Dynamic Rating Adjustment : A BoxOffice Prediction System

Biao Huang, Haochen Wang, Weiyang Wen, and Zhiyuan Wang*

December 8, 2018

Contents

*{bh1918, hw1985, ww1351, zw1946}@nyu.edu, Names are in alphabetical order.

1 Business Understanding

Projection and analysis of box office revenue are essential for the movie industry. In addition to its financial implications, such prediction and analysis could impact the decision-making process of filmmakers. In the pre-release stage, filmmakers could make cast selection and budget estimation accordingly through revenue projection. It also has implications for subsequent business decisions and operations. For instance, it makes perfect sense for a movie with high revenue projection to adopt aggressive marketing, overseas releases, or producing movie-related by-products; those that are less widely acclaimed, however, will likely occur losses doing so. As for theater chains, film revenue projection can be helpful as well. As a manager of a theater, for example, you could arrange bookings out of your best interests by taking the box-office projection into consideration.

For our task, we want to identify available predictors before release, by which we would utilize and make box-office projections. Box office revenue is used as the target variable while the rest features are considered independent variables. It is anticipated that features like actors and directors are significant since star actors can attract loyal viewers, and we resorted to NLP-related technics to quantify impacts of celebrities. Moreover, such extracted information would uncover obscured insight that can provide guidance in the filmmaking process that helps narrowing down top rosters so that filmmakers can easily shortlist their candidates. Our model is designed to help companies to maximize business success and reduce uncalled-for losses at least at any stages of the movie production cycle.

2 Data Understanding

The dataset that will support data mining to address our business problem comes from the database of IMDB website. We obtained it in two ways. First, the majority of the data came from its official API. In addition, we used several web scrapping techniques to collect some complementary data from other websites counting statistical records of movies, such as 'The Numbers', and 'Box Office Mojo'. In total, we gathered more than 15500 movie records. With such large number of data, we hope to build a model to predict the box office for incoming movies in which we can make businesses based on this powerful model.

3 Data Preparation

This part details the conventions used for the cleaning and checking of potentially suspicious or out-of-range values on variables of this dataset. The cleaning took place in 2 stages: a preliminary stage when raw data were

obtained from the API and then a final cleaning stage. This part outlines the procedures used at both stages of data cleaning.

3.1 Preliminary Cleaning

After we obtaining the data by requesting the IMDB API, a set of procedures was followed to complete a preliminary cleaning of the file.

- First, we recode all numeric variables, such as *Runtime*, *Budgets*, and *BoxOffice*, from string to int or float. For the target variable *BoxOffice* specifically, we need to remove the dollar sign and commas before casting it to float. For variables *imdbRating*, *Internet_Movie_Database*, *Rotten_Tomatoes*, *Metacritic* involving ratings but using different scales, we need to transform them to a scale of 10 to 1.

- In addition, we truncate several variables to make them simpler. For variable *Actor*, we select the first four actors who are the most famous ones to represent each movie. For movies with less than four actors, we use *'nan'* to fill the blank. Similarly, for variables *Director* and *Writer*, we only choose its top value for each movie.

- Last, we apply the method of feature engineering by adding and removing variables. Based on our domain knowledge and common sense, we believe the number of premiere countries should have a great impact on the target variable. Therefore, we create a new variable *Country_count* by counting the number of premiere countries of each movie. Also, we replacing the variable *Genre* with 24 dummy variables representing different types of movie. Since we are only interested in movies, instances such as *TV series*, *documentary*, and other categories should be removed from the data set. The final results are in Table 1.

Table 1: Data Attributes

Variable	Type	Description
BoxOffice	float	Box office of the movie
Director_Score	float	Feature engineered score of director
Writer_Score	float	Feature engineered score of writer
Actor_Score	float	Feature engineered score of actors/actresses
Director	string	Name of the director
Runtime	int	Runtime of the movie
Title	string	Name of the movie
Writer	string	Name of the write
Year	int	Released year
imdbVotes	int	Number of votes
imdbRating	float	Rating from IMDB
Internet_Movie_Database	float	Rating from Internet Movie Database
Rotten_Tomatoes	float	Rating from Rotten Tomatoes
Metacritic	float	Rating from Metacritic
Actor	int	Name of actors
Country	string	Names of premiere countries
Country_count	int	Number of premiere countries
Adjusted_Budgets	float	Movie budget
Genre	factor	Type of the movie (Action, Adventure, etc.)

3.2 Final Cleaning & Verification in Detail

While the Preliminary Cleaning stage caught many cleaning-related issues, there were still issues that either were not caught by the cleaning programs or were somehow missed. Thus, we reexamined each variable once we got the dataset. The following changes were given particular attention: 1) Variables with missing values; 2) Transforming several variables; Each of these issues is addressed in more detail below.

3.2.1 Handling Missing Values

For the whole dataset, there are only three variables with missing values, which are *BoxOffice*, *Budget*, and *Rating*. We applied two methods to fill up the missingness. For the target variable *BoxOffice* and variable *Budget*, we use another dataset obtained from the IMDB Pro. On the other hand, we impute the mean to the missing values of *Rating*.

3.2.2 Transforming Variables

It is clear that several variables are greatly influenced by year. For instance, the inflation of each year should have a great impact on variables such as *BoxOffice* and *Budget*. To make them more comparable, we used the inflation rate of each year to adjust these two variables for each movie. After adjusting, we divided them by 1,000,000 and

then applied the log transformations to them to make the relationship more clear. Also, the variable *imdbVotes* should be adjusted based on year, since the number of Internet users has a strong positive relationship with year. To make it more accurate, we fit a simple linear regression model to adjust the vote for each movie.

3.3 Data Visualization

After cleaning the data, we visualize these data using bar graphs, box plots, and scatter plots to check the relationship between target variable and other potential important features. The results are following.

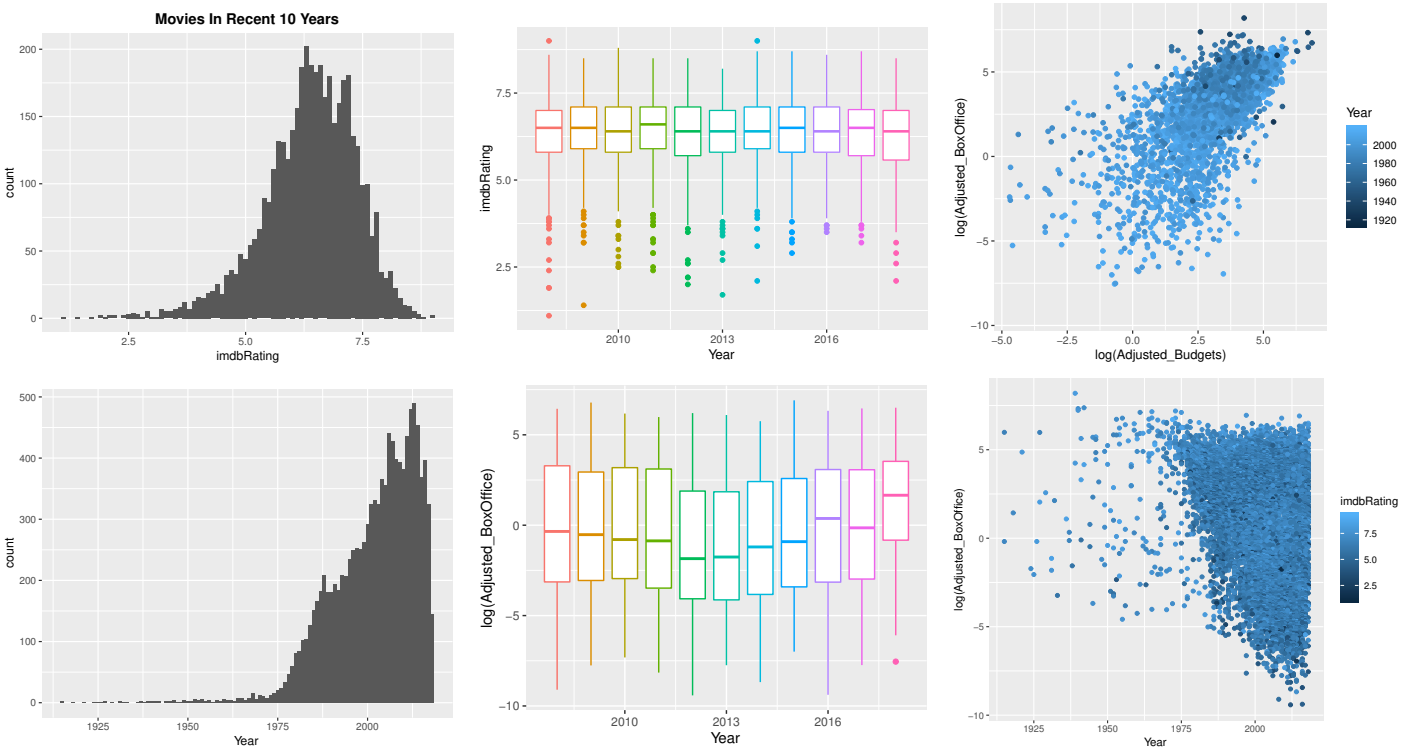

Figure 1: Data Visualization

In addition, we group movies into several groups based on their box offices to check the distribution of our target variable.

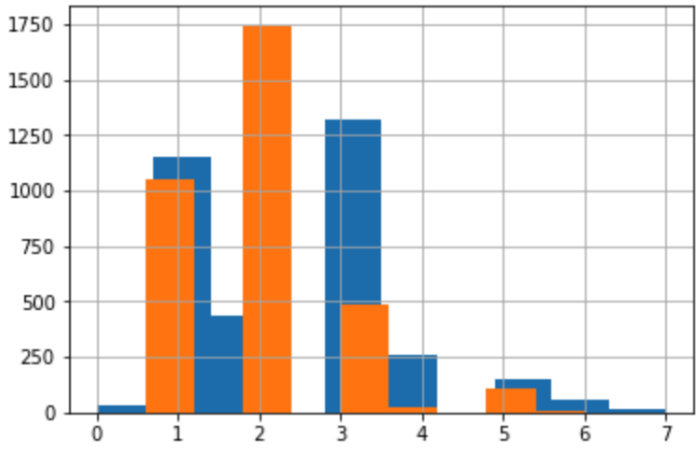

Figure 2: BoxOffice in Categories

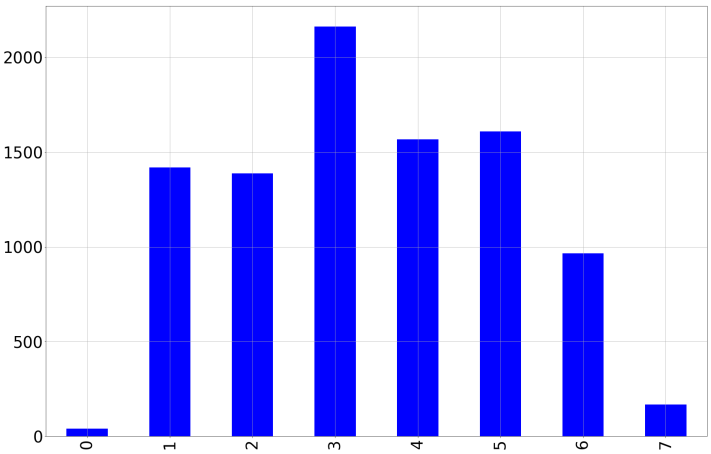

Figure 3: BoxOffice in Categories VS. Budgets

4 Feature Engineering

Our data set contains movie ratings from various sources, including IMDB, Rotten Tomatoes, Metacritic and Internet Movie Database, which are considered standardized, informative metrics we would like to take advantage of. However, these ratings will not be available before the movie comes out, and thus cannot contribute to our model immediately. To make good use of those information and improve our model performance, we propose a way of feature engineering that incorporates those features. We will borrow ideas from term frequency and document frequency to adjust the engineered features, and will leverage smoothing techniques to make our features robust against sparse entries.

4.1 Weight-Adjusted Bag of Names

For casts of each movie, we construct three dictionaries, representing Director, Writer and Actor/Actress respectively. While constructing these dictionaries, we assign different weights to each role while counting according to their relative importance and appearance position. For instance, the first two actors/actresses in the cast table will receive more weights comparing to the rest of actors/actresses. Intuitively speaking, if we regard a movie as a corpus, then the appearance of primary actors and actresses is a lot more than other casts, and hence adopted such weight adjustment system.

4.2 Rating Decomposition

Per similar rational above, we use our weight system to decompose the ratings. To be more explicit, let R denotes the average rating of one film, W_D, W_W, W_A, $W_{A'}$ denote the weights we assigned for director, writer, first two actors/actresses and the rest actors/actresses, then the decomposed rating for say director of a certain film i

would be:

$$R^i_{\text{name}} = \log(R^i) + R^i * \frac{W_D}{W_D + W_W + 2 * W_A + 2 * W_{A'}}.$$

4.3 Smoothed Weighted Average

Some of the names in the casts table might be rare, and thus their decomposed ratings can be unstable and unreliable due to low appearance frequency. To alleviate situations like this, we resort to general Laplace smoothing while computing the weighted-averaged decomposed ratings. More explicitly, we adopt the following weight smoothing:

$$W_{\text{name}} = \frac{1 + \alpha}{\#\text{ of weighted name appearance} + \alpha * \#\text{ of total movies}},$$

where $\alpha = 0.005$ is our smoothing factor.

And now, our weighted average rating for some director can be obtained via:

$$R_{\text{name}} = W_{\text{name}} \sum_i R^i_{\text{name}}.$$

4.4 Frequency Adjustment

Next, for each of the ratings we obtained, we adjust it by a "confidence" factor that measures how frequent we would see some person: the more we see the person, the higher confidence factor we would assign him/her, since we believe his/her rating would be much more credible. Mathematically, we give each person adjusted score based on his/her role, ratings and frequency of appearances:

$$R^{Adj}_{\text{name}} = R_{\text{name}} * \frac{1}{\log(\frac{\#\text{ of total movies}}{\#\text{ of name appearance}})}.$$

4.5 Discussions

Here are some highlights of our engineered features:

- Give proxies of the moving rating by only looking at cast level information, which is available long before movie's debut, and thus can be used to make box office predictions without worrying about data leakage.
- Provide detailed rating system for different roles (director, writer, actor, actress) based on their historical ratings to give more objective, exhaustive overview of a movie.

- Leverage weight adjustment, rating decomposition, Laplace smoothing, and frequency adjustment to construct (somewhat) unbiased features.

There are also limitations of our approach:

- Our choices of weights as well as smoothing factor are somewhat arbitrary, and thus might not be the most reasonable way in terms of adjustment.

- The data set we obtain might be biased in the first place, which would make our adjusted ratings biased even further. One possible treatment would be adopting inverse frequency adjustment instead of our current frequency adjustment. However, such treatment would yield counter intuitive scores after manually comparing a few representative "names" in our rating system, which we consider a bigger issue.

5 Modeling & Evaluation

5.1 Choices of Algorithms

We formulate the task at hand as a classification problem as exact predictions are rarely needed. Box office values are grouped into 8 bins according to sizes. The response variable is thus transformed into a categorical one with 8 possible values, where value 0 denoting smallest box office that range in thousands while value 7 denoting super blockbuster movies that takes in billion. Two measures are taken to reduce the problem of unbalance data set. Firstly, log transformation is applied to reduce skewness of box office. In addition, cut-off points are chosen such that the number of movies in each bin is approximately the same. We use accuracy as an evaluation metric. Accuracy works best if false positives and false negatives have similar cost, and we believe this is the case for box movie prediction.

In terms of evaluation framework, we do 5 fold cross-validation on training set and use grid search to identify the best parameters of each one. As an example, the below graph shows the accuracy on validation-set against tuning parameters (tree depth and number of trees) in this case. From the graph, we can see that accuracy increase at first, and start to decrease at some point due to over-fitting. We identify the highest point as the optimal model and use it to generate testing result.

Five different classification models are tested in our study. Below we detail the reasons why we choose these models, their pros and cons, and the parameters we tuned for each of them.

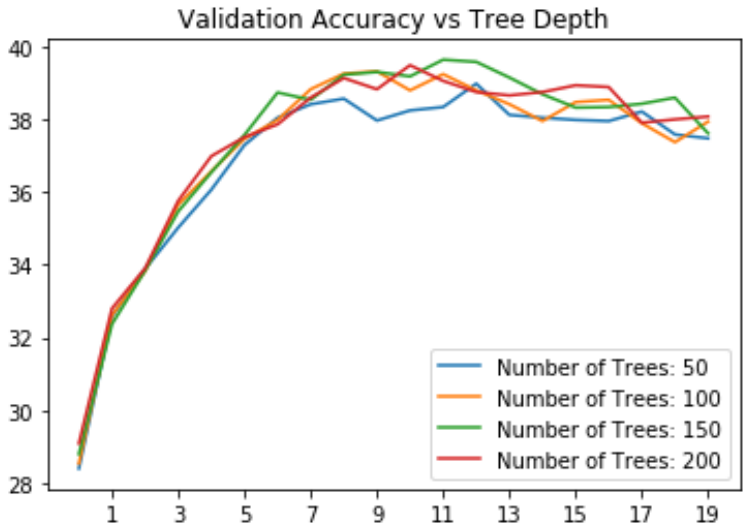

Figure 4: Validation Accuracy VS. Tree Depth

- **Logistic Regression** Logistic regression is commonly the first choice for classification problem. It is easy to interpret and robust to noise. More importantly, it is more informative than other classification algorithms. Like traditional linear regression, it shows the direction of association and how important each feature is through its coefficient. The disadvantage is that it suffers multicollinearity. Regularization method and regulation strength are tuned for optimal performance.

- **Support Vector Machine** SVM is another commonly used classification problem. SVM is chosen because we like its ability to model complex, nonlinear relationships and it does not suffer multicollinearity. However the downside is that the result is hard to interpret. We use the one-against-one approach in *sklearn* for multi-class classification. Here we try different kernels to locate the best settings via SVM.

- **K-Nearest Neighbors** kNN is among the simplest learning algorithms in classification. It only approximates locally and thus it is fast and highly salable. It also naturally handles multiclass classification. As in all kNN problems, the challenge is to identify the optimal K in our model.

- **Random Forests** Random forests is an ensemble learning method. It is an aggregation over multiple small and simple decision trees and count the votes of them. Random Forests has the advantage of high accuracy and low variance as compared to other classification models, and it also runs efficiently on large data sets. For this method we experiment on the number of trees and tree-depth.

- **Neural Networks** Neural Networks is a powerful tool in machine learning. It is the most accurate classifier if trained appropriately, but the downside is the lack of interpretability. We use *Keras* with

tensorflow backend in this setting. A simple 3-layer fully connected network is used to avoid over-fitting and reduce training cost. Features are standardized for this setting. We make use early stopping and tune the patience parameter, which is the number of epochs with no improvement to stop training.

For base line model, we also use a logistic regression with all the above-mentioned features expect the NLP scores we generated. Cross-validation and parameter tuning are also conducted with the same procedure above. We also construct an **upper-bound model**, which is a logistic regression with all features above plus audience ratings, i.e: $imdbRating$, $imdbVotes$, $Rotten\_Tomatoes$ and $Metacritic$. Audience ratings are highly predictive as they are direct reflection of audiences sentiments. Of course this model is not attainable as audience rating is not available before movie is release, but the performance of this model can serves a good upper benchmark for our other models.

5.2 Result & Visualization

Table 2: Model Comparison

Model	Feature Set	Parameters	Training Acc.	Testing Acc.
LR (Baseline)	Desc	P=l2, C=1.0	31.66	31.64
LR (Upper_bound)	Desc + Ratings	P=l1, C=10.0	41.26	42.33
LR	Desc + NLP	P=l1, C=10.0	36.61	36.54
SVM	Desc + NLP	Ker=linear	36.89	37.18
kNN	Desc + NLP	K=40	31.69	31.03
RF	Desc + NLP	N=150, depth=12	39.63	40.61
NN	Desc + NLP	patience = 12	42.68	37.47

We use LR to stand for logistic regression, RF to stand for random forests, NN stands for neural networks. We also group our features into categories. $Runtime$, $Genre$, $Country\_count$ are grouped into $Desc$ (descriptive features), $Director\_Score$, $Writer\_Score$ and $Actor\_Score$ and grouped into NLP , and $imdbRating$, $imdbVotes$, $Rotten\_Tomatoes$ and $Metacritic$ are grouped into Ratings. Accuracy is shown in percentage.

As shown above, random forest achieved 40.61% testing accuracy and is highest among our tested models. The neural network model also performs decently with highest in-sample accuracy and second to the best out-of-sample performance. Most of our models have significant improvements compared to benchmark, and random forest is even comparable to upper-bound model. So we choose random forest as our final model to continue with the analysis.

In more details, we visualize our models performance in different categories. The result is shown in Figure 5. With regard to accuracy, our model seems to perform better in small and medium size movies, but less so for movies with larger box office. In terms of recall, our model does well in mid-large class (3 and 5) movies, which means it has strong identifying power of the movies belong in this group. F1 score shows similar pattern as

recall.

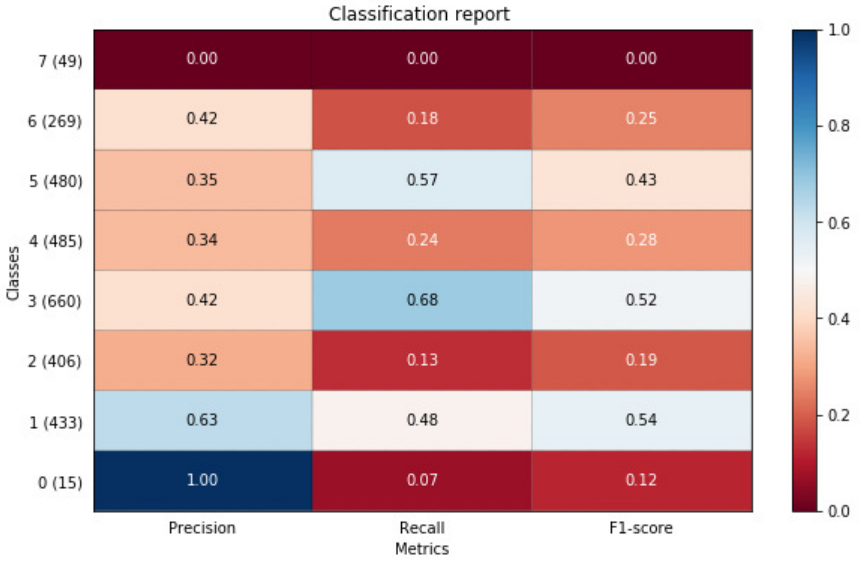

Figure 5: Classification Report

Further model improvement is possible. For example, Budget is an important feature. Intuitively, budget serves as a good reference point to box office range as it is uncommon for a high budget film to get very low box office and vice versa. Unfortunately, budget information is not always publicly available, so we are unable to test it in full scale. Just to confirm our hypothesis, we did a simple regression on a small subset of movies which we could get budget information. The R^2 increase significantly from 25% to 50%. It is very likely that our model could be further improved with budget information.

5.3 Solving the Business Problem

Box office is hard to predict in nature. In upper-bound model, even with information leakage we can only attain 42.33% accuracy, which is a 10.57% increase compared with baseline. Our model uses only pre-release information and successfully attained comparable performance with this model. With budget at hand, the accuracy could be further improved. In other words, it gives users the capability to **predict box office as accurately as if one could get mass audience feedback in advance**. This is the among the best performance in the relevant studies we know so far.

6 Deployment

Our model provides a tool for people in film industry to optimize their business decision. For movie makers, our model could help them to identify the best combination of writers, directors and actors in order to achieve

expected budget range. For theaters, our model could help them to identify better movies and arranging show schedule in order to maximize profit. Since our model is fast and salable, it should be easily deployable in any applications.

In actual production, it is always advised to constantly monitor and update the model. For example, writer scores, director scores and actor scores should be updated frequently in order to keep up with the latest trends. Long term scores and short-term trends may be established separately, as part of the future work, in seeking for better model performance. It is also sensible to add in market-specific features. For instance, there exists a restriction in Chinese film market such that foreign film's show time is not allowed to exceed certain limit. Moreover, most countries do have audiences level guidance where movies are classified into different audience level. Factors like these would have significant impacts and implications toward box office revenues, and thus shall be incorporated into local market models as enhancements to the existing model suite.

Ethically, our model is unlikely to cause problem as no personal data is used. The main risk of our model is the accuracy loss due to pattern shift. For example, audience's taste might shift from Sci-Fi movie to action movies. There is also possibility that talented new actors will outperform superstars and our model fails to capture that the first time. The way to mitigate this risk is again to re-train the model regularly to capture the new information timely. Users should also incorporate subjective judgments in their decision making processes.

7 Acknowledgement

We would like to extend our sincere gratitude to professor Brian d'Alessandro who offered thoughtful comments, extensive related resources, and much guidance on our work. We would like to express our appreciation for Leslie Huang who provides in-depth explanation on some of the related techniques in our project.

References

[1] Guijia He and Soowon Lee. Multi-model or single model? a study of movie box-office revenue prediction. In *IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 321–325, 2015.

[2] David A. Field. Laplacian smoothing and delaunay triangulations. *International Journal for Numerical Methods in Biomedical Engineering*, 4(6):709–712, 2010.

[3] Anantha M. Prasad, Louis R. Iverson, and Andy Liaw. Newer classification and regression tree techniques: Bagging and random forests for ecological prediction. *Ecosystems*, 9(2):181–199, 2006.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. 4:3104–3112, 2014.

[5] Wu Lei, Steven C H Hoi, and Yu Nenghai. Semantics-preserving bag-of-words models and applications. *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, 19(7):1908, 2010.

[6] Xiang Zhang, Junbo Zhao, and Yann Lecun. Character-level convolutional networks for text classification. pages 649–657, 2015.

[7] Hongning Wang, Yue Lu, and Chengxiang Zhai. Latent aspect rating analysis on review text data: A rating regression approach. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 783–792, 2010.

[8] Altmann Andr, Tolo?I Laura, Sander Oliver, and Lengauer Thomas. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340, 2010.

[9] Krushikanth R. Apala, Merin Jose, Supreme Motnam, C. C. Chan, Kathy J. Liszka, and Federico De Gregorio. Prediction of movies box office performance using social media. In *Ieee/acm International Conference on Advances in Social Networks Analysis and Mining*, pages 1209–1214, 2013.

[10] Randy A. Nelson and Robert Glotfelty. Movie stars and box office revenues: an empirical analysis. *Journal of Cultural Economics*, 36(2):141–166, 2012.

Appendix

Contributions

We formulate the problem and work on the report together. The other works are divided as follows:

- *Weiyang*: Model selection, cross-validation, parameter tuning and result visualization
- *Biao*: Data retrieval, data cleaning and verification
- *Zhiyuan*: Weight adjustment, feature engineering and hyper-parameter exploration
- *Haochen*: Web scrapping and data visualization

Data Source

[1] IMDb. *The Internet Movie Database.* http://www.imdb.com/, 2007.

Source Code

All of our code is made available open source at Github.

https://github.com/Distancs/DSGA_1001_Final