CMPT 353 - Computational Data Science

An Exploration of Movie Data Statistics

Wael Yakoub Agha - 301348817

Samer Sefrani - 301384562

Aditya Rajvanshi - 301349754

Table of Content

Motivation

Movies are a popular form of entertainment and are consumed by the masses. They stand as achievements of creativity, storytelling, visual design, and soundtrack. However, there is never a clear formula for what makes a box office success. In this paper, we explore three questions that we believe will help in shedding light on what really makes a movie successful. Why would a movie about blue aliens on a distant planet be one of the highest-grossing films, while a beloved bat-like superhero struggles to succeed in the cinema despite having high world-famous actors, directors, and a high budget? To explore this, techniques such as data cleaning, ETL, Machine Learning, and statistical analysis tools were utilized to help answer this question.

Research Questions

1. Does IMDb give higher review scores to their movies than Metascore does?
2. What effect does the movie's plot have on the movie's rating and the number of voters?
3. What movie features can best predict the movie's IMDb rating and how accurate?

Gathering Data

To gather the movie data, we used the wiki data query service[1] and stack overflow for help with writing SPARQL queries[2]. We also used the OMDb API to run through the IMDb title ids from the wiki data to gather information on each movie. This was then stored as partially clean data, which would be further modified for each question specifically. We also incorporated into the dataset a previous collection of movie queries from wiki data that was stored as a JSON file.

Question 1

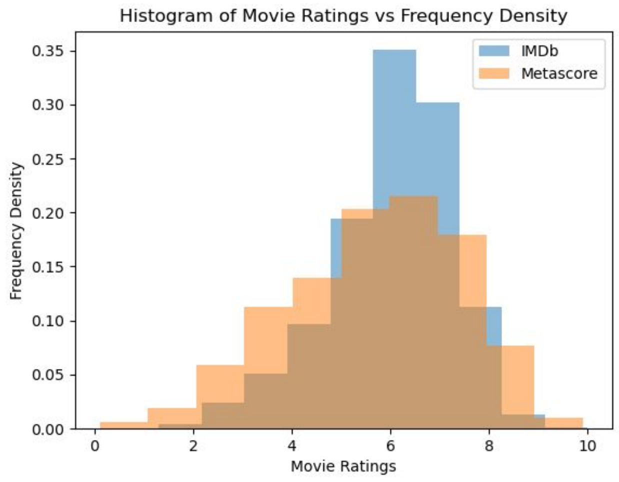
Before diving into the factors that lead to the success of a movie, we first wanted to use our analysis tools to see whether IMDb rates its movies higher than Metascore does. This would allow us to consider this as a factor when analyzing future questions. To do so, the data needed to be cleaned further.

Data Cleaning

Before cleaning the data, we first had to decide what data we wanted to keep. For this question, we only needed the movie title, the IMDb rating, and the Metascore rating the "omdb-data.json" file. Two data frames were created, one for IMDb with the movie title and IMDb rating, and one for Metascore with the movie title and Metascore. For ease, we will refer to the rating and Metascore columns as "score". All the "N/A" scores were removed and the score columns were then converted to floats and integers which can be manipulated easily. Finally, the indices for both data frames were reset.

Modeling

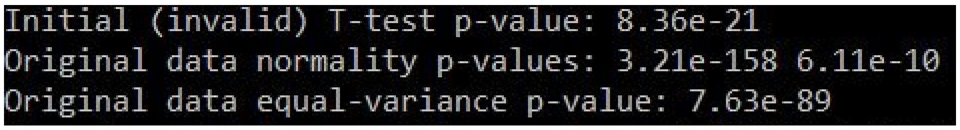
At this point, we decided to plot our data as a histogram, to see what the distribution of the data looked like.



The results were unsurprising. The distributions seemed fairly normal, at least to the eye. It is worth noting that after cleaning the data, we were left with roughly nine thousand data points for the IMDb data, but only one thousand four hundred data points for the Metascore, so there is a significant difference there. Regardless, we proceeded by conducting a few tests on the data.

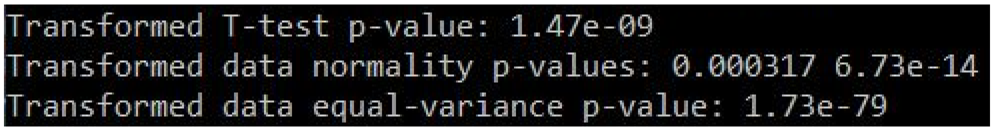
We began with a T-test, as a simple and quick way to see whether the means of the data were different or not. Here, our H_0 = "The means of both data sets are the same". The initial T-test had a p-value of 8.36e-21. This value means that we can reject the null hypothesis and claim that these two data sets have different means. Additionally, the mean score for both websites were roughly 6.03 for IMDb and 5.68 for Metascore. However, before conclusively stating that IMDb typically rates their movies higher than Metascore, we did some more testing.

We did a normality test, and a Levene test to ensure that the results of the T-test were valid. In this case, for both of the tests, our H_0 = "the distribution is normal, the data is of equal variance". However, the tests both came back with p values that were far smaller than 0.05, which was our alpha for all tests. That meant we had to reject the null hypothesis.



```
Initial (invalid) T-test p-value: 8.36e-21
Original data normality p-values: 3.21e-158 6.11e-10
Original data equal-variance p-value: 7.63e-89
```

These values were very surprising since the data looks very close to normal, yet the results suggest nearly 100% confidence in the fact that the data is not normal and not of equal variance. This meant that we tried to transform that data and see if it improved the results. Several transformations were tested, such as square roots, exponentials, logarithms, and even higher-order powers. Yet, the best results came from squaring the data but were still resulting in very small p values. The normality tests seemed better for IMDb ratings but worse for Metascore, which left us very confused.



```
Transformed T-test p-value: 1.47e-09
Transformed data normality p-values: 0.000317 6.73e-14
Transformed data equal-variance p-value: 1.73e-79
```

We decided to use a Mann-Whitney U-test since it does not require the data to be normal or have equal variance. The values were also independent and ordinal, which meant that this test could be used. The null hypothesis remained the same as that of the T-test, and our p-value was 2.74e-11.

Ultimately, despite the strange results from the normality and Levene tests, even after several transformations, using the Mann-Whitney U-test allows us to conclude that IMDb does score movies slightly higher than Metascore. We would feel more confident in concluding this if we were able to access more data in a reasonable time frame. However, knowing this, we can now be mindful of the results of the forthcoming discoveries.

Question 2

After using our analysis tools to see whether IMDb rates its movies higher than Metascore does. We decided to take our research a step further to see how the movie plots (descriptions) affect the ratings and how it affects the number of people who rated the movie.

Data Cleaning

For this question, we only kept the [‘Plot’, ‘imdbRating’, ‘imdbVotes’] columns and we dropped the other ones. After that, we cleaned the data by dropping the rows that are empty or are nulls, and all scores that were “N/A” were removed. The “imdbVotes” values were strings with commas in the number, so we deleted the commas then converted them to numeric values. However, the “Plot” column required much more cleaning in order to be used for our analysis, and these steps were:

- All stop words such as pronouns and prepositions (defined by the NLTK.corpus library) were removed
- All words in the sentences were transformed to lower case
- All punctuation was removed

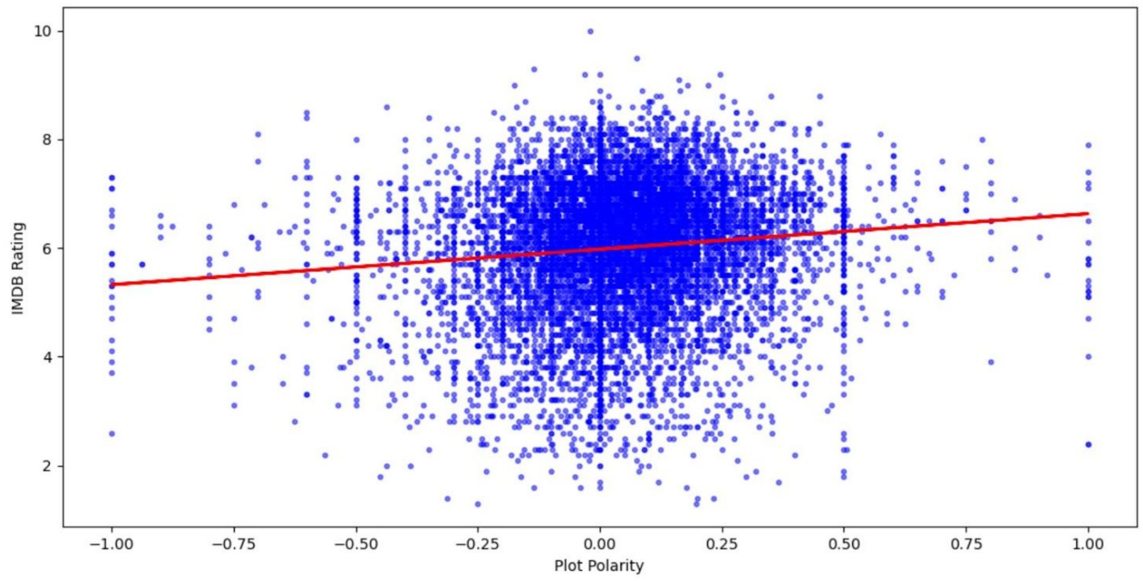
The reason for this was to make the sentences as simple as possible for the sentiment analysis which will be discussed in the section.

Modeling

We used the TextBlob library to get the sentiment score for each movie plot. The function "sentiment.polarity" returns the sentiment score for every string. The sentiment score is defined as follows:

- A value in the range of [-1, 1]
- A value of -1 sentiment score will be given to words with a negative sentiment such as kill, sad, torture, die.
- A value of +1 sentient score will be given to words with a positive sentiment such as family, happy, heaven.
- The zero value means a neutral sentiment.

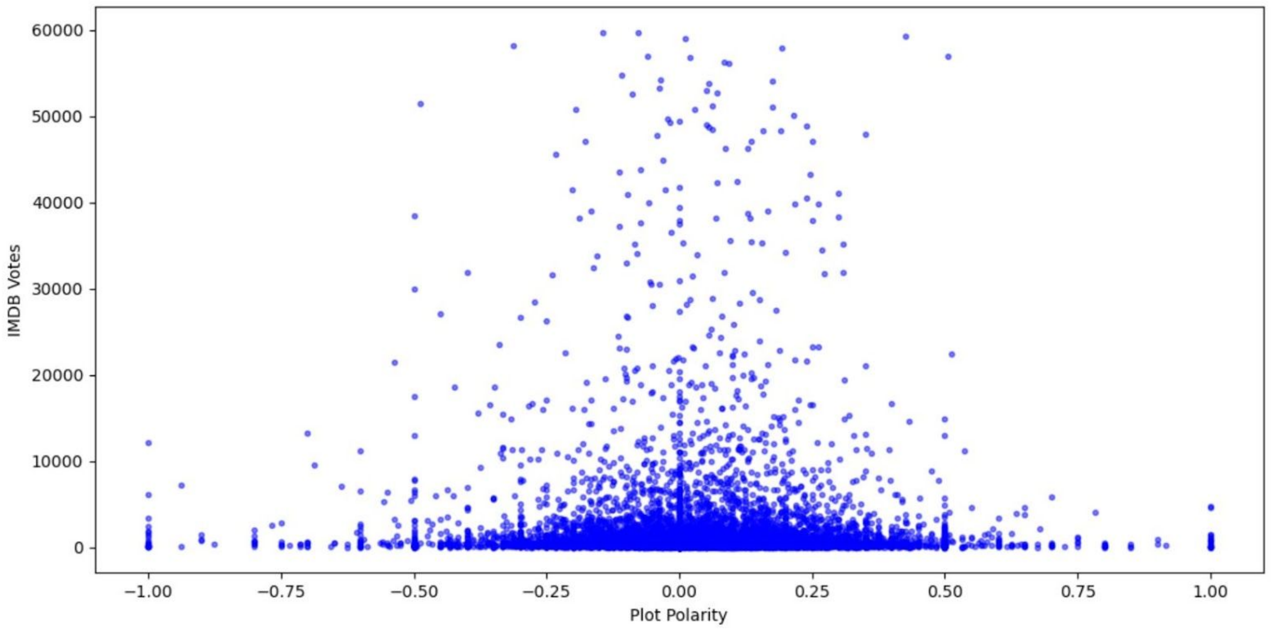
The polarity scores of the movie titles were added to the data set, and it was time to study the relationship between the polarity scores and the IMDB ratings. To achieve that, we produced a plot of polarity score vs. the IMDb ratings. Also, a Linear Regression model (from Python stats library) was fit to the data to see if there is a linear relationship. The following graph was the result:



The linearly increasing (slope = 0.65) relationship suggests that viewers tend to rate the movies with a more positive plot higher than the movies otherwise. The linear fit had a very small p-value of 9.64e-27 which is way less than 0.05. Therefore, we can reject the null hypnosis and conclude that the polarity score and movie ratings are dependent linearly.

After that, we wanted to see if the polarity of a movie's plot has an effect on the number of people who would watch the movie. However, we did not have the number of viewers as one of our available features, and thus, we chose the number of IMDb voters instead. The assumption here is that the more people rate a movie on IMDb, the more people have actually viewed the movie.

The following is a plot of polarity score vs. the number of IMDb voters:



By observing the results from the graph, it shows that the highest numbers of IMDb voters are mainly around the center and in the polarity range [-0.25, 0.25]. This suggests higher voting (and also viewing based on our assumption) with neutral polarity scores. It also seems with extremely positive and negative plots, there is way less voting/viewership.

Question 3

This question aims to identify a combination of the movie features that can best predict its rating. For such purpose, an ML classifier model was created, and various combinations of input features were tested to see which produce predictions with the least testing error.

Data Cleaning

There were two main data sets to be created:

1. Target Values y

   The target values were the IMDb movie ratings after deleting the rows in which no rating was found. Since an ML classifier was used, the ratings needed to represent labels. In order to have a better prediction, the ratings were multiplied by 10 and turned into integer values. Thus, there were 100 labels from 0 to 100 which correspond to ratings from 0.1 to 10 on IMDb.

2. Input Features X

   The exact combination of input features is the target of this question. Therefore, the data cleaning was done on many movie features and then, various combinations were tested to see which produced the best model. The input features had to be numeric, so the year, the runtime (movie duration), and the number of votes only had the "N/A" values removed and the numeric value extracted. However, non-numeric values such as the genre, the language, the actors had to be dealt with differently. The features were either strings or lists of strings. The lists of strings were first split into an appropriate number of columns, each containing an element of the list. For example, the "Genre" features were split into "Genre1", "Genre2" and "Genre3" where each column had one genre value. After that, each column's string values were mapped to unique integer values using Python's dictionary. Since question 2 used TextBlob to calculate the polarity of each movie's plot, an additional column with the polarity scores was also added to the input features.

Model

The Random Forest Classifier from the SciKit was used as our ML classifier model. After manual tuning of the model parameters, the following produced the best results: N_estimators = 50 and Max_Depth = 10.

Model.Score( ) was not used to measure the quality of the model since it measures the model's success in terms of the number of successful classifications. This does not reflect how good the model is because the aim is to see how far the predicted ratings are from the actual ratings. Instead, the following error calculation was used to measure the model's performance:

$$Error = \frac{abs(Predicted - Actual)}{Actual} * 100$$

The model uses train_test_split to randomly split the X input data to 66.6% training data and 33.3% testing data. Because of this random split, the error percentage can go down or up a little due to the randomness which made it less easy to detect the improvements in the model. The solution was to run the process of creating, training, and testing the model 100 times for each suggested combination of input features. The one the produced the least average testing error of 16.86% was:

- The year the movie was made
- The three main genres that define it
- The rating by the industry (PG13, Adult… etc.)
- Top two languages it is available in
- Top two countries of production
- The duration

Project Limitations

The first limitation was the size of the dataset we are able to obtain. The API we used had a limit on the number of requests, and so we were only able to gather 10,000 points. Although it is not a low number, given a much bigger data set, the exact research we did would have had definitely more insightful results, especially the predictive classifier. Additionally, for question 2, we wanted to study the relationship between a movie's plot and the number of views it received. Since we did not have that information, we made the assumption that the feature with the number of votes on IMDb can replace it. Lastly, trying different combinations of input features for the classifier and re-training the model for each one was quite time-consuming. Given a more powerful machine, these calculations would have been faster and better input features would have been found that would improve the overall model's accuracy.

References

[1] https://query.wikidata.org/

[2] https://stackoverflow.com/questions/41824779/how-to-get-imdb-id-for-some-film-article-from-the-russian-wikipedia

[3] https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

[4] http://www.omdbapi.com/

Project Accomplishment Statements

Wael Yakoub Agha

- Designed an ML model to predict the IMDb ratings of movies based on a set of input movie features
- Conducted a thorough cleaning of the data and conversion of non-numeric data to numeric so it can be accepted as input for the model
- Used Python's Random Forest Classifier that achieved with parameter tuning and feature engineering, it achieved a low-error of 16.86%.
- Trained and tested the model over many combinations of the input features to find the model with the best performance

- Took part in the group meeting, project ideas research and helped debug my teammates' code

Samer Sefrani

- Cleaned the dataset and especially the movie plots to prepare for analysis
- Implemented TextBlob to compute polarity for the movies
- Compared the plot polarity with IMDB ratings, and plot polarity with IMDB votes
- Visualization of the comparison mentioned above through graphs
- Expanded the dataset from 900 rows to 10000 by fixing the bug in the program that fetches data

Aditya Rajvanshi

- Took part in group meetings, discussions and presented many ideas for possible questions
- Helped in accessing APIs and using SPARQL
- Cleaned the data, converted non-numeric data to numeric data for calculations
- Used data tools to analyze and explore relationships