Capstone Project 1: Milestone Report

Problem Statement

The dirty secret in the movie business is that less than half of all films released will turn a profit. Everyone has their hand in the jar to take a piece of the revenue. From top ranked actors who score percentages of box office draw to distribution companies, movie houses, and promotional networks, the pot gets ever smaller before the investors can start to recoup their costs The return on investment for a production company is barely sustainable.

One group that feeds from the movie revenue sources are the actors. Many times, perceived star power allots too much money to one individual in a film. It is a big risk to invest so heavily in the revenue generating abilities of a select few people. After all, most actors, even great ones, will put out a bomb.

Investors should wisely decide how much to invest in the leading roles of a film, while comparing this expense to how much revenue the film is expected to create. When looking at which actors could favorably tilt this balance, an actor's value should be largely based on the return on investment generated by that actor's previous films.

Can the financial success of a movie be fitted on its actors to give them an additional tangible quality? Can this value be quantified to make predictions of an actor's value? Can a production company use these predictions to make better investment decisions on who to hire for their films?

Dataset

I will build my own dataset from API requests from the TMDb website. The people at TMDb make extracting their vast set of movie information very easy to do. They will gladly provide an API key, which can be used to collect their data. They have an extensive user guide to help navigate their website that includes the ability to test out a request to see what the response will be ahead of time. Also, they've recently lifted their request limits. So, data acquisition is worry free. Their data is extensive, as well. It includes characters, billing positions, release dates, genres, casts, crews, imdbID's, budgets, information on whether movies belong to a collection, and revenues. For actor data, they have dates of birth, imdbID's, and filmographies.

The data acquired for the first capstone project was obtained from The Movies Database (TMDb) website using a personal API key that they provided to me. The data was collected over the course of eight days by the use of a request and storage program that I wrote. It sequentially requested movie and actor data based on the unique TMDb ID for each item, starting from ID number 1. The movie and actor data were collected separately through iterations of the requests script and stored in their own JSON files to be munged by additional programs. The sequence of requests terminated when the iterator reached the ID of the item that was added to

the TMDb database most recently. These final ID's can be found with a specific type of request from the website.

Creating the request and storage program was simpler than I had anticipated. I needed to account for failed requests, because the connection would terminate after three tries. I scripted a counter with a sleep failsafe that waited three seconds until sending the next request in the event of a third failed attempt. After collecting my data, I had many large JSON files, each with 100,000 rows. I created a program that merged and stored this data into two very large JSON files, one with 2,700,000 rows for the person requests and one with 800,000 rows for the movie requests. They contained many empty rows that were the result of failed requests, mostly due to IDs being removed from the TMDb database at some point. By building the datasets through iterating over a range of integers I hoped that their TMDb IDs would be unique, which would simplify any incorporation of data from other sources which used that key. When I checked, this was not the case for the actor dataset. I was not sure of the reason for this, but I knew that there will always be the option to go back and do some sample requests manually to see if this error was due to the website or my program. Regardless, these observations were relatively small in number. So, I removed the duplicate rows for now.

With both the person and movie datasets, I performed the standard data wrangling methods of dropping both the empty rows and unnecessary features, checking for missing data in the form of zero values and empty lists and dictionaries, converting dates to datetime objects, and ensuring all values were of the data type that would be most appropriate for the project. For both datasets, I dropped all of the rows that corresponded to adult films, as well.

For the person dataset, after checking the value counts of the movie department feature, I noticed that there were counts for actors and acting. So, I combined those columns. The ID feature values were mostly floats, but had several strings. Upon observing these rows, I saw that the values for each of their columns were labeled as "Acting". I removed those rows. Also, I dropped the observations for those that represented non-acting jobs, as I did not need the crew data for this project. The gender column had four different values. After searching the TMDb chat room, I discovered that gender zero was for missing data. I never found out what the fourth gender represented, but as there were just a few of them, they were removed along with the gender zero observations. This left gender one for female and gender 2 for male, which I converted to value zero to express the variable with binary values. The actor birthday variable had some strange values that went back into the 1800s. Some of these came from documentary movies, where people from archival footage are listed in the credits. I left these values in the dataset, as they belonged to legitimate roles that could be a factor in bringing revenue to a documentary movie.

For the movie dataset, I dropped all rows that did not have English as the original language. Also, I only kept movies that had values labeled released for the movie status column, as other types of projects should not have revenue numbers. The genre values were contained in dictionaries within lists. Most of the lists consisted of multiple dictionaries. Each dictionary

represented one of nineteen different TMDb base genre types, keyed by a number with a value that was a string description of the genre. I wrote a script to sort through the column and return a Series, keeping only the genre name strings in unnested lists. Then, I replaced the old column with the one I had created. I verified that the IMDb IDs were all unique, as well. This gave me two ways to check if two movies were the same, in case I needed to fill in missing values with data from another source. I dropped all rows for movies with runtimes under 75 minutes, as that is the cutoff length for feature length films.The credits column consisted of dictionaries, each containing two lists, one for cast and one for crew. I extracted the two lists with the Pandas JSON normalize function and put them into separate columns. Finally, the reviews column contained dictionaries, as well. I only wanted to keep the text of the reviews, which I extracted the same way and put them into their own column, as well. Once again, the two datasets were stored as JSON files to be further wrangled on a per use case.

As mentioned, the data was given a generalized form, as missing values had not been removed or imputed and most features were retained. Now, I had the ability to perform further munging to get the data into the best form to be used in this particular project. As far as dealing with missing values, I felt that this dataset was so rich that I could proceed by simply dropping their rows. If I found it necessary to have more observations, I could easily rebuild the datasets with simple modifications to my programs, then extract more values from other sources, while keying them in using the unique IDs found in each observation.

Because many of the dates in the datasets were pre-epoch, I needed to store them in ISO 8601 format, which gave them timezone stamps. After reading in the data, I converted these dates back to Pandas datetime objects, while removing the timezone information. This made the time data usable, again. I dropped the IMDb ID column, as I wasn't planning on adding data at this point. I removed any text data, as well as the movie crew column. These were gathered for projects that could involve more advanced methods of analysis.

I decided to represent the monetary values in today's dollar amounts. I obtained the Consumer Price Index (CPI) from the Federal Reserve Economic Data (FRED) website. From this, I created a CPI multiplier column and converted the values to current dollar amounts. This put the data on equal footing for comparison. I decided to keep the high end monetary outliers, because this project aims to analyze the return on investment numbers, and I wanted to observe the maximum range of what has been achieved for that figure. As far as the low end monetary outliers were concerned, I felt that there should be a minimum budget value to be considered for this project. After looking at a lot of the movies in that range, I decided it would be best to set the low end cutoff at $40,000. That didn't seem very high, but I would have lost some good data on well known actors had I lifted the line higher. All of the other outliers are real values that reflect the broad range of movies being made. They were retained for now.

As mentioned earlier, I wanted to have return on investment values for each movie to use as aggregated metrics for each actor associated with the movie. I chose a simple profit over budget ratio to use as these values. While not reflecting the true nature of movie economics, these

ratios can be used to weigh actors against each other, without implying any literal monetary significance.

Similarly, I combined the vote count and vote average for each movie, but instead of using a simple ratio, I used a true Bayesian average. This added a weighting factor which adjusted the average score of movies with low vote counts to be more reflective of the average vote across all movies.

The genre lists that I created for each observation potentially contained multiple base genres. I extracted those base genres and featurized them into 19 columns. This meant that if a movie was described by multiple base genres, it was now represented as having a count of one for each of those genres.

The final feature of the movie dataset was the cast information. Each observation was a list of dicts. Each dict held the information that pertained to that movie for one actor. I extracted the TMDb ID for each person to use as the unique key when it came time to combine both datasets. The other item I kept was the actor's billing in the movie. This would be needed to weight each actor as to their importance in the movie. It was assumed lead actors would be more responsible for the success or failure of a movie than someone in a minor role. Through the use of Pandas explode and JSON normalize methods, I was able to featurize these two sets of values from the cast lists.

As mentioned, I wanted to apply a weighting factor to the actors which amplified those at the top of the billing. The transform function I chose, killed off an actor's influence around the fifth or sixth spot in the billing order.

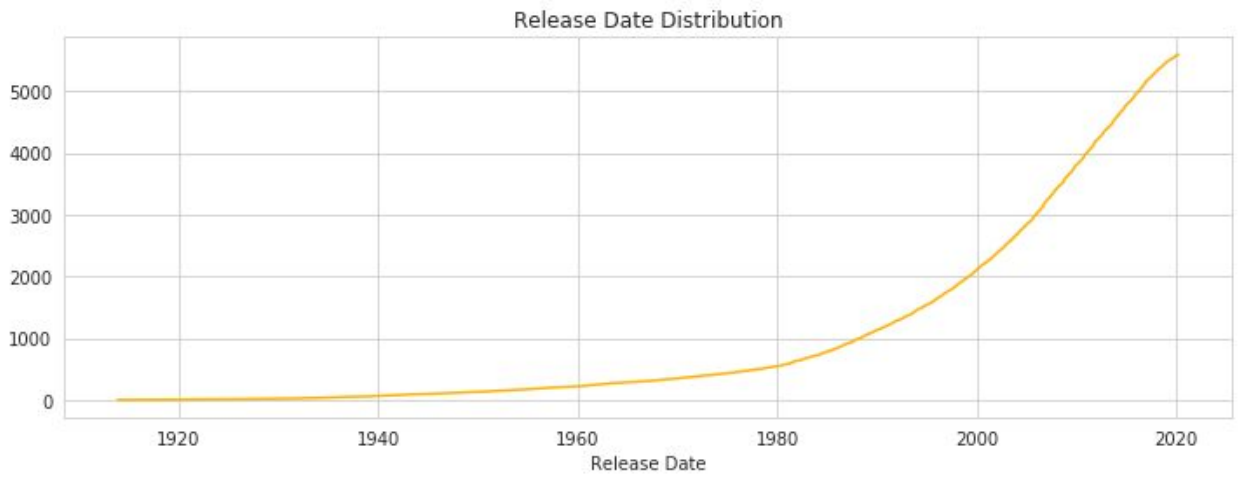
These weighted billing orders were then absorbed into the ROI data for each movie to give weighted ROI values. The billing order feature was then dropped.

After merging the two datasets, I created an age feature derived from the release date and the birthday features, whose values are the age of the actor at the time of the movie release. Then, the parent features were removed along with the movie ID and actor name features, as those values could be found in the parent datasets if needed.

The dataset was then rearranged through a Pandas groupby call by the actors, with the genre values being summed over each actor. The remaining values were averaged over each actor, then binned to reduce discontinuity in the response, due to any punctuated effects in the data. The observations, at over 26,000 in number, far outnumbered these 4 non-genre predictors. This allowed these predictors to be binned such that the range of response within each predictor band would be small, allowing the average response to be more precisely determined. The data type of these variables were then converted to integer to increase performance. The data was now ready for EDA and modeling.

Data Story

After exploring the dataset, there were a couple of interesting findings. The number of movies released each year gradually increased until the 1980s, when the rate accelerated in the 1990s and early 2000s.

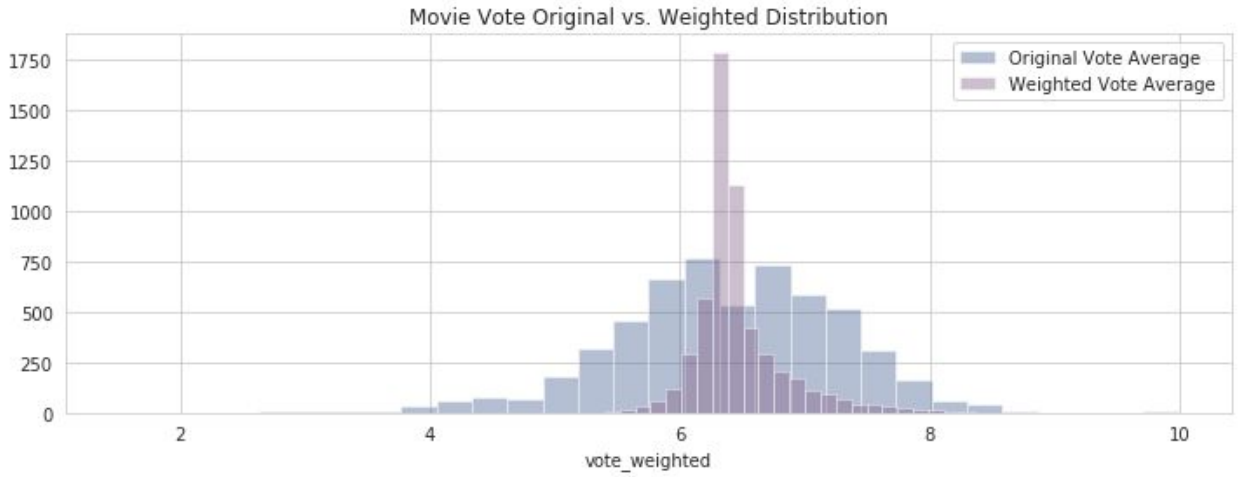


Next, I was surprised by the titles which held the top spot for revenue. While Avatar was well known for holding this distinction, that didn't hold true once the figures were adjusted for inflation. The top movie for revenue was Gone with the Wind ($7.4 billion / 1939). Three more movies followed before getting to Avatar ($3.3 billion / 2009). They were all animation movies released over half a century ago: Alice in Wonderland ($5.7 billion / 1951), Bambi ($4.2 billion / 1942), Snow White and the Seven Dwarves ($3.4 billion / 1938).

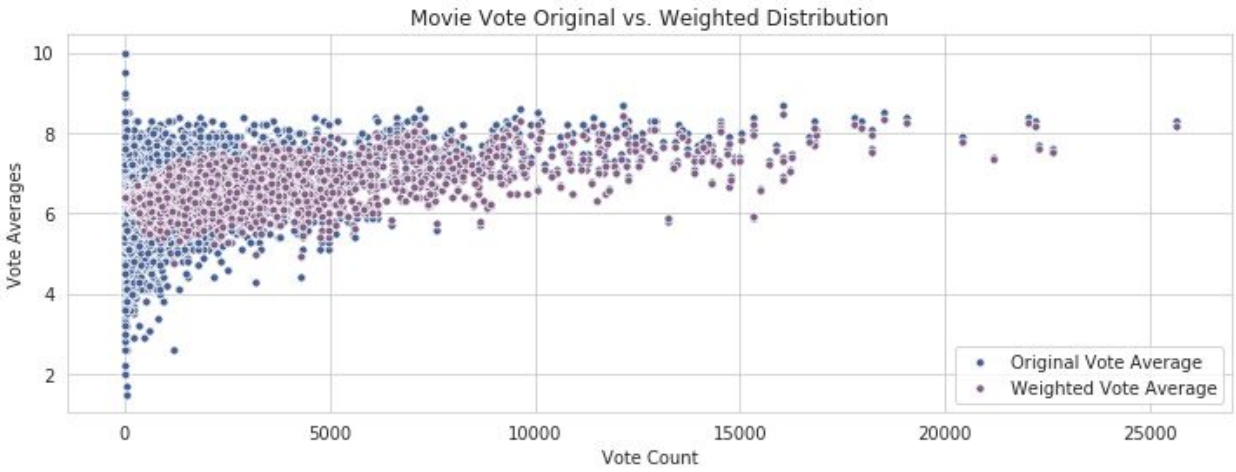
The percentage of movies in my final dataset that had positive return on investment (ROI) values was 69%. The ROI simplified a true profit value, as it didn't account for below-the-line

expenses, such as distributors, exhibitors, retailers, and sales groups, as well as actor and director shares of the producer profits. The true value has been known to be less than 50%. Not surprisingly, the movie in my dataset with the largest ROI was The Blair Witch Project (1999). It was made on a shoestring budget of $93,000 and collected $384 million in revenue, giving the film an ROI value of 4,100. To be clear, this was how many dollars came back for each dollar that was spent on production costs. I looked up a couple of more famous movies that I knew did very well in this category. The original Mad Max (1979) had a revenue of $366 million with production costs of $1.5 million, netting an ROI value of 249. The original Rocky (1979) earned $522 million and cost 4.5 million through production. Its ROI value was 116.

As I mentioned earlier, I applied a true Bayesian weighting function to all of the rating scores. I discovered this was a common way to adjust ratings to account for spurious data points and used by IMDb, Amazon, and many more companies. In this Bayesean way of thinking, our prior (the starting assumption) was that average rating scores were more common than very high or very low rating scores. Assigning an extreme rating score as a film's final average score would go against this prior assumption. The true Bayesian weighting function required more evidence (total number of extreme scores) before the rating was allowed to be shifted from the average to either extreme. This sorted out movies with a low quantity of ratings and returned their ratings as more representative of the whole sample. This method narrowed the variance of the average rating score distribution.



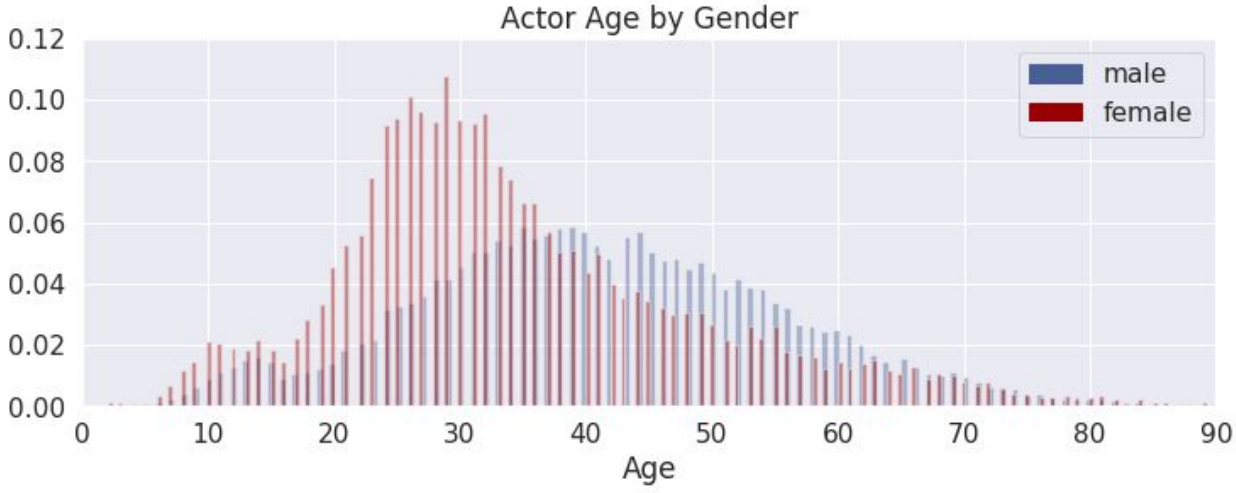
The movies with few total number of scores had a wide range of rating scores compared to movies with larger total number of scores. The latter were mostly concentrated between average rating scores from 4 to 8.5. This supported our prior assumption, as it was very uncommon to observe a movie with both a high total number of rating scores and an extreme rating score.



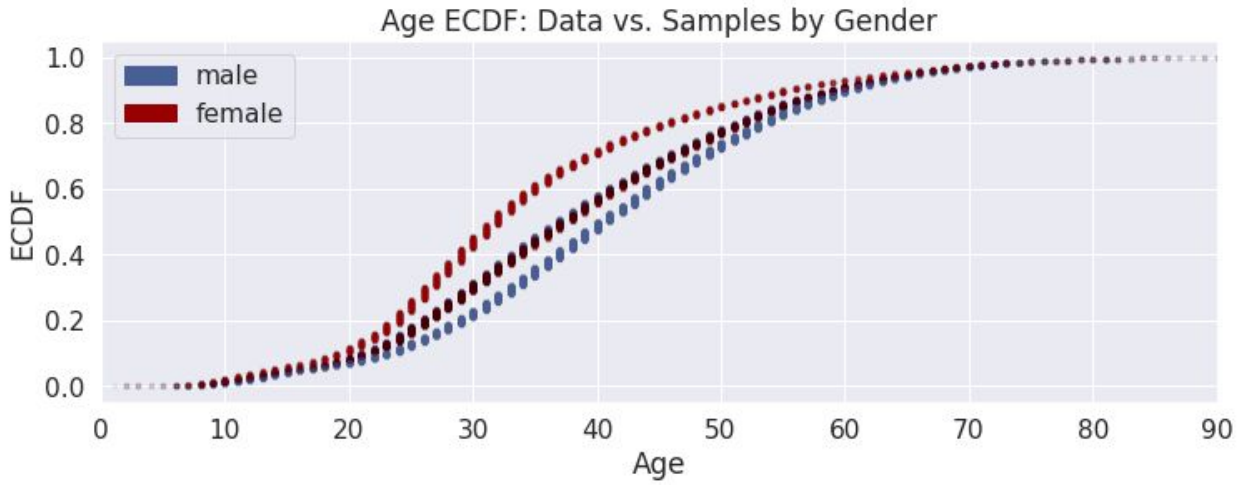
The sample average vote count was about 1,600 ratings. There was only one movie in this range with an average rating score less than 3, Dragonball Evolution. Movies with a very high total number of ratings were all rated above the sample average rating, which was around 6.4 out of 10.

Inferential Statistical Analysis

Observing the correlation values, I noted the strongest correlation among actor features was a negative one between age and gender, which had a rho value of -0.21. This implied that female actors had a shorter shelf life than their male counterparts for being cast in films. The roles for female actors were bunched together between the ages of 25 and 35, while the roles the male actors were more evenly distributed throughout their lives.

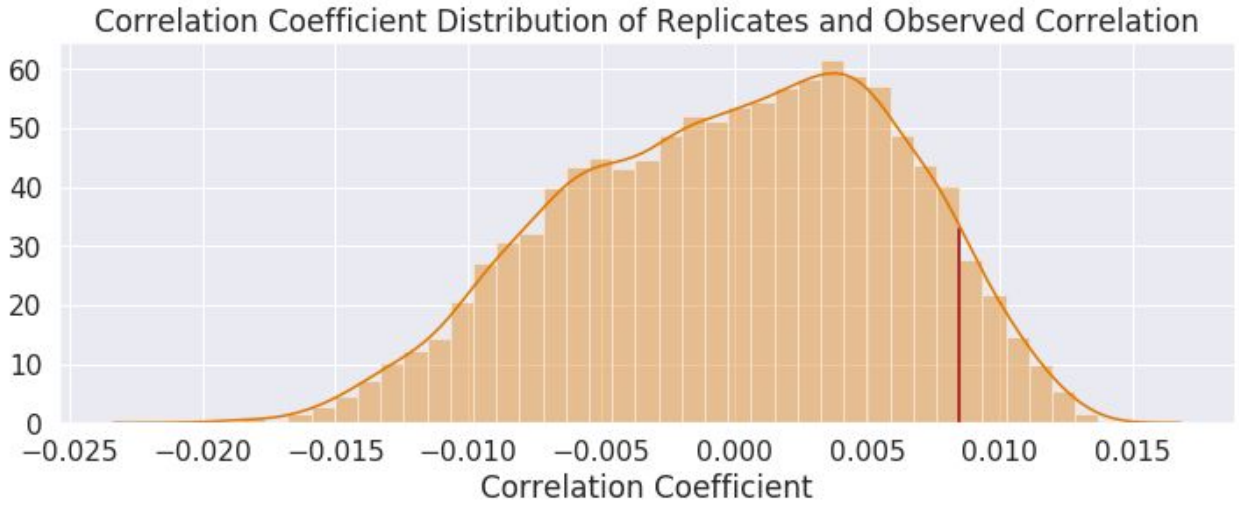


I explored this correlation with inferential statistics to verify its authenticity. I generated 10,000 permutation samples of each set. I plotted the Empirical Cumulative Distribution Functions (ECDF) for each and compared them to those of the observed data.



The curves showed a distinct gap between the observed data, while the curves of the samples overlapped. This indicated that the distributions between the genders were not the same. Next, I compared the difference of the mean age between the two genders with that of 10,000 permutation replicates. The p-value was given as zero. This meant that the probability was extremely low of observing a difference of the mean age as extreme as the one observed between the two gender sets.

I applied hypothesis testing to examine the correlation coefficient between the target variable (actor value) and one of the predictor variables (gender). The correlation coefficient I computed was very small (0.084), but they were small for all other predictors, as well. I permuted the values of each gender array and generated 10,000 permutation replicates of the correlation coefficients from them.



The p-value for this test was just over 0.07. This meant that there were 725 replicates out of 10,000 whose correlation coefficients were greater than the ones calculated from the observed data. This meant that we could not be more than 92% confident that the two variables were correlated.