Box Office Predictor Project

1st Yaqian Chen
117010032

2nd Bingyan Hu
117010090

3rd Zheyuan Zhou
117010423

4th Zhewen Lyu
117010194

***Abstract*—The movie industry is vital in this generation, and there appears box office predictor to forecast the revenue. The predicted movie gross can be further used to optimize the movie schedule arrangement and provide popularity advice for movie producing. Existing box office predictors have certain limits, the most obvious three are: neglecting training set for the model; relatively few attributes; no real-time sentiment-based prediction. Therefore, our group refined the current predictor and constructed a relatively accurate predictor taking data set with selected attributes, real-time sentiment data as input and output the total gross of the target movie in North America on the website our group built. The implementation will be illustrated in this paper. The overview of the process includes: scraping data from website; cleaning, processing, and formatting data set; applying machine learning model to select the best performing algorithm; testing twenty percent of the data; constructing user-friendly website; inputting the target movie information and outputting the result on the website. In the future, this model can be continuously promoted by using the predicted data as part of the training data and enhancing the accuracy of the parameters.**

***Index Terms*—predictor, machine learning, box office, movies, sentiment**

I. Introduction

The movie industry is booming in this generation. According to McClintock, global box office revenue amounted to 41 billion U.S. dollars in 2018 [1]. Under this situation, there appears the box office predictor trying to forecast the movie gross. The box office predictor can be applied in the real movie industry. For the cinema owner, it can promote the movie schedule arrangement according to the predicated movie gross to optimize their revenue. For movie producers, the prediction data gives an idea of what kind of movie will be popular and provides the direction for movie producing.

II. Literature Review

Certain work has been done in the box office predictor field, however, there exist drawbacks. To illustrate them, our group lists two representative papers. The first paper Dynamic Box Office Forecasting Based on Microblog Data [2], which provides a relatively accurate model for prediction. However, it is not convincing since it does not have a test case, only applies different algorithms to the training set and selects the best performance one as a result. Moreover, the dynamic predication it claims is predicting once a week, which is not useful in reality considering the typical movie duration is only about three weeks.

Another study on the box office prediction utilized the sentiment analysis [3], however, they selected too few attributes into the prediction model, which decreases the accuracy; on the other side, it can only predict whether the movie will be a hit or not but neglects the specific data. Therefore, our group refines the current movie box office prediction algorithm in the following ways: firstly, adding more related attributes to generate a better model for prediction; secondly, using sentiment analysis to produce a real-time prediction system; thirdly, constructing a user-friendly website to visualize the prediction.

III. Structure

The overview process of our box office predictor is indicated as Fig 1: The first step was data collection through scraping the Internet websites with tool libraries like BeautifulSoup and requests, which generated the history of raw data in .csv form. Then by applying NLP model, our group accomplished the data processing, which included data insight, data pre-processing and data formatting, and selected the attributes according to the correlation matrix. This step generated the training data set. The third step lied on applying training algorithms with Machine Learning Model and got the training result data to determine the best performing algorithm, packed it into a Model. At the same time, prepared the data set of ten movies with the determined attributes for prediction. Moreover, using the textBlob tool to scrape comments of the directors, actors, and movies, analyzed the comment to generate real-time sentiment attribute and input it into the predictor model. And finally, output the predicted result on the website we constructed. To sum up, our group's box office predictor model takes the data set of the target movie with the selected attributes and real-time sentiment data set as inputs, output the dynamic total gross of the target movie in North America.

IV. Data Mining

A. Basic Information of Movies

We found a movie dataset on the Kaggle database. It includes 5043 movies from 1916 to 2016 as well as their corresponding information. Each movie has 28 attributes: color, director name, the name of main actor, the name of second actor, the name of third actor, number of critics for reviews, duration, director Facebook likes, main actor's Facebook likes, second actor's Facebook likes, third actor's Facebook likes, genres, movie title, number of votes users, cast total Facebook likes, face number in poster, plot keywords, movie IMDB link,

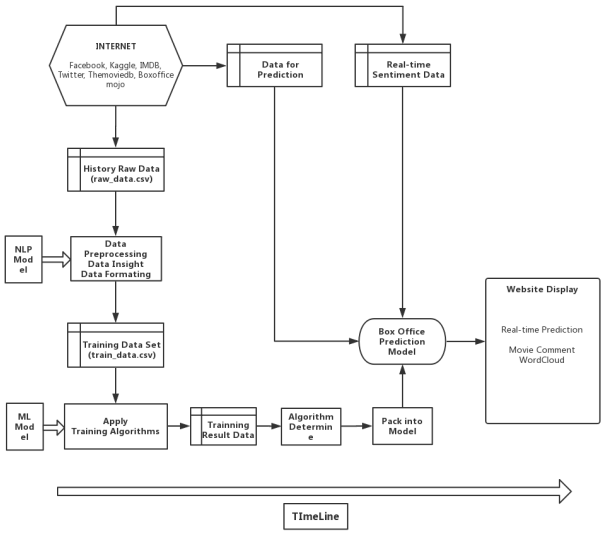

Fig. 1. Project Structre

number of users for review, language, country, content rating, budget, year, IMDB score, aspect ratio, movie Facebook likes and gross.

B. Sentiments

We applied natural language processing (NLP) in our prediction model by sentiment analysis of comments. After crawled comments from related websites, we generated a set of sentiment polarities by TextBlob, ranges from -1 to 1, where -1 indicates the most negative sentiment and 1 indicates the most positive sentiment [1]. We crawled two types of comments from Twitter and IMDB respectively.

1) Comments on Actors and Directors: We crawled tweets from Twitter to obtain comments on actors and directors of movies. To implement Twitter crawling, we modified a Twitter crawler on GitHub [4]. We crawled tweets by setting the query of Twitter search API as the actor or director name and a fixed period right before the release date of that movie. Considering that many movies in that dataset are so old that Twitter had even not developed when they were released, we modified the crawling mechanism. For movies that were released before 2006, where Twitter was founded, we crawled current tweets instead of crawling comments tweeted around the release date. As for each set of comments crawled, there are 9 attributes: tweet ID, account name, user ID, time, text, links, replies, retweets and favourites. A separate .csv file was generated to store comments on each director or actor. Each tweet text will give a specific sentiment polarity by using TextBlob. Then, the mean value of sentiment polarities for all tweet texts in one file will be regarded as the sentiment polarity of the corresponding director or actor.

2) Comments on Movies: We crawled reviews from IMDB website to obtain comments on movies themselves. We wrote the crawler by ourselves and crawled reviews for each movie in the dataset. A separate .csv file was generated to store reviews of each movie. Each review will give a specific sentiment polarity by using TextBlob. Then, the mean value of sentiment polarities for all reviews in one file will be regarded as the sentiment polarity of the corresponding movie.

C. Data Insights

First, by observing the frequency distribution of each attribute in Fig 2, we filtered out some inappropriate attributes in the movie dataset. There are many old movies in this dataset, which means many actors and directors may be no longer active on present social media. Many of them do not even have a Facebook account. Besides, some directors and actors' Facebook user names are different from their real names, which makes it difficult to crawl corresponding "likes" data from Facebook. In consequence, some of them have few or no corresponding Facebook likes. Thus, considering the extremely imbalanced frequency distribution of these attributes, it is unreasonable to include them in our prediction model. We also filtered out attributes that show little relevancy to gross, such as face number in poster and keywords.

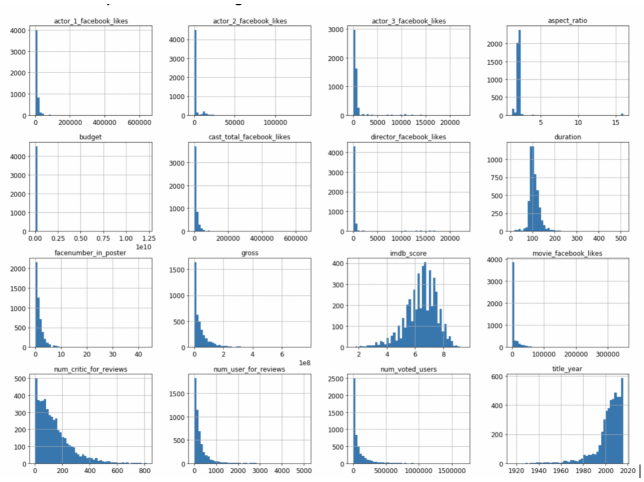

Fig. 2. Frequency Distribution of Each Attribute

For categorical attribute "genre", we sorted counts of genres for these movies. Among all 5043 movies, "Drama", "Comedy" and "Thriller" appear most frequently. Specifically, when we consider the 100 highest-gross movies, "Adventure" shows absolute advantage among all genres. It appears 80 times in total, which means 80% of the most popular movies belong to the "Adventure" genre. This result indicates that the genre of a movie does have an impact on its gross. The more popular a movie's genres are, the higher the possibility for it to achieve high box office. Thus, we would like to include this attribute in our prediction model.

After that, we do correlation analysis among numeric attributes from the movie dataset as well as sentiments we obtained from tweets and IMDB comments. By looking at

correlation scatter diagrams in the scatter matrix, we can tell that most of these attributes are positively correlated to gross.

More specifically, we computed the correlation coefficient for each attribute. Attributes with relatively high correlation coefficients are included in our prediction model. For attributes related to sentiments such as IMDB score, the sentiment of comments on actors and movies, even though their correlation coefficients are low, we decided to consider them since the relationships between them and the gross are not linear.

TABLE I
CORRELATION COEFFICIENT TO EACH ATTRIBUTE TO GROSS

Attribute	Correlation Coefficient
number of voted users	0.637
number of users for reviews	0.559
number of critics	0.480
log(budget)	0.456
IMDB score	0.198
sentiment of comments on actors	0.155
sentiment of comments on movies	0.083

At last, we have nine attributes, country, genres, IMDB score, sentiment of comment, number of voted users, number of users for reviews, number of critics, the sentiment of the leading actor and the log budget, as our initial inputs.

V. DATA PREPROCESSING

A. Fix Missing Records

For each attribute in the data set, there exists missing value data for the following three reasons: firstly, the data set contains the records of the movies that are uncommon and too old, it is hard to obtain the corresponding data like the score of the IMDB website; secondly, the sentiment data for the movie director, actor and comment is hard to obtain, which is understandable; thirdly, network problem, scraping data from the network depends largely on the network and firewall. The missing values certainly infect the correlation value of the corresponding attributes and affect the accuracy of the algorithm. Therefore, although it is impossible to eliminate the error, our group still narrow down the errors for the sentiment data. The specific method is firstly dropping all NaN records if all the sentiment data for director, actor, and movie comment is lost since this kind of data is hard to fix; secondly, fill in the blanks with average values since the distribute of the sentiment data is generally Normal Distribution.

B. Features Scaling

Considering the algorithms our group selected are all Machine Learning algorithms, which do not perform ideally when the input numerical attributes had very different scales. However, this is exactly one of the features for the attributes our group selected, for example, the scale of the budget is over million, the scale of the sentiment is even less than one. The solutions are listed as the following: firstly, for the data like budget, take the log of the data with base ten; secondly, use the standardization formula:

$$x' = \frac{x - \bar{x}}{\sigma} \tag{1}$$

These two solutions are practical since, on the one side, the purpose of this project values more on the order of the magnitudes; on the other hand, the distribution of the selected attributes are similar to a normal distribution and the standardization won't ruin the essential information in the data.

C. Categorical Attribute

Handling numerical attributes is relatively intuitive, however, there also exists non-numeric data like the produced country of the movie and the genres of the movie. This kind of data is stored in the categorical format, and it is difficult to compute its mean or median value. One possible solution our group applied is transforming the categorical data into vector data by using *One-Hot* encoding system to extend the attributes. The principle of *One-Hot encoding* system refers to splitting the column which contains numerical categorical data to many columns depending on the number of categories present in that column. It creates a binary attribute for every category and extends the total number of attributes. In our case, we have 21 types in attribute genres and 66 types in attribute country. After the encode of *One Hot*, the total required number of input attributes rises to 94.

VI. MODEL CONSTRUCTION

It should be noted that in our predictor, we focus on the magnitude of the gross instead of the exact value of the gross. Therefore, we took the log of the gross with the base of 10 as our target values.

In this section, we used four models to predict the final gross and test the performance: Linear regression, Elastic net regulation, decision tree, and random forest. Here are the brief introductions of these four models, we will choose one of them to implement our predictor later. Table II is the mathematical notations we used in the report.

TABLE II
MATHEMATICAL NOTATIONS

Notations	Mathematical Meanings
y	ground truth vector
$\hat{y}$	predicted values vector
θ	model's parameter vector
$\mathbf{x}$	instance's feature vector
$\mathbf{X}$	the matrix of sample instances vectors
h_θ	hypothesis function, using the model parameters θ
L	loss function

A. Metrics of Model Performance

Since our predictor is similar to a regression work, we apply root mean squared error as our metric to weight the performance of the model.

$$RMSE(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(\mathbf{X}^{(i)}) - y^{(i)})^2} \tag{2}$$

B. Linear Regression

In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The formula of linear regression model is shown below:

$$\hat{y} = h_\theta(\mathbf{x}) = \theta^T \cdot \mathbf{x} \tag{3}$$

C. Elastic Net

Elastic Net is a middle ground between Ridge Regression and Lasso Regression. The regularization term is a simple mix of both Ridge and Lasso's regularization terms, and we can control the mix ratio r. In general, to avoid overfitting, we prefer using a regularized regression to fit the training data. In general, Elastic Net is preferred over LASSO since LASSO may behave erratically when the number of features is greater than the number of training instances or when several features are strongly correlated. Therefore, we consider Elastic Net as our algorithm candidates.

$$L(\theta) = MSE(\theta) + ra\sum_{i=1}^{n}|\theta_i| + \frac{1-r}{2}a\sum_{i=1}^{n}\theta_i^2 \tag{4}$$

D. Decision Trees

Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks and even multioutput tasks. They are very powerful algorithms, capable of fitting complex datasets. Here we apply the function offered by Scikit-Learn, whcich uses *Classification And Regression Tree* (CART) algorithm to train Decision Trees. Also, Decision Tree is better at handling categorical features than those general linear regressions. Fig 3 shows the basic idea of how to use Decision Tree to make regression.

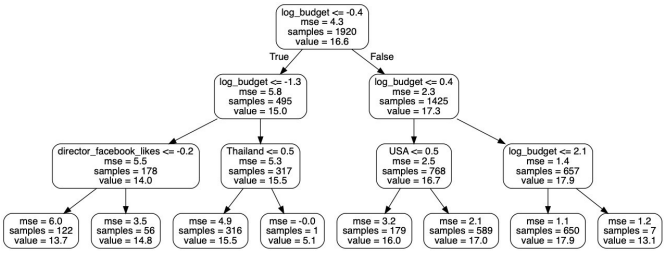

Fig. 3. Decision Trees Regression

E. Random Forest

A Random Forest [5] is an ensemble of Decision Trees, generally trained via the bagging method. The ensemble algorithm is to aggregate the predictions of each predictor and predict the class that gets the most votes. The number of features that can be split on at each node is limited to some percentage of the total. This ensures that the ensemble model does not rely too heavily on any individual feature, and makes fair use of all potentially predictive features. Each tree draws a random sample from the original dataset when generating its splits, adding a further element of randomness that prevents overfitting. For the most cases, this voting classifier often achieves higher accuracy than the best classifier in the ensemble. In other words, it is an implementation of the wisdom of crowds on machine learning.

We used cross-validation to test these four algorithm in our dataset and found that Random Forest has the best performance among them. The RMSE scores of each algorithm are shown in Fig 4.

```
Scores: [1.56082944 1.33944302 1.50119286 1.68907076 1.60007733]
Mean: 1.5381226803964938
Standard deviation: 0.11659907239336736
**********************************************


Scores: [1.60538499 1.33254878 1.51222832 1.66702916 1.67358391]
Mean: 1.5581550282498853
Standard deviation: 0.12681917473099186
**********************************************


Scores: [1.89011701 1.74747242 1.83847177 1.96066354 1.87959012]
Mean: 1.8632629743544815
Standard deviation: 0.0699952009023076
**********************************************


Scores: [1.4812408  1.25130531 1.29355842 1.48496149 1.49592173]
Mean: 1.401397552080257
Standard deviation: 0.10625408611536691
**********************************************
```

Fig. 4. Model Selection Test

Finally, we choose Random Forest as our model to build the predictor. By using the grid search, we fine-tuned a set of best hyperparameters to fit our dataset. The detials of the hyperparameters are shown in III.

TABLE III
HYPERPARAMETERS OF THE BEST ESTIMATOR

Hyperparameters	Values
bootstrap	True
max depth	None
max features	10
max samples	None
min samples leaf	1
min samples split	2
n estimators	100

The RMSE score of test set is 0.592, which keeps at a good level.

VII. IMPLEMENTATION

The implementation part will be divided into two sections: data flow and website development.

A. Data Flow

We firstly crawled data for ten on playing movies including their budgets, countries, genres, IMDB scores, comment sentiments, log budgets, numbers of voted users, numbers of voted users, numbers of critics and produced years. The budget information is scraped from *themovieDB* website while other data are from IMDB. The detailed data are shown in Fig 5. Then the data will be sent into the application program with

movie_title	budget	country	genres	imdb_score	sentiment_comment	log_budget	number_of_voted_user	numb
Last Christmas	30000000.00	USA	Comedy	6.6	0.251901918	17.21670794	12440	348
Playmobil: The Movie	75000000.00	France	Animation	4.5	0.152087048	18.13299867	1043	33
A Beautiful Day in the Neighborhood	4000000.00	China	Biography	7.9	0.251703758	15.20180492	6299	160
21 Bridges	33000000.00	USA	Action	6.6	0.209004584	17.31201812	4223	114
Playing with Fire	41143082.00	USA	Comedy	4.6	0.111109971	17.53256635	1731	74
Midway	100000000.00	USA	Action	6.9	0.237651219	18.42068074	14485	410
Joker	55000000.00	USA	Crime	8.7	0.203816406	17.82284374	505653	8117
Frozen 2	742060467.00	USA	Animation	7.3	0.160096735	20.42494129	27022	535
Knives Out	40000000.00	USA	Comedy	8.1	0.144948188	17.50439001	31495	485
Ford v Ferrari	97600000.00	France	Action	8.3	0.236837384	18.39638805	50231	569

Fig. 5. Data Input for Prediction

the prediction model generated from the program and stored in a .pkl file in the last step to generate the prediction result Fig 6.

We also use the callback function in the application program to send the prediction result to dash and display the latest results in the application interface every 5 seconds. The prediction result file will also be updated by adding the new actor sentiments. The actor sentiment is evaluated by crawling comments about the actors from twitter and use NLP to measure the sentiment value. The value will be continuously added to the training set. We again use the random forest to generate a new prediction model stored in .pkl file and update the prediction data.

	prediction	movie_title
1	7.352766454263631	Last Christmas
2	5.708181611713724	Playmobil: The Movie
3	6.295521056716576	A Beautiful Day in the Neighborhood
4	6.9903203876601365	21 Bridges
5	6.766498565046346	Playing with Fire
6	7.515933833872441	Midway
7	7.989887920156149	Joker
8	7.665749613626631	Frozen 2
9	7.46924346871305	Knives Out
10	7.107208246259148	Ford v Ferrari

Fig. 6. Prediction Result

B. Functionality

By clicking the option, the user can send the signal to the program. After receiving the signal, the program is able to fetch the corresponding prediction results and sent it to the application to display it on the interface. The line char figure for box office prediction will also automatically be flushed every 5 seconds since the program will automatically update the prediction results every 5 seconds by a callback function. The interface is shown below.

There are two ways to trigger the figure for box office prediction. Firstly, after each five-second time interval, the graph will be automatically flushed with the updated prediction. The second is when the user changes the predicting movie from drop-down options, the application will send the corresponding movie value to the back end and the program will new the figure on the interface. While comment word cloud and comment treemap will only be flushed when the

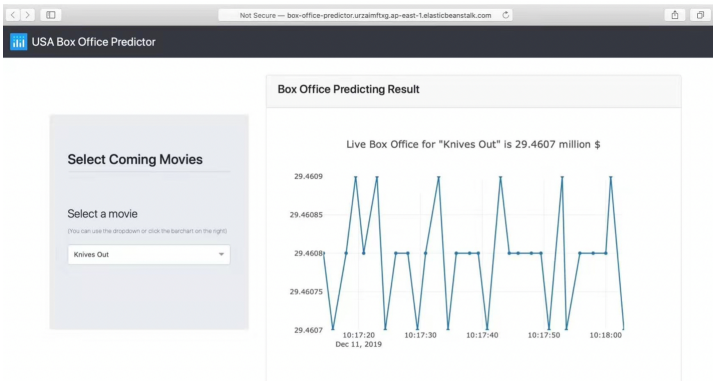

Fig. 7. Real Time Prediction

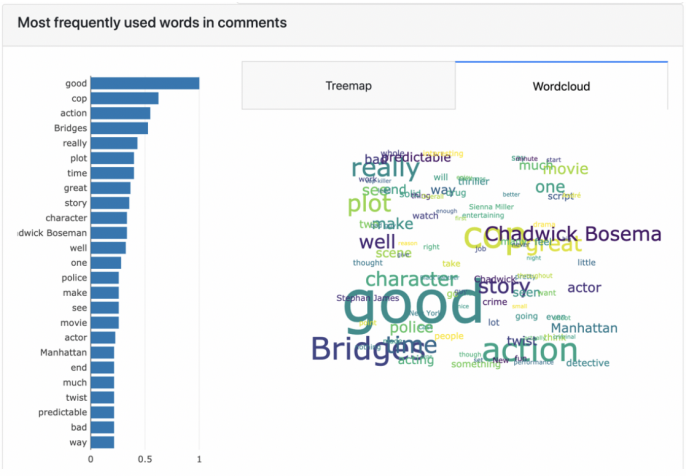

Fig. 8. Word Cloud of Sentiment

user changes the predicting film. The whole callback map is shown in Fig 9.

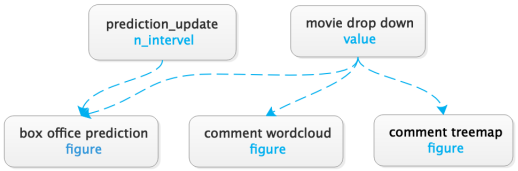

Fig. 9. Callback Map of Website

We deployed the application with the Elastic Beanstalk on AWS which is short for Amazon Web Services. Amazon Web Services (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis.

C. Predictions on Recent Films

We put the prediction data for ten on playing movies on December 4th and got real revenue for seven of them by December 10th. We take the log of the real revenue and the predicted revenue and the result is shown below. We can prove that our model generally predicts the order magnitude for each movie.

As we can see from 10, box office prediction for Last Christmas and Midway have relatively good performance while prediction for Joker and Frozen2 did worse than others. There are mainly three reasons for this phenomenon: only crawled the latest actor sentiment, did not consider the company information and did not consider the movie series.

First, when we were building the training set, we only crawled the leading actor and directors' sentiments from recent tweets. However, the factor affects box office revenue should be the sentiments about the directors or actors around the file release years. Many directors might be quite famous recently while it was just a freshman while directing the file in the dataset. Some directors and actors may be no longer as famous as they were before, while the sentiments for some film directors and actors at that time are not available. Since there are a great number of old movies in our dataset with the oldest one released in 1916, in a time when there were no social media, we can't find comments for every actors and director around the film release years. This is the problem caused by our dataset and we can try to fix it by using other parameters like total box offices for a specific director or actor up to that film release year.

The second possible reason for prediction error is that we haven't included the production company into our dataset. The two movies with relatively high prediction error: Frozen2 and Joker are both films with great produce companies (Disney and DC) while having relatively unpopular genres (animation and crime). They are both extreme cases for their genres. However, their great production companies make their box office revenue reasonable. We can try to solve it by adding produce company into our consideration in order to improve the prediction model.

The third reason is that we haven't considered the movie series. If the predicting movie is a sequel to a movie series. Their box offices will largely depend on their previous films. The two movies (Frozen and Joker) also belong to this category. In our future improvement, we can also take special treatment to the series movies, adjusting their box office prediction based on their previous films.

VIII. Conclusion

To conclude, the whole process for this project is divided into six steps: data collection, data insight, data preprocessing, data training, application build and feedback adjustment. We firstly do the data collection, crawled 31 attributes with 5043. Then we did the raw data insight to choose the most correlated nine attributes to further applied to our prediction model. Data preprocessing is used to clean the dataset and scale the data. We applied four prediction models which are linear regression, elastic net regularization, decision tree and random forest and finally choose random forest as our prediction model. We then deploy our application on the AWS with beanstalk and will continuously add the new crawled data for the recent movies in our training set to optimize our prediction model. Our project, based on machine learning and sentiment analysis, is able to predict the ranking and order magnitude with relatively high accuracy. The project offers a viable template of combining machine learning, NLP and web technology, which can provide useful information to the movie industry and movie theater to maximize their profits.

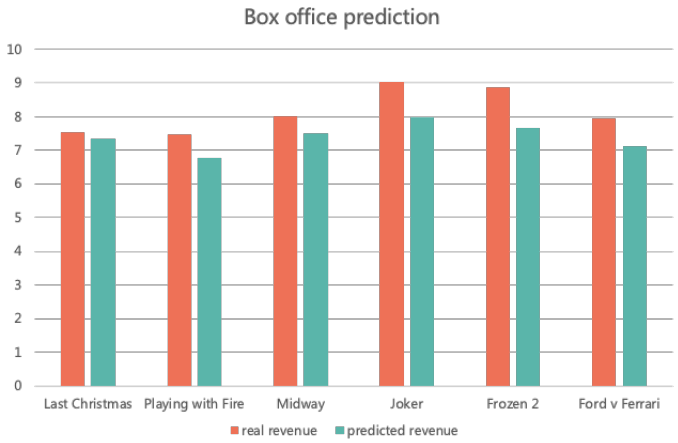

Fig. 10. Comparison with Other Predictions

References

[1] P. McClintock, "Global box office revenue hits record $41b in 2018, fueled by diverse u.s. audiences," Mar 2019. [Online]. Available: https://www.hollywoodreporter.com/news/global-box-office-revenue-hits-record-41b-2018-fueled-by-diverse-us-audiences-1196010

[2] R. Chen, W. Xu, and X. Zhang, "Dynamic box office forecasting based on microblog data," *Filomat*, vol. 30, no. 15, pp. 4111–4124, 2016. [Online]. Available: http://www.jstor.org/stable/24899494

[3] V. Jain, "Prediction of movie success using sentiment analysis of tweets," *International Journal of Soft Computing and Software Engineering [JSCSE]*, pp. 308–313, 10.7321/jscse.v3.n3.46. [Online]. Available: http://dx.doi.org/10.7321/jscse.v3.n3.46

[4] K. Kurita, "twitter-past-crawler," https://github.com/keitakurita/twitter-past-crawler, 2017.

[5] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324