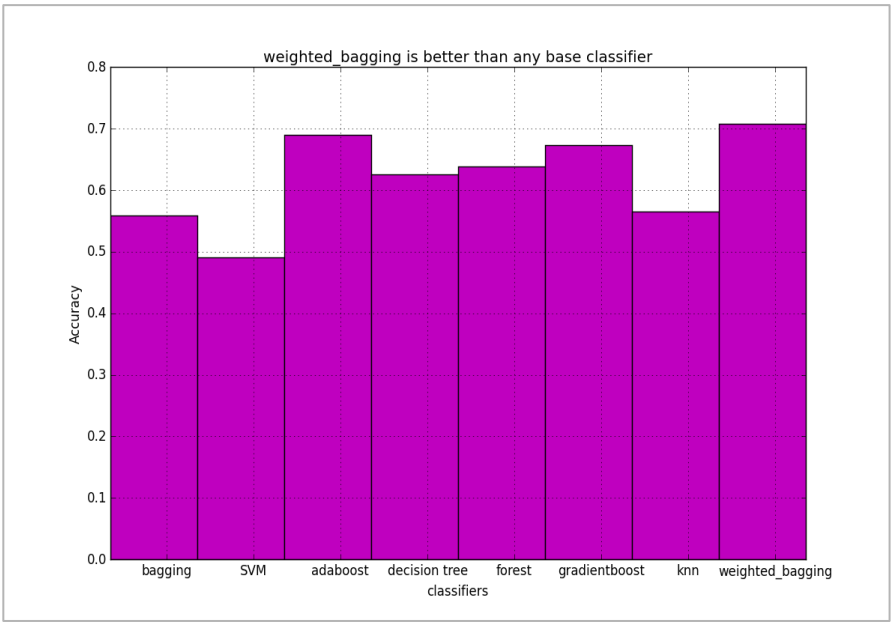

Figure 3: Comparison among all classifiers and weighted bagging

This weighted-bagging trick is very useful to our problem because you can see that weighted-bagging plays better performance than any base classifier. What's more, that is definitely a breakthrough. For the first time, we reach over 70%.

4.2.2 BFS-Bagging

Even though weighted bagging methods can effectively reduce the influence of the few weakest learners but it doesn't ensure to improve the whole accuracy when base classifier's accuracy is similar to each other. Moreover, even some of the base classifiers are weaker than others, they still vote during the process. Ideally, we wish some blind-classifiers, who seems play better than guessing but eventually vote for wrong part in some key points, could be kicked out from the base classifier pool. For example, we have four classifiers(A,B,C,D) whose accuracy are 0.65, 0.6, 0.55, 0.55 and they are going to predict five point in validation set as 0, 1, 0, 0, 0.

0, 1, 0, 0, 1

A: 1, 1, 1, 0, 1 3/5

B: 0, 1, 0, 1, 0 3/5

C: 0, 1, 0, 0, 0 3/5

D: 1, 1, 1, 1, 0 2/5

Vote: 1, 1, 1, 0, 0 2/5

If we simply vote for every decision according to their prior accuracy, you will see the final result will be even lower than any one of the base classifier. However, if we could remove the fourth classifier D, you will see the result reaches 4/5. Therefore, I'm thinking perhaps removing one or more classifier of the whole base classifier pool might have positive influence to the whole weighted-bagging. The algorithm is as following steps:

1) Calculate each base classifier's accuracy and use it as weight to predict. Set the max equals to the predict accuracy.
2) Kick out the lowest training accuracy classifier and predict.
3) If the predict is better than max, update it.
4) If not, move to next lowest classifier and predict and go back to 2).
5) If have kick out each classifier once already, based on the best combination so far, continuing kicking another classifier until such conditions that, after this round kicking, the max remains the same value as the last round.

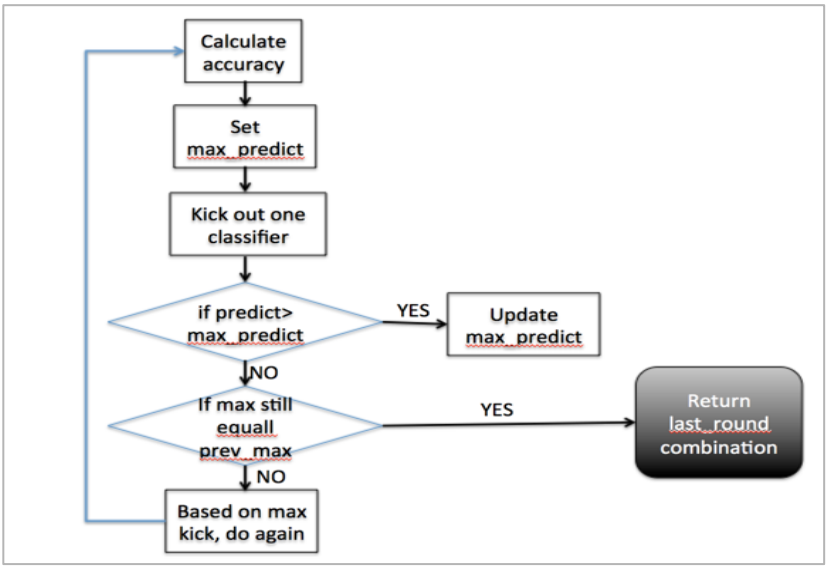

Figure 4: Process of BFS weighted Bagging

In this way, the algorithm would find the best combination of several base classifiers by kicking out one or more useless classifiers. The whole process of kicking out classifier applies the logic of BFS. We try to iterate all the combination of base classifiers to make sure which one is the best.

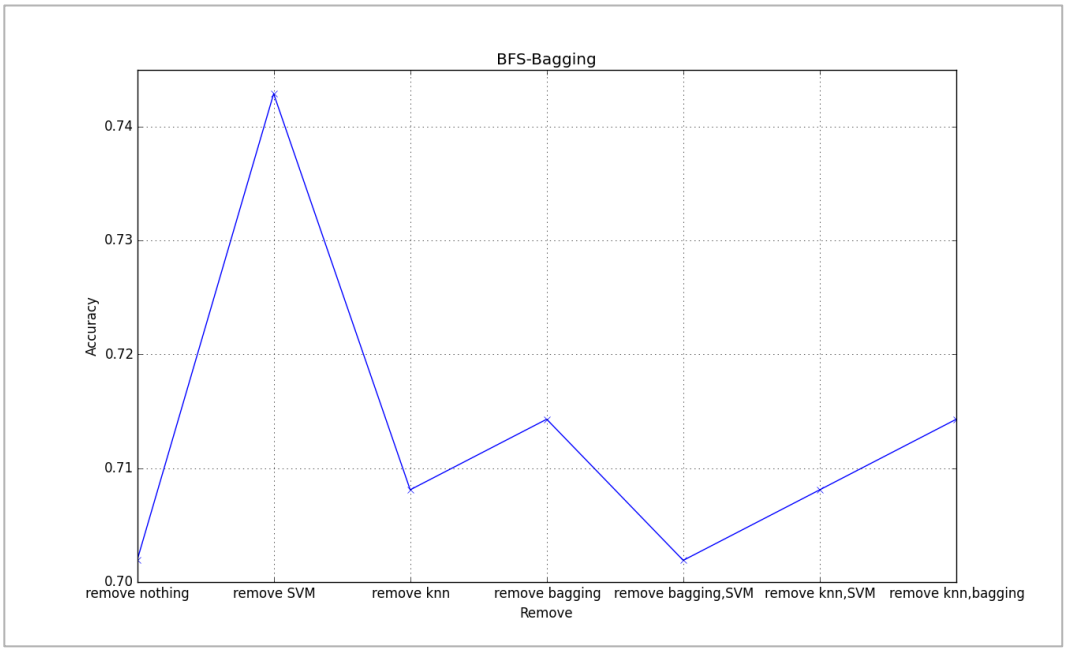

Figure 5: Performance Demonstration During BFS Weighted Bagging

In this experiment, I saved the process of removing all other base classifier but only try removing worst three: SVM, bagging and KNN. It turns out removing SVM reaches the highest accuracy among all and removing two classifiers doesn't make it better than removing SVM. Therefore, the combination of all classifiers other than SVM is the best choice after BFS-Bagging.

4.2.3 Multi-Class

Besides binary classification, we also tried multi-class classification. We divided the class into 10 class because the rating ranges from 3 to 10(the least rating movie is 2.8, round to 3) and we applied the weighted-bagging method used in last problem to multi-classification as well.

Table 2: General Approaches of Different Classifiers for Multi-class

bagging	**SVM**	**adaboost**	**decision tree**	**forest**	**gradient boost**	**KNN**	**weighted-bagging**
0.3354	0.3291	0.3291	0.3354	0.4534	0.4099	0.2670	0.3105

Apparently, the forest is the best among all, even better than the weighted-bagging. I think there are several reasons: 1) Decision tree is very friendly to categorical data and forest is an algorithm determined by several decision tree. 2) Forest decides each point by the votes from each decision tree and each decision are built differently. Since we have a large scale of categorical data, specifically binary data, the forest can be precisely divided into two groups

at each node, unlike continuous data.

4.3 Chi-Square Test

Since we have more than 4000 features in our data, we are supposed to consider about dimensionality reduction. However, most of the features are actors or directors that only appear once. In such case, using chi-square test is a good way to examine the contribution of each feature. Chi-square is concerned about the frequencies of each feature and the larger chi-square is, the more influential features have for the rating.

$$\lambda^2(C, E) = \sum_{i=A}^{Z} \frac{(C_i - E_i)^2}{E_i}$$

where C_A is the count (not the probability) of feature A, and E_A is the expected count of feature A.

After we tried calculating the chi-square of our features, interestingly, we found the correlation ranking quite make sense.

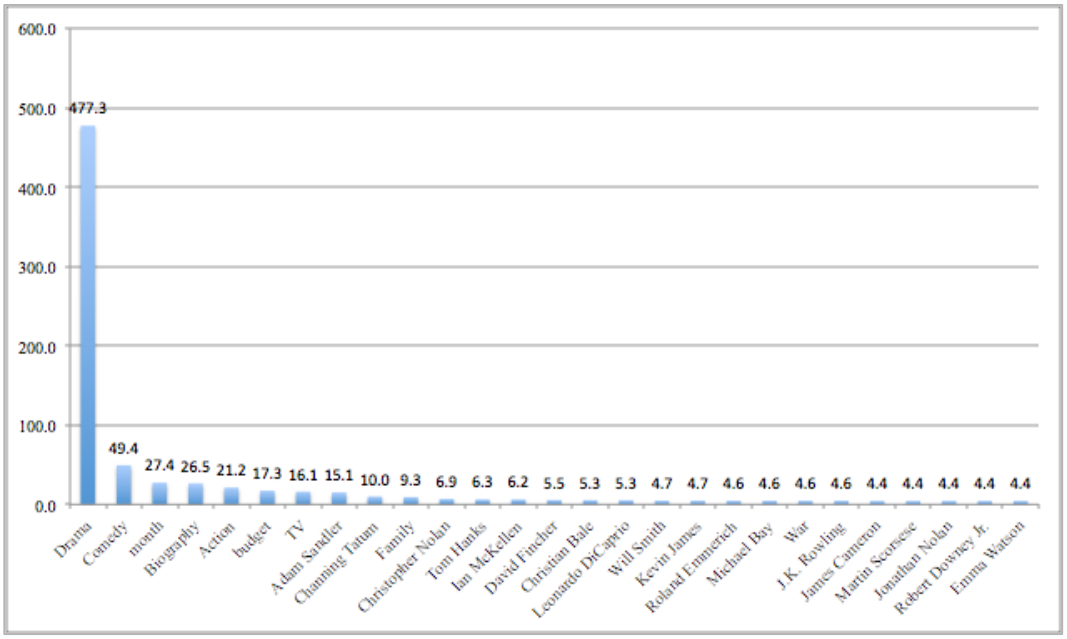

Figure 5: Chi-Square Test Top 28 Features

You can see clearly from above. Two types of genre e.g. Drama, Comedy are the most significant features for evaluation. Also, you can see the month of the film released and if the film is movie or not are also very important. Famous director Christopher Nolan, James Cameron as well as famous actors Adam Sandler, Leonardo Dicarprio, Emma Watson are also ranking top.

Table 3: Top 5 Actors, Directors, and Genres

Top-Five Directors	Top-Five Actors	Top-Five Genres
Christopher Nolan	Adam Sandler	Drama
David Fincher	Tom Hanks	Comedy
James Cameron	Ian Mckellen	Biography
Quentin Tarantino	Christian Bale	Action
Steven Spielberg	Leonardo DiCaprio	Family

4.4 Adding Posterior Information

Like we mentioned before, number of votes, domestic gross, worldwide gross box office are actually another way of rating. If a movie is really a hit, we have reason to believe more people would love to vote on IMDB and people are likely to watch this film in the cinema. We admit that audiences do not equally appreciate many well-sold movies and many movies are voted because of its popularity not because people love it. Yet, in a general view, those posterior information is still strongly linked with ratings. Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

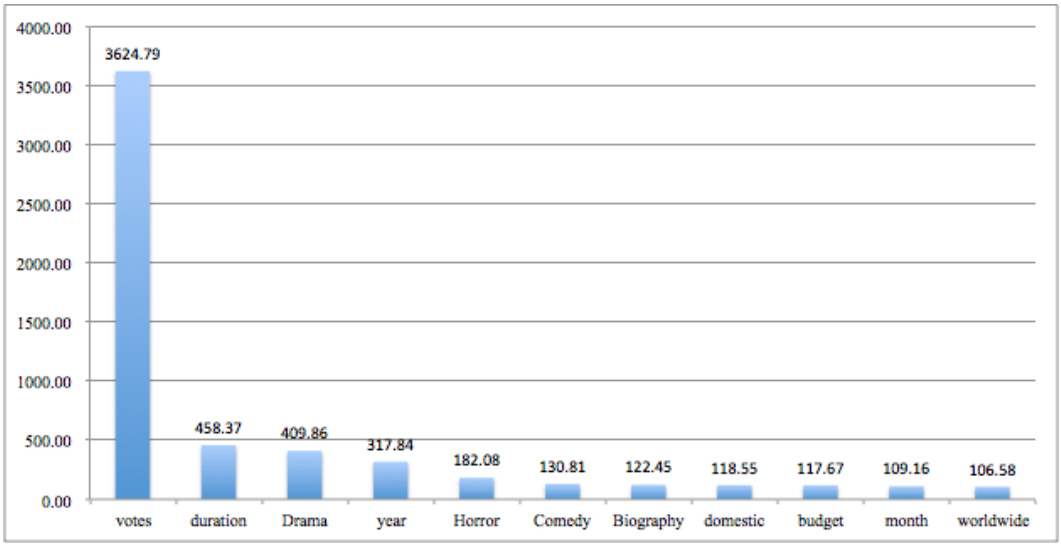

Figure 7: The Chi-Square Test after Adding Posterior Information

Picture above is the chi-square test after adding votes, domestic gross and worldwide gross. These three features contributes a lot to the final results, especially votes. Adding new features, the accuracy of validation set rises significantly.

Table 4: Performance of Each Classifier after Adding Posterior Information

bagging	SVM	adaboost	decision tree	forest	gradient boost	KNN	weighted-bagging
0.6737	0.5041	0.8090	0.7556	0.7876	0.8339	0.6156	0.8291

Obviously, the accuracy of almost every base classifier is promoted after adding posterior features. However, the weighted-bagging methods is not better than the rest of the base classifiers any more but still higher than average.

5 Conclusion

Among general approaches, except SVM, all other applicable methods could reach over 60% accuracy. Forest and decision tree are proven to be very useful for categorical data and meanwhile boosting and bagging also work great.

Combining those base classifiers by weighted-bagging is very effective. It promotes the accuracy to around 70% and avoids using odd number for voting. Furthermore, consecutively updates base classifiers weights are proven to be a good trick for improving accuracy. BFS Weighted Bagging is a useful method to find the best combination of several different base classifiers and it promotes the accuracy to around 75%. It takes longer time but it does help.

Given more posterior information, the accuracy of predicting movie's quality could be above 80% and however this is actually a kind of cheating. That Future prediction is determined by future phenomenon shall not be deemed as a kind of prediction. Chi-square is a friendly way for categorical data to examine its influence to the final results.

6 References

[1] Marovic, M., Mihokovic, M., Miksa, M., Pribil, S. & Tus, A. (2011) Automatic movie ratings prediction using machine learning. MIPRO, 2011 Proceedings of the 34th International Convention.

[2] Persson, K. (2015). Predicting Movie Rating: A comparative study on random forests and support vector machines. University of Skovde.

[3] Armstrong, N. & Yoon, K. (1995) Movie Rating Prediction. Carnegie Mellon University.

[4] Yu, Q (2011) Weighted bagging: a modification of AdaBoost from the perspective of importance sampling. *Journal of Applied Statistics,* Volume 38, Issue 3, Pages 451-463.

[5] Horthorn, T. & Lausen, B.(2002) Bagging Combined Classifiers. In Jajuga, Krzystof, Sokolowski & Andrzej (eds.), *Classification, Clustering, and Data Analysis 7*, pp. 177-184. Springer Berlin Heidelberg.

[6] Horthorn, T. & Lausen, B.(2003) Double-bagging: combining classi"ers by bootstrap aggregation. *Pattern Recognition* 36, pp. 1303 – 1309