Do Reddit Comments Help Predict Movies Success?

Team 21

Social Data Science

Summer 2019

Abstract

We predict the performance of a movie based on different measures of success using a series of different features available before the release of the movie. The models employed to make the predictions are the Lasso and Elastic net models. One novel feature we include is sentiment scores of Reddit comments created using VADER. We find that including Reddit comments does not contribute to the accuracy of the model.

Character count: 36,509

Contents

1 Introduction **3**

2 Theory **3**

2.1 Machine Learning . . . 3

2.1.1 OLS regression model . . . 4

2.1.2 Linear regression with regularization . . . 4

2.1.2.1 Lasso regression model . . . 5

2.1.2.2 Elastic net . . . 5

2.1.3 Cross-validation . . . 6

2.2 Sentiment Analysis . . . 6

3 Data **7**

3.1 Data Sources . . . 7

3.1.1 IMDb . . . 7

3.1.2 Rotten Tomatoes . . . 7

3.1.3 Box Office Mojo . . . 8

3.1.4 Reddit . . . 8

3.2 Merging Datasets . . . 9

3.3 Ensuring Data Quality . . . 11

3.4 Ethics . . . 12

4 Descriptive Analysis **12**

5 Machine learning **15**

5.1 The models . . . 15

5.2 Results . . . 16

6 Discussion **17**

7 Conclusion **19**

8 References **20**

1 Introduction

One might be interested in predicting the success of any movie for any number of reasons. Knowing how a movie performs before the release could for instance help studios make informed decisions on how much to spend on marketing or how many theaters their movie should air in. Alternatively, movie fans and data geeks might be interested in the challenge of prediction.

In this paper we try to predict the success of films from data available before the release of the film. We measure the success of a film using scores from IMDb, Rotten Tomatoes and box office earnings, which we scrape from online sources. We collect a number of readily available features to predict the success of the movie, such as genre, length, actors, and directors in the movie. In addition to these features, we also use a more unique measure to attempt to gauge expectations regarding the movie: Reddit comments made before the release of the movie. We perform sentiment analysis on these comments to form a measure of the public's expectations of the film and include this as a feature in the model predicting movie success. The idea is that more positive comments should predict a movie performing better.

We find that while we are able to predict the success of the movie to a somewhat reasonable degree of precision, the Reddit comments do not contribute much to the accuracy of the predictions. We conclude that Reddit comments either do not hold much information about the future success of a movie or that one should use more sophisticated tools to analyze the Reddit comments before using them to predict movie success.

2 Theory

2.1 Machine Learning

In the following we will describe different models which we use to predict movie success. We divide our sample into a training and test dataset. We train our model on the training data, and test it on the test data. We define a cost function and minimize this by changing the weights to find optimal weights. The minimization is done by gradient descent, which is an optimization algorithm.[1]

[1]Python Machine Learning, 2nd ed. (2017) by Sebastian Raschka & Vahid Mirjalili, p. 35-37

2.1.1 OLS regression model

When using linear regression, ordinary least squares (OLS) is often the go-to estimator as it is one of the simplest methods. The OLS method minimizes the squared residuals and doesn't do any kind of model variable selection. Therefore, OLS is better suited for causal analysis than prediction in the case of many features. The cost function for OLS is defined as:

$$J(w) = \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2,$$

where $y^{(i)}$ and $\hat{y}^{(i)}$ are the realized and predicted values of the target variable. $\hat{y}^{(i)}$ depends on the weights $w = w_0, w_1, \ldots, w_k$ linearly:

$$\hat{y}^{(i)} = w_0 + w_1 x_{1i} + w_2 x_{2i} + \cdots + w_k x_{ki},$$

where x_{ki} is the i'th observation of feature k. One issue with OLS is overfitting. To combat this, one can use regularization.[2]

2.1.2 Linear regression with regularization

When estimating a model on training data, we risk overfitting the model. Overfitting refers to the model capturing spurious patterns in the training data that don't generalize to the test data. The model will in this case perform well in the training data but very poorly in the test data. This kind of model will have too much variance, and will be too sensitive to the randomness of the training data. The opposite problem is underfitting, which happens when the model has too much bias, and isn't sensitive enough to each observation. This will result in a model, which is consistently wrong. This is illustrated in figure 1[3].

To solve the problem of over- and underfitting, we use regularization. With regularization we add information, i.e. a parameter to penalize complexity. In the case of regularized linear regression, we use the Least Absolute Shrinkage and Selection Operator (Lasso) and Elastic net.[4]

[2]Python Machine Learning, 2nd ed. (2017) by Sebastian Raschka & Vahid Mirjalili, p. 319-320
[3]Social Data Science 2019 at University of Copenhagen, lecture 12.
[4]Python Machine Learning, 2nd ed. (2017) by Sebastian Raschka & Vahid Mirjalili, p. 73-74

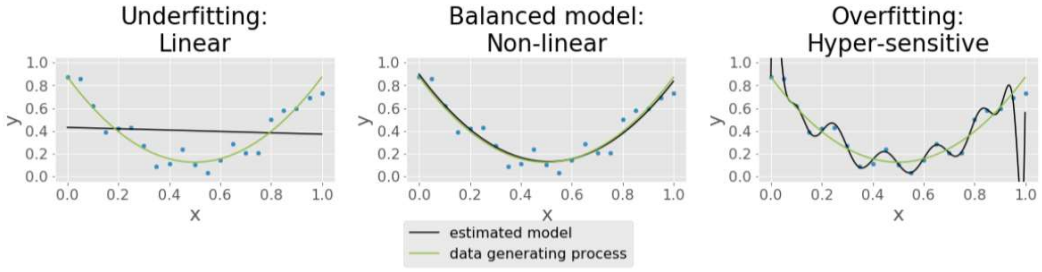

Figure 1: Examples of underfitting and overfitting

2.1.2.1 Lasso regression model

For Lasso regression we add the sum of the absolute weights to the OLS loss function:

$$J(w)_{\text{Lasso}} = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda \sum_{j=1}^{k} |w_j|,$$

where λ is a hyperparameter. A larger value of the hyperparameter λ implies a larger penalty to larger coefficients, i.e. increasing the cost of complexity. Because we are taking the absolute value of the weights the optimum is often an edge solution, meaning that certain weights can become zero. Thus, this method is good when dealing with irrelevant features. Furthermore, we notice that for $\lambda \to 0$, the Lasso regression is equal to the OLS regression, and for $\lambda \to \infty$ our weights will be zero and there will only be an intercept.[5]

2.1.2.2 Elastic net

Lasso is a special case of the more generalized Elastic net method:

$$J(w)_{\text{Elastic net}} = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda_1 \sum_{j=1}^{k} |w_j| + \lambda_2 \sum_{j=1}^{k} w_j^2,$$

where λ_1 and λ_2 are hyperparameters. For $\lambda_1 = 0$ we will end up with the Ridge regression. For $\lambda_2 = 0$ we will end up with the Lasso regression. For $\lambda_1 = \lambda_2 = 0$ we end up with OLS. The idea behind the Ridge is the same as with the Lasso, but since it uses the squared sum of the weights, Ridge is useful for solving the problem of exploding weights. Elastic net is a powerful tool because it captures Lasso and Ridge as special cases.[6]

[5]Python Machine Learning, 2nd ed. (2017) by Sebastian Raschka & Vahid Mirjalili, p. 332-333
[6]Python Machine Learning, 2nd ed. (2017) by Sebastian Raschka & Vahid Mirjalili, p. 332-333

2.1.3 Cross-validation

When training our model the purpose is to fit a model which performs the best on data it hasn't seen before. Therefore we wish to both give the model as much data as possible to train on but also get the most precise evaluations of model performance using as much test data as possible. This is where k-fold cross-validation is a very useful method. Here we randomly divide the training data into k folds. We then train the model on $k-1$ folds and test the performance on the remaining fold. We repeat this k times, such that we have k models and performance estimates. We then calculate the average of the k performance estimates to find an overall performance estimate.

We use k-fold cross-validation to find the optimal hyperparameters. We first select some values for the hyperparameters and then eveluate their performance using k-fold cross validation. We then try other hyperparameters and repeat until we have exhausted a specified parameter space. We then pick the optimal hyperparameters as the ones which lead to the optimal performance estimate.

When we have found the optimal hyperparameters we fit the model to the whole training data and test on the test data which never entered into the k-fold cross validation.[7]

2.2 Sentiment Analysis

In this section we present our approach for performing sentiment analysis on the 380,000 Reddit comments downloaded. For the sentiment analysis we use VADER (Valence Aware Dictionary and sEntiment Reasoner). VADER is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. Therefore, VADER is ideal for our purposes. VADER is open source and available for Python.

VADER takes in a string and outputs four numbers: The positive score, the neutral score, the negative score and the compound score. The three first scores indicate how large a fraction of the string is positive, neutral and negative, respectively. VADER is trained in various ways including to account for consecutive words, such that adjectives etc. can amplify the negative and positive score of words. The compound score summarizes the overall sentiment, with -1 being the most negative and 1 being the most positive.[8]

We use sentiment analysis as follows. For every movie, we have on average

[7] Python Machine Learning, 2nd ed. (2017) by Sebastian Raschka & Vahid Mirjalili, p. 189-195
[8] https://github.com/cjhutto/vaderSentiment

242 comments from Reddit. We calculate the compound score for each of these comments individually. Then, we average the compound score. This yields an overall compound score for the movie as a whole, which we use as our measure of the overall sentiment towards that movie.

3 Data

3.1 Data Sources

3.1.1 IMDb

IMDb (the internet movie database) is an online database of information regarding movies and TV. One key statistic provided in the database is user-generated reviews of all the movies. The database itself is freely available for download as a tab-delimited file on https://datasets.imdbws.com/. As we intend to map performance, the main interest for us is the average rating. We therefore merge the two datasets 'title.ratings.tsv.gz' and 'title.basics.tsv.gz' by an alphanumeric unique identifier of the title provided by IMDb.

We keep the titles, runtime, year of release, and average ratings for all movies. The resulting data has almost 240,000 movies, with the main variables of interest being the title of the movie, the average rating, and the year of release.

3.1.2 Rotten Tomatoes

Rotten Tomatoes is an review-aggregation website for movies and TV. The main features of the website are two types of ratings for every movie on the website. The first rating is the 'Tomatometer'. The Tomatometer score is the percentage of professional critic reviews that are positive for the given movie. A team of curators hired by Rotten Tomatoes determine whether or not a professional review is positive or not. The second rating is the 'audience score'. The audience score is the percentage of users who have rated the given movie positively (at least 3.5 out of 5). Thus, Rotten Tomatoes provides both a measure of critics' and other users' rating of a movie.[9]

We get data from Rotten Tomatoes by scraping the website. First step in the scraping process is to create a list of all the movies on Rotten Tomatoes. This is done by browsing all movies and sorting by release date. Iterating on page numbers, we scrape all 312 pages. This yields 10,000 movies. Rotten Tomatoes informs that there

[9] https://www.rottentomatoes.com/about/

are approximately 20,000 movies in their database, so the 10,000 movies appear to be a hard-coded limit. However, we believe that 10,000 movies with data such as a link to the poster and URL is more than adequate for our needs, as the movies are sorted by release date. The next step is then utilizing the scraped URL's to visit the individual page of each movie and gathering aspects such as Tomatometer, audience scores, actors, and directors.

We end up with a dataframe of almost 10,000 movies, with the main variables of interest being the title of the movie, the Tomatometer, the audience score, and the release date.

3.1.3 Box Office Mojo

Box Office Mojo is a website which tracks box office revenue for movies. We use the worldwide box office revenue as a measure of a movies financial success.

We get data from Box Office Mojo by scraping the website. First step in the scraping process is to create a list of the movies on Box Office Mojo. This is done by visiting the overview of all 129 movie studios on Box Office Mojo. For each of these movie studios, Box Office Mojo has a list of all the studio's released movies, spread out over pages of 100 movies a page. By scraping all these pages, we get box office information of all movies in the Box Office Mojo database.

The result is a dataframe with over 10,000 movies, with the main variables of interest being the title of the movie, the world box office revenue, and the release date.

Box Office Mojo also has information on the actors and directors whose movies have had the highest total box office revenue. We also scrape this data as a measure of the most successful actors and directors. The result is 882 actors and 912 directors. We use this to create a variable indicating how many top actors and directors appear in a movie.

3.1.4 Reddit

Reddit is an online discussion forum. Users can discuss movies in the subreddit (subforum) r/movies. The forum has more than 21 million subscribers and features in the neighborhood of 3,000 and 30,000 daily comments from users.[10] We are interested in the opinions of users regarding different films based on their Reddit comments before the movie was released.

We get data from Reddit by using the Reddit API. To interface with the Reddit

[10] https://subredditstats.com/r/movies

API we use PRAW, the Python Reddit API Wrapper. Using PRAW we search for the 5 most relevant posts in r/movies that show up when searching for the title of a movie. In some cases, only collecting comments from 5 posts seems to not be exhaustive as several blockbusters have more than 100 relevant Reddit posts about them. However, for the case of lesser-known movies we have found that some movies might only have 1 or 2 relevant posts. To minimize the amount of noise from irrelevant posts, we choose to scrape only 5 posts per movie. The relevance is chosen by the Reddit search engine. We have not been able to find official documentation on how their search engine sorts by relevance but manual searches point towards it taking in factors such as popularity (upvote/downvote-ratio), the amount of times the search term was mentioned in the comments, and how recently the post was created. We chose to sort by relevance instead of popularity as we experienced that sorting by popularity led to posts that were more likely to be irrelevant for smaller movies. We then save each top-level (reply to the original post) comment in these threads. We have chosen to collect only top-level-comments as we believe that the risk of comments being unrelated to the movie increases for comments on comments.

The output is a dataframe for each movie containing on average 242 comments, with the variables being the text of each comment as well as the timestamp of said comment.

3.2 Merging Datasets

After the data collection from IMDb, Rotten Tomatoes, and Box Office Mojo, we have three datasets with information on our movies. We merge these datasets to produce one large dataset with several different data points on each movie. To ensure relevant Reddit comments and faster processing, we have chosen to use movies from 2010 and forward. Going further back in release dates would lead to comments from Reddit becoming more scarce and irrelevant as the site had not quite gained traction in the general public[11]. Using only movies with later release dates would lead to a higher amount of relevant posts and comments per movie but would restrict the amount of movies we could utilize in our analysis. To keep a large amount of movies we chose 2010 and forward but arguments could easily be made for other years.

Movies do not have a unique identifier, such as the ISBN for books. This makes merging the datasets challenging. We choose to merge the datasets on both the title of the movie and the year of release. Merging on these parameters poses a few challenges:

[11] https://subredditstats.com/r/movies

1. **Movies released in the same year with the same title:** There are a few cases of two movies being released in the same year with the same title. This creates duplicates with various combinations of data from IMDb, Rotten Tomatoes, and Box Office Mojo. To combat this problem, we have dropped observations with identical titles and release dates, while manually making sure that only the correct combination of data is left.

2. **Movies with titles which can be written in different ways:** Some movie titles can be written in different ways. This is prevalent in sequels with numbering and subtitles. Thus, the same movie might have slightly different titles in different databases. Consider for instance the movie 'Star Wars: Episode VIII – The Last Jedi'. Some places, the 'Episode VIII' part is not included in the title. Furthermore, it is not clear whether one should write 8 in Arabic or Roman numerals or whether or not to include ':' and '-'. To mitigate this issue, we have dropped special characters. Different capitalization can also create issues but lowercasing all letters solve this problem. This issue with discrepancies in titles leads to losing movies in the merge as the merge only brings in data with titles that match. All 3 databases use international titles but it would not be farfetched to assume that some titles would keep their original in one of them.

3. **Movies with ambiguous release dates:** The release date of a movie is not always clear. Sometimes, a movie might have been released at a movie festival in one year and then released to the general public the next year. Or the movie might have been released to some countries right before the change of year while it was released to other countries after the change of year. This could in theory create discrepancies between the different databases. However, we consider that this is probably a very minor issue, affecting at most a handful of movies, if any at all.

Merging the three datasets, the resulting dataset has 1594 movies. This is lower than the potential maximum of 9984 movies, which is contained in the dataset with the fewest movies (Rotten Tomatoes). We think that there are probably two main reasons for the loss of observations. First of all, the datasets do not overlap completely in movies. As we are doing an 'inner' merge, as long as a movie is not present in one of the three datasets, the movie will be dropped. Therefore, many movies are probably dropped due to this. Additionally, a not insignificant number of movies is probably lost due to the same movie having slightly differently stylized titles, as outlined in point 2 previously.

	IMDb	**Rotten Tomatoes**	**Box Office Mojo**	**Merged**
Observations	237,407	9984	11,159	1594
Variables	8	13	8	27

Table 1: Overview of the size of datasets

All in all, we consider that the end result of 1594 movies is plenty for our analysis. Randomly selecting movies has led us to find that a relevant post in r/movies usually includes a link to the official trailer where comments are expected to be quite relevant.

Having more movies to collect comments on would start to pose a significant problem in terms of how long it would take to collect all the top-level comments from Reddit. The collection of Reddit comments took approximately 6 hours but would have been drastically longer if more than the top 5 relevant posts had been scraped for comments.

Table 1 contains an overview of the size of the three datasets and the final, merged dataset when limiting the release dates to be 2010 or later.

3.3 Ensuring Data Quality

We have taken a number of steps to ensure the quality of our data. Concerns mainly arise regarding the data scraped (the data from Rotten Tomatoes and Box Office Mojo). These steps are as follows:

1. **Logging requests:** For the two websites we scraped, we have created a log of all requests. The first scrape of Rotten Tomatoes where we list all movies showed a response code 200 in all but 2 instances. Both instances happened seemingly at random (page numbers 192 and 222) and the scrape correctly collected the missing data immediately thereafter. Scraping data from the individual movies on Rotten Tomatoes caused only 3 errors where the scrape collected the missing data immediately thereafter. Scraping Box Office Mojo led to no failed requests. Plotting the size and time before responses led to no peculiarities in either scrapes.

2. **Random manual inspection:** We have gone through a number of random observations in the dataset to ensure the data on movies is correct by cross referencing with the live websites. The same has been done with random Reddit comments to make sure that the API consistently found the desired

top-level comments on the top 5 relevant posts. This was especially relevant given the lack of an automated log for the API.

3. **Looking for impossible results:** No observations have been found with e.g. Tomatometer-scores outside the range of 0-100 and no IMDb-scores outside 0-10.

4. **Data cleaning:** The merged data has been checked for duplicates and has been checked for missing values. Any instances have been removed from the dataset.

5. **Coding with data quality in mind:** When coding the data scraping, we have tried to make the code as robust as possible, such as to avoid small changes in the layout from one page to another causing issues in the scraping process.

3.4 Ethics

We do not consider there to be any ethics concerns with our project. First of all, the data is not on an individual level. In the case of the movie data, all the data is publicly available on several different websites online. For the Reddit comments, the data is also freely available online. Furthermore, we only store the text of the comment and the timestamp, not which user wrote the comment. Additionally, we aggregate the data of all comments on the movie-level, such that the individual comments don't feature in any way in the final results.

Furthermore, we have taken steps to ensure that the data collection itself does not pose problems to the websites from which the data is collected. In the two scraping cases (Rotten Tomatoes and Box Office Mojo), we have put in a one second sleep between every request, such as to avoid denial-of-service. In the IMDb case, no such concern is present, as all we have done is to download two publicly available datasets from the webpage. In the Reddit case we have used the publicly available API, so there should also be no concerns on this front, since the API limits the number of requests. Last, but not least, we are not monetizing or sharing any of the data collected, so there should be no legal issues on this front.

4 Descriptive Analysis

In this section of the paper we will present descriptive statistics of the cleaned dataset. The variables are presented in table 2.

Variable	Description	Use
IMDb rating	The average rating on IMDb	Target
Audience score	The audience score on Rotten Tomatoes	Target
Tomatometer	The critic score on Rotten Tomatoes	Target
MPAA rating *	Rating of a film's suitability for certain audiences based on its content	Feature
Genres *	20 genres. Up to three different genres per movie	Feature
Studio *	The studio which released the movie	Feature
Top actors	A count of how many top actors are in the movie. A top actor is defined as being in the top 500 total box office revenue.	Feature
Top directors	A count of how many top directors are in the movie. A top director is defined as being in the top 100 total box office revenue.	Feature
Sentiment	The average VADER sentiment score of the movie based on Reddit comments.	Feature
# comments	How many Reddit comments we scraped	Feature
Award words	How many words mentioning movie awards (e.g. 'Oscars' or 'Best picture') there were in the Reddit comments.	Feature

*Table 2: Presentation of variables used in analysis. * indicates dummy variables.*

After merging data from four sources the cleaned dataset ended up with 1581[12] movies made in the period of 2010 to present day. Figure 2 shows three histograms of the targets, which we use in the empirical analysis. Table 3 show descriptive statistics of the key features used in the analysis. Figure 2 (A) shows that the distribution of the Tomatometer is left skewed, where the bin with the most observations is the one showing movies with RT Tomatometer of 90-100. There are fewer observations with lower scores. This shows that the critics on Rotten Tomatoes tend to be very positive. The mean RT tomatometer score is 59.5 on the scale 0-100. In comparison, the IMDb rating (figure 2 (B))is somewhat normal distributed around 6.4, which is the value of both the mean and median. Lastly, figure 2 (C) is right skewed with the most observations in the first bin (0-50 million USD). There are 1132 observations in this bin which is equal to 71.6% of all observations.

[12]13 observations are dropped compared to the merged datasets due to some observations not being available.

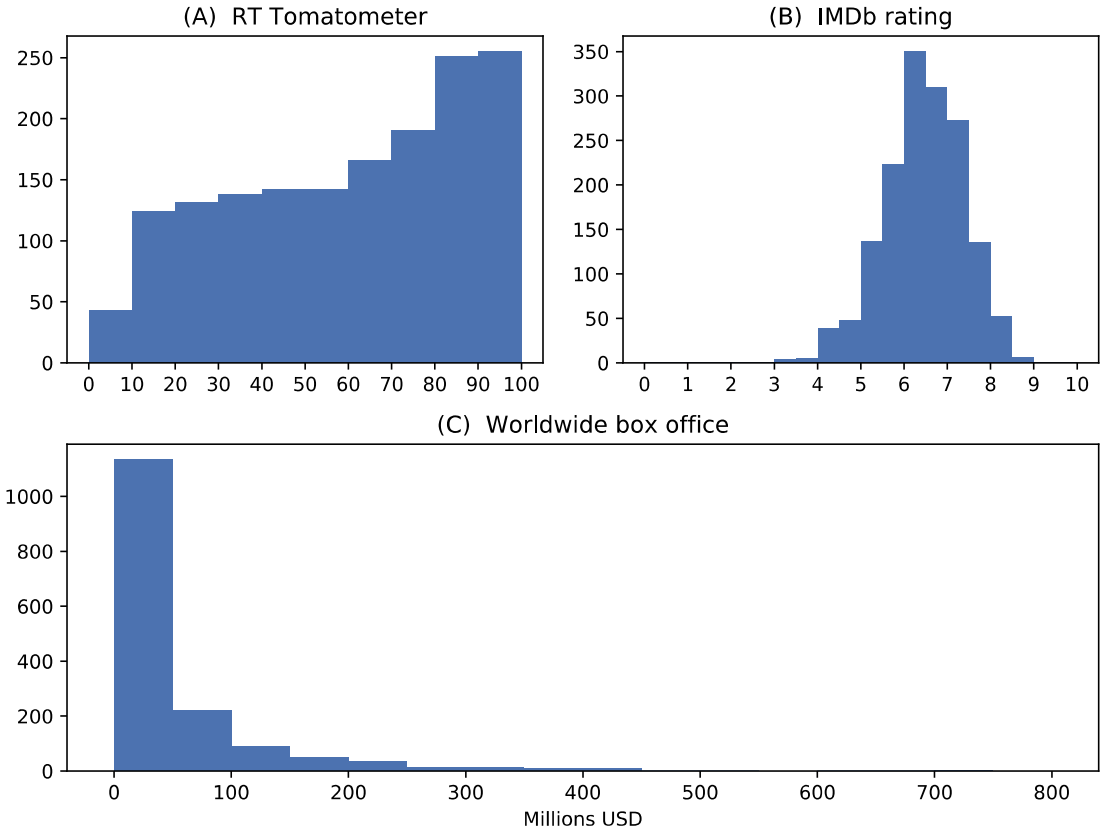

Figure 2: Histograms

Table 3 shows that one movie got as many as 2262 comments and one got as little as one comment. The mean is 242 and the median is 61, which means that 50% of our movies got 61 or less comments. The variable number of awards related words in comments show that at least 75% of the movies in our dataset has no awards related words. One observation got as many as 143 award related words in its comments. Looking at the sentiment score, we can see that 50% of the scores lie between 0.1 and 0.2 on the scale of -1 to 1. Therefore, there is little spread in the sentiment scores and they are mostly positive. The most negative comment has a sentiment score of -0.8 and the most positive comment has a score of 0.96.

Figure 3 shows three scatter plots of sentiment score before release and the three targets RT Tomatometer (A), IMDb rating (B) and box office (C). There is a positive correlation with the Tomatometer and a less positive correlation with IMDb. The correlation between sentiment and box office is negative. This opposes our hypothesis of movies with more positive expectations doing better. In no case is the correlation very large, indicating that sentiment does not explain movie success well.

	Mean	Std	Min	25%	50%	75%	Max
RT Tomatometer	59.5	27.2	0.0	37.0	50.0	84.0	100.0
IMDb rating	6.4	0.9	1.6	5.8	6.4	7.1	8.8
Box office (mio. USD)	47.4	84.0	0.0	0.5	14.0	56.8	858.2
Number of Comments	242.0	354.2	1.0	10.0	61.0	431.0	2262.0
Number of Award words in comments	1.5	6.6	0.0	0.0	0.0	0.0	143.0
Sentiment score	0.2	0.2	-0.8	0.1	0.1	0.2	0.96

Table 3: Descriptive statistics of variables

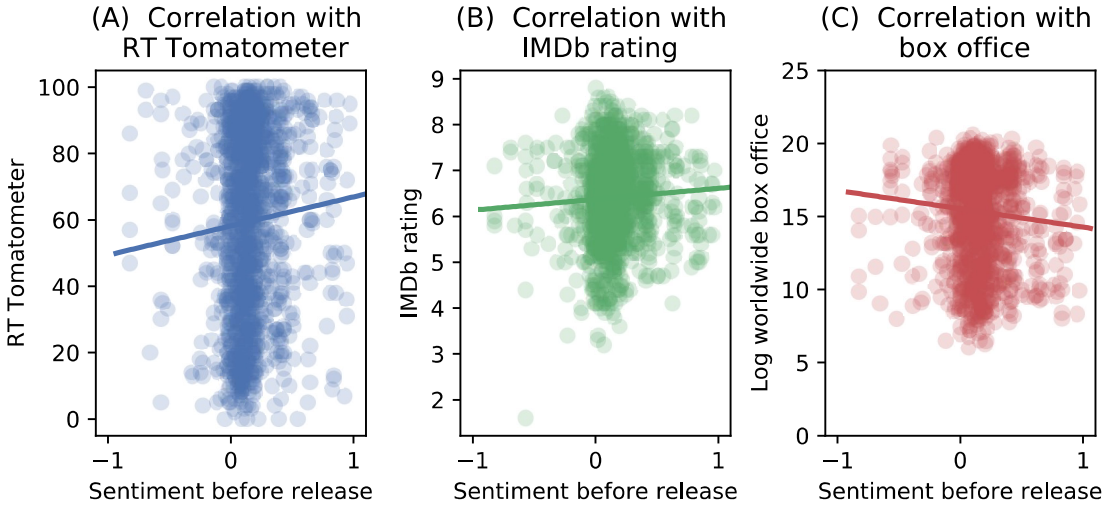

Figure 3: Scatter plots

5 Machine learning

5.1 The models

In this section we predict three different criteria for movie success (different targets): The Rotten Tomatoes Tomatometer (critic score), the Rotten Tomatoes audience score, and the worldwide box office. These three measures catch different aspects of what might make one consider a movie to be successful.

We fit three different model types: OLS, Lasso and Elastic net. We first split the data up into training and test data, where the test data takes up 25 percent. To pick the hyperparameters we use 10-fold cross-validation. For Lasso we use logarithmically evenly spaced λ's from 10^{-4} to 10^{4}. For the L1 ratio we try the parameters [0, 0.25, 0.5, 0.75, 1].

5.2 Results

The mean squared error and R^2 for the preferred models (the models with the lowest MSE) are presented in table 4.

	MSE			R^2		
	IMDb average rating	RT Tomato-meter	Log box office	IMDb average rating	RT Tomato-meter	Log box office
OLS	$4 \cdot 10^{11}$	$6 \cdot 10^{14}$	$9 \cdot 10^{12}$	$-4 * 10^{11}$	$-8 \cdot 10^{11}$	$-1 \cdot 10^{12}$
Lasso	0.726	618	2.93	0.245	0.185	0.664
Elastic net	0.701	609	2.88	0.273	0.196	0.669

Table 4: Mean squared error (MSE) and coefficient of determination (R^2) in test data for different models.

We first of all notice that OLS performs much worse than just a constant for all three targets. This is clear sign of overfitting. We have 305 features and 1181 observations in the training dataset. Therefore, OLS achieves a high degree of precision within the training dataset, but performs extremely poorly in the test data. This is due to two issues which capture the problem with overfitting: More spurious coefficients and larger coefficients, leading to good predictions in the training data and very poor predictions outside the training data. The fact that OLS is as poor as it is might be a surprise. By examining the coefficients, we find an explanation. For instance when the target is log box office. Here, the coefficient on the dummy for the movie being made by Abramorama Entertainment is about minus 10 million. This large coefficient is likely a consequence of collinearity, which in general is an issue with several features. Considering that no log box office exceeds 20, just a single movie being made by Abramorama Entertainment would lead to an extremely large error being made by the model. This is indeed the case, as a movie made by Abramorama Entertainment in the test data has a predicted log box office of minus 2 million. This and a few other outliers in the predictions lead to the extremely poor model performance on average for OLS.

Lasso performs much better than OLS, achieving relatively small MSE's and high R^2's. Naturally, Elastic net performs even better than Lasso. However, the performance gain is so small that it cannot be considered significant.

Additionally, we notice that in terms of R^2, the models predict the box office much better than the ratings. Perhaps this is not so surprising: What makes a good movie is very subjective and different from person to person, while box office can

be better proxied by for instance top actors and/or directors, who historically have earned much money for their movies.

Having concluded that we have built models which to a non-trivial degree are able to predict movie success, we focus on the specific contribution of this paper: Can Reddit comments help us predict movie success better than 'just' the other variables? Here, we have three variables containing different pieces of information with regard to the Reddit comments (the average sentiment, the number of comments, and the number of words mentioning movie awards in the comments). We call these Reddit variables 1, 2 and 3. Here, we first look to the coefficients of the Lasso model. When using the IMDb average rating as a target, only Reddit variable 1 gets a non-zero coefficient. This is the same when using the Tomatometer. When using the box office however, Reddit variable 1 gets a zero coefficient, while 2 and 3 get non-zero coefficients. This indicates that the Reddit comments offer some use to predicting movie success, but it is very limited.

Additionally, we have fitted the models without the Reddit variables. Doing this, the MSE's and R^2's are basically unchanged. For instance, when excluding the Reddit variables, R^2 with box office as the target falls from 0.664 to 0.660 and 0.669 to 0.664 for Lasso and Elastic net, respectively. This indicates that the Reddit comments don't offer much to the prediction accuracy. We therefore conclude that, at least using our approach, Reddit comments don't improve prediction accuracy noticeably. This might be due to any number of factors, which we touch on in the discussion.

We report the validation curves in figure 4. Here, we see which lambda parameters minimize the MSE in the test data. These are the lambda values which represent the prefered models reported previously. From the graphs we see that we appear to at least have found a local minimum for all the lambdas (and not just an edge solution).

Figure 5 shows the learning curves for the Lasso and Elastic net models. In most of the models, we see that a larger sample size (larger training dataset) improves the precision in the test data, as expected. Furthermore, most models have a small difference between the MSE's in the test and training data for larger sample sizes, indicating a good fit.

6 Discussion

One potential issue of our analysis is the somewhat small number of observations (1581), compared to what is possible if we only used one source for data and thereby

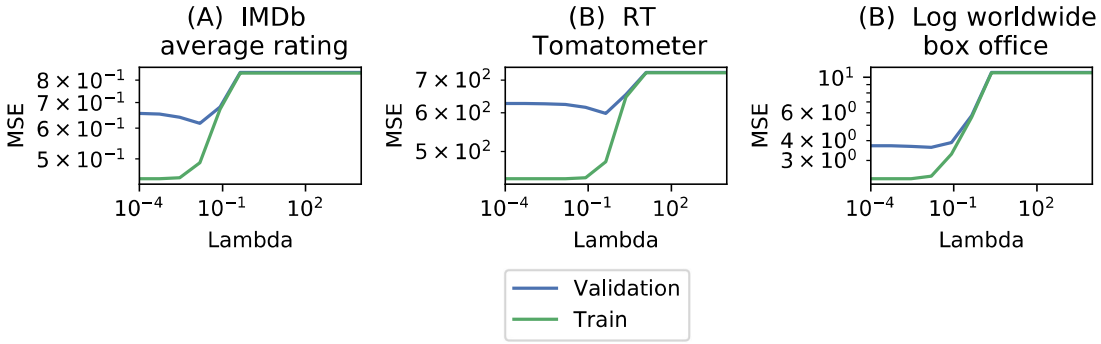

Figure 4: Lasso validation curves

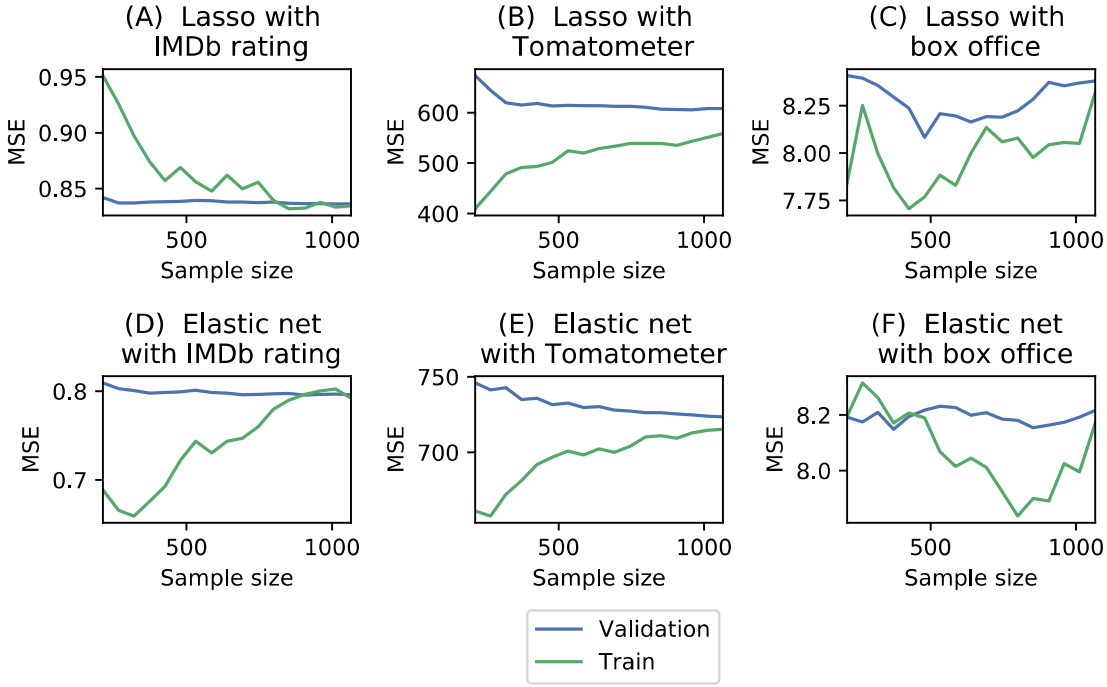

Figure 5: Lasso and Elastic net learning curves

avoided losing observations when merging, as discussed in section 3.2. Instead, we chose to have three different sources, such that we have fewer movies but data for all of our features used in the analysis.

One might wonder why adding the Reddit comments doesn't improve the precision of the predictions. There might be any number of reasons for this. First of all, people on Reddit just might not know how good a movie is before it is released. Second of all, noise in the Reddit comment data is a concern. Especially for smaller movies, there might not be any relevant discussion on Reddit, thus making the sentiment score for that movie irrelevant. Furthermore, it might be the case that if we created a specific model for the purpose of analyzing movie success, instead of just

using a generic sentiment analysis tool like VADER, we could extract more information from the Reddit comments. Additionally, we could have compared alternative programs for doing sentiment analysis. Lastly, it might be the case that the Reddit comments don't represent the general population. It is possible that people on Reddit, and specially in the subsection of movies, are more passionate about movies, or in other ways differentiate from the general population.

We might have been able to achieve larger degrees of precision in predicting movie success if we had more features. One example of such a feature is the movie budget. We have not included budgets for a couple of reasons. First of all, budget information is not publicly available in many cases. Second of all, we did try to include budgets, but this resulted in us losing about half of all observations, due to all films not having a publicly available budget. Additionally, we could have analyzed many other websites instead of Reddit: Twitter, YouTube, Google, Facebook etc. Each of these poses their own unique challenges and advantages. We decided that Reddit would be the easiest to extract the most useful information quickly.

7 Conclusion

We conclude that we can predict the success of a movie before its release better compared to just using the mean. Furthermore we have shown that adding Reddit comments as features to this predicting does not seem to improve the accuracy of the predictions, at least when using our specific approach.

8 References

Raschka, Sebastian, and Vahid Mirajalili. *Python machine learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow.* Birmingham, UK: Packt Publishing, 2017. Print.

https://www.rottentomatoes.com

https://www.imdb.com

https://www.boxofficemojo.com

https://www.subredditstats.com/r/movies

https://www.reddit.com

https://www.github.com/cjhutto/vaderSentiment

https://www.praw.readthedocs.io/en/v3.6.0/