Capstone Project 1: Final Project

Project Proposal

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 draws to distribution companies, movie houses, and promotional networks, the pot gets ever smaller. All of this happens before the movie investors can start to recoup their costs. The return on investment for a production company is barely sustainable.

One group of people that feed 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. All the while, they should be 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 revenue generated by that actor's previous films. This track record should be analyzed in the context of the specific qualities of both the actor and the actor's previous films. This will help to ensure that the actors hired for the top billing roles are the ones who are the best fits for the scope of the cinematic project.

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 to future works? Can a production company use these predictions to make better investment decisions on who to hire for their films?

Possible Clients

The types of clients that would be interested in the results of an actor value prediction model would be anyone involved in the hiring of actors. They would include, but not be limited to casting directors, directors, and producers in the movie industry. Agents may also want to glean some insight into the competition for acting roles, as well as assess the value of their own clients.

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, runtimes, proprietary review scores, review counts, genres, casts, crews, budgets, information on whether movies belong to a collection, and revenues. For actor data, they have dates of birth, locations of birth, gender, proprietary popularity scores, and filmographies.

Solution

The unique idea of this project is assigning numerical quantities to actors. These quantities can not be inferred from a quick scan of their acting credits. Most projects in this domain focus on predicting successful movies. Instead, I will aggregate the movie revenues by each actor in each film, then assign that value back to the actor. This actor centric value will be the target variable of a regression model search. The goal of this project is to find the best fit linear regression model for predicting these actor values. Weighing these values with the expense of hiring any particular actor will help movie investors find their best fit actor.

Deliverable

The final product will be available through a Github repository. It will contain a paper with all of the relevant analysis, detailed with proper visualizations and a compelling data story. To accompany this paper, I will provide a slide deck and the companion code that produced the results, as well.

Data Acquisition

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.

When I first started collecting data for this capstone, I pulled many datasets from different sources, such as Internet Movie Database (IMDb), Kaggle, Movie Lens (Group Lens website), and Opus Data (The Numbers movie website). For this capstone to succeed, I would need financial data for a lot of movies. This data is difficult to find, as most production companies are very secretive about their financial numbers. Metadata for acting roles would be useful for having features that distinguish different people, as well. I searched through these sources with

these items in mind. The datasets from IMDb and Movie Lens didn't have useful data for this project. The Opus Data datasets didn't have the proper keys that could be used to join their data with other data sources.

While the IMDb website has a lot of financial information about movies, most of it is only viewable to subscribers of their premium service. I wasn't sure if being a subscriber would allow my API key to request this data for download. So, I put this option on the back burner as a last resort. There is a python module (imdbapi.py) that can be used to access the data through an API. I spent some time learning how to use this module in case I needed more financial data for my final dataset.

I found three datasets on Kaggle that I felt I should inspect to see if any of them would be useful for financial data. The movie_metadata.csv dataset contained over 3,800 values for both budget and gross revenue. It also contained the same number of Facebook like values for the top three billing roles for each movie. The tmdb_5000_movie.csv dataset had over 4,800 values for both budget and revenue. There was no useful metadata for actors.

The the_movies.csv dataset looked really good at first. It has over 7,000 values for budget and revenue. I thought that I could use this one as my starting dataset, but I soon ran into troubles. The data was collected by a Kaggle user through an API, but before the values were stored as a JSON file, they were converted into strings. Through this, the data was in a form that pandas JSON methods could not accept as arguments. Ultimately, I was able to get around this, but then I began to question the accuracy of the data. The first red flag was that the dataset contained rows whose IMDb IDs were not unique. There were 27 pairs of observations that had the same unique identifier, but some of those pairs did not contain duplicate values throughout their remaining features, as would be the case if just the IMDb ID was mislabeled. Either one or both of the rows of these pairs were mislabeled data. I made a few API requests to compare the website data with these rows in hopes of simply dropping the ones that held incorrect data. It turned out that some of the duplicate pairs were such that neither observation in the pair had the correct data. At this point, I questioned the accuracy of the entire dataset. I could either test the data by sampling it and comparing those samples to requests that I would have to make from a website or simply build my own dataset. As the data was available from the TMDb website with an unlimited request rate, I chose the latter.

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.

Data Wrangling

After concatenating the data, I wrote a program to get it into a form such that it was generalizable to a variety of analysis methods, including some beyond the scope of this project. For instance, there was a lot of text data in the form of biography, reviews, taglines, and overviews. These could be analyzed with NLP algorithms to look for correlations that may reveal movie themes or public opinion about an actor that point to which people would have the most success as profit generators. There was data on full cast members in the movies, as well, which could be included in a graph for network analysis.

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 two 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.

After the initial preparations to get the data into a generalized form, I began to shape the two datasets to suit the scope of this particular project. Beginning with the movie dataset, I noticed that there were some monetary errors that needed to be addressed. Both the budget and revenue features had values that were not scaled properly. For instance, some figures that were supposed to be expressed as millions of dollars were carelessly recorded as shorthand values, i.e. as $75 instead of $75,000,000. There were similar errors with the 1,000s of dollars scale, as well. I attempted to verify the correct figures through other websites, but there were some movies for which financial data was unavailable. After some analysis of the monetary data, I concluded that I only needed to remove the movies where this error occurred in either the budget or revenue feature, but not where it was found in both. This was due to the high correlation between the two variables. If both variables were scaled incorrectly, that scale was the same, and removing these observations would not be necessary. Therefore, I removed any such instance of this recording oversight by filtering out movies with only one monetary value less than $1000. It should be noted that similar errors were found in the birthdays of the actors. I removed those observations, as well.

After cleaning them, 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 from different eras on equal footing for comparison. I decided to keep the high end monetary outliers, because this project aims to analyze monetary values, and I wanted to observe the maximum range of what has been achieved for those figures. As for the need to handle low end monetary outliers, their range was defined when they were eliminated along with the recording errors found at values below $1,000.

I decided to eliminate two genres from the data that I felt were out of scope for this project. These were the documentary and television movie genres. I felt that very few people had ever been hired for their acting chops in the history of documentary films. Also, the monetary dynamics of television would not be translatable to the movie world, in my opinion.

For the runtime variable, I did find one observation that ran for over 5 hours. After doing a bit of research on the movie, I came to find out that it was considered a television mini-series by some, even though it had been shown in its entirety on the big screen. I decided to eliminate this movie, as it began its life on television.

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 retained as one of the few features directly linked to the actors, as opposed to the many others that were values derived from the actors' movies. Through the use of Pandas explode and JSON normalize methods, I was able to featurize these two sets of values from the cast lists.

Thankfully, the actor dataset required less wrangling. The first thing that I did was to extract the actors from the production team, as I would not be using directors, producers, etc. After I had the actor data separated, I turned my attention to the place of birth variable, which was not recorded in an easy to parse standardized form. Some of the fields were filled with only the name of a country, while others had the county, state, and city data. I decided to extract some quick value from this feature by giving a binary encoding to actors dependent on whether or not they were born in the United States. Those from the USA were given values of one, while all others were given values of zero.

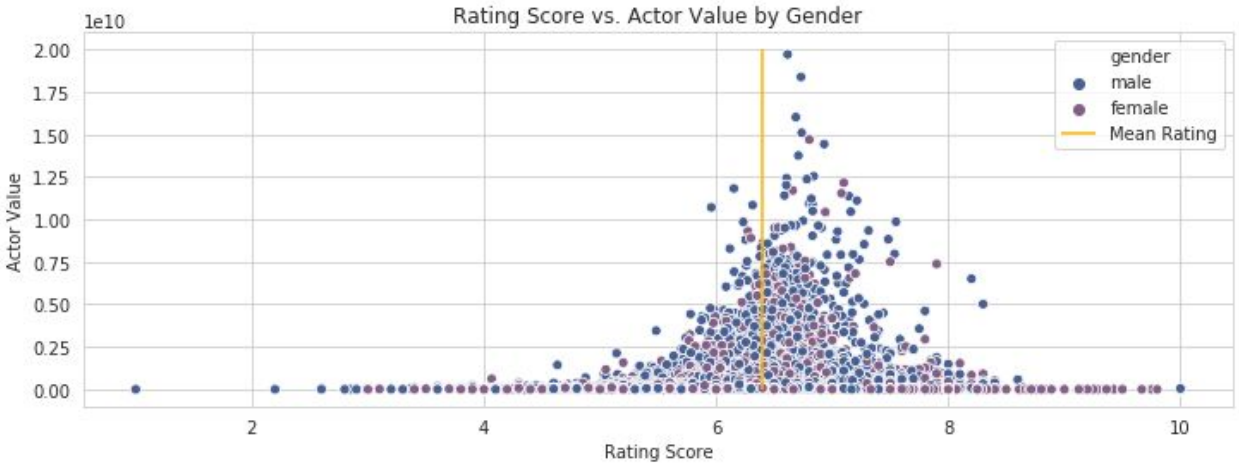
After merging the two datasets, I created an age feature derived from the release date of the movie data and the birthday feature of the actor data. Now, I had a feature whose values were

the ages of the actors at the times of their movie release dates. Then, I converted the birthday feature into one that only held the actors' birth years.

Finally, the dataset was converted to being actor centric through a Pandas groupby call. This call aggregated each of the variables by each actor, turning movie attributes into actor qualities. At the end of the process, I was left with 11,641 observations, 27 predictor variables, and one target variable. The target represented the sum total of all the revenues from all of the movies that feature each actor. The data was now ready for EDA, modeling, and prediction.

Data Story

I began to visually analyze the data by searching for patterns in the actors' revenue histories and movie ratings with respect to their ages on the dates their movies were released.



First, it could be seen that the actors who had large revenues were mostly in movies that had above average viewer ratings. This suggested a positive correlation between actors that consistently were in movies that had large ticket sales and the ratings those movies received. Also, most of the actors with high total revenues were male actors.

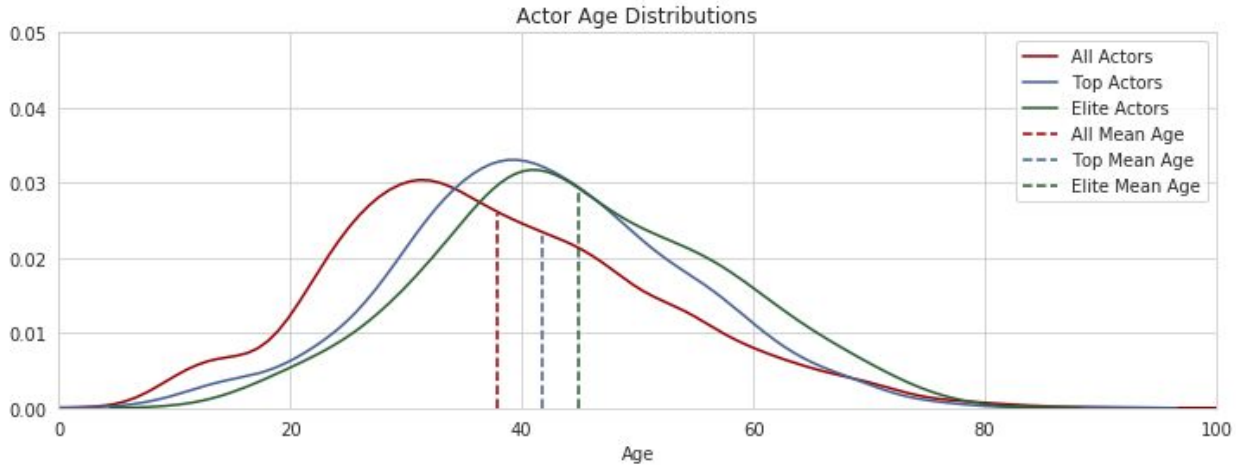
Next, I chose to group the actors by different tiers with respect to their earning histories. I decided to extract the top 10% and the elite 1% of the actors and to compare them to all of the other actors to identify any trends.

First, I looked for differences between these tiers with respect to the sum of the budgets of their movies.

The distributions showed that as the average movie budget totals grew, the elite actors came to dominate the observations. While this may be attributed to a positive correlation between movie budgets and revenues, it was interesting to observe how the elite tier of actors covered the whole range of budget totals.

On the other hand, the range of budget totals ended abruptly when observing all actors. This would suggest that the movies which featured elite actors were not just limited to movies with the largest budgets. I noted how the elite actor distribution remained horizontal throughout the entire lower range of the budget totals.

Next, I observed the distribution of the actor tiers with respect to the age of the actors on the release dates of their movies.

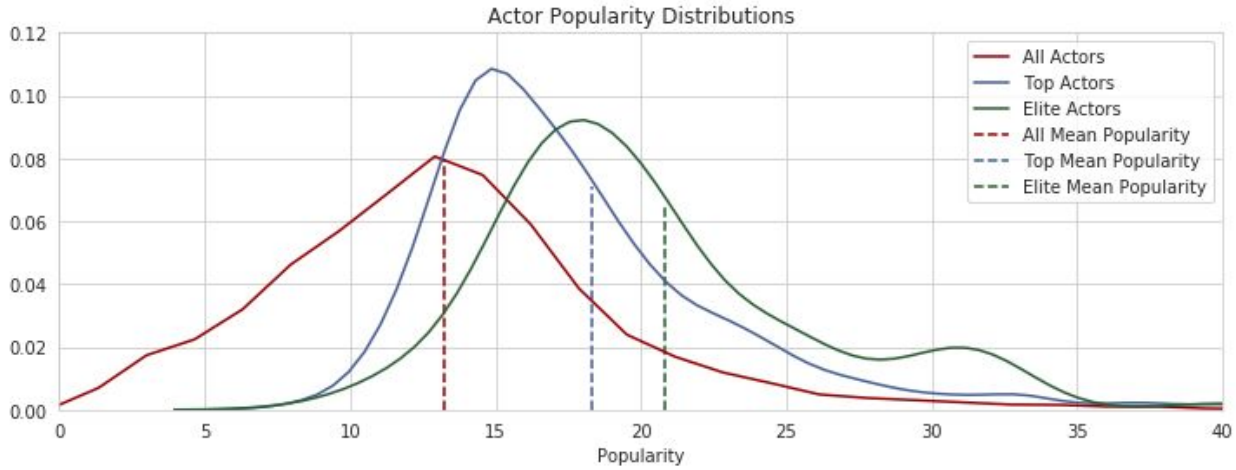


As shown in the plot, the average actors' ages increased as the actors moved up in the tiers. This would be expected, as the tiers were ranked by revenues aggregated over the entire career of the actors. Older actors would have been in more movies, which would only add to this sum.

The next variable that I observed was that of the average billing order spot for each actor. The billing orders represented the actors' rankings in the credits of their movies. Order zero, better known as the lead, represented the top billing of the movie.

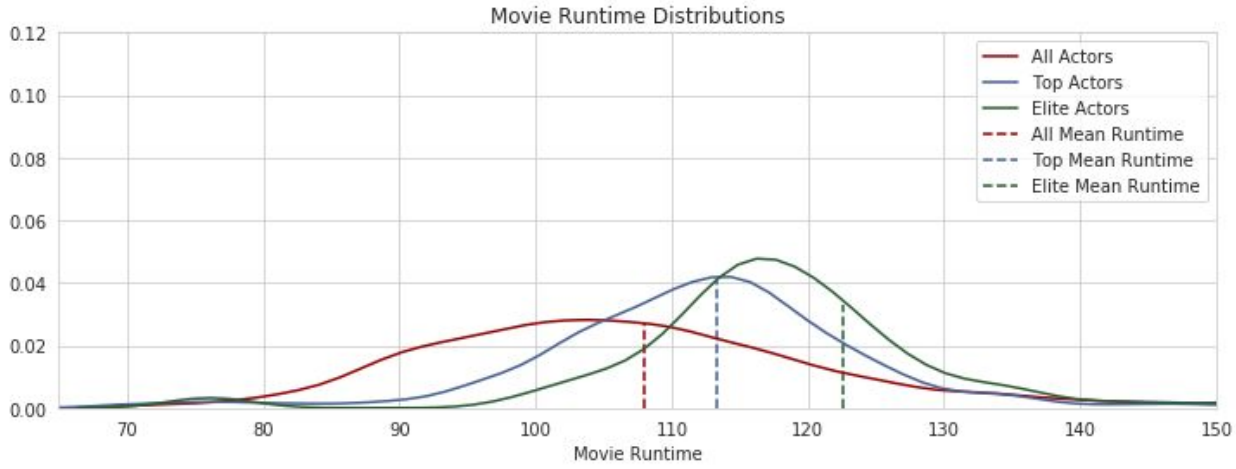
The elite actors tended to have the starring roles. This gave support to the idea of star power. This idea would assume that revenues could be increased by hiring star actors for leading roles. This distribution clearly showed that actors whose movies had the highest revenues held the top billing spots in their movies.

The next variable that I investigated was the actor's average popularity score. This was a TBDb proprietary ranking based on website traffic related to each particular actor.



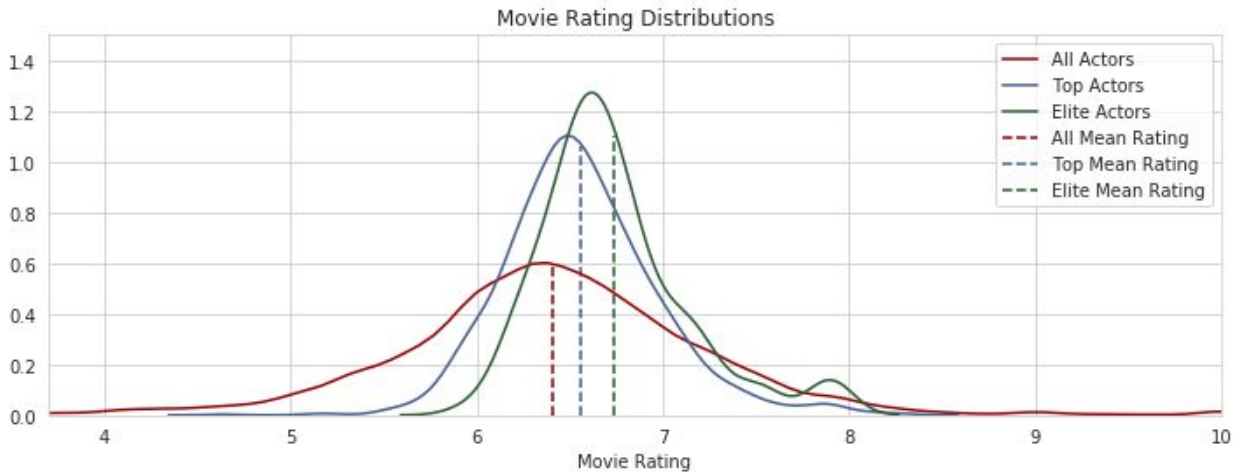
The popularity of an actor rose as the actor's movie revenue totals increased. A second mode of the distribution appeared at the higher end of the popularity scores for elite actors. Was there a super elite group of actors waiting to be discovered? This mystery would have to wait for another day.

The next observations took place on the average runtimes of the actors' movies. Runtimes had been limited to 70 minutes by design.



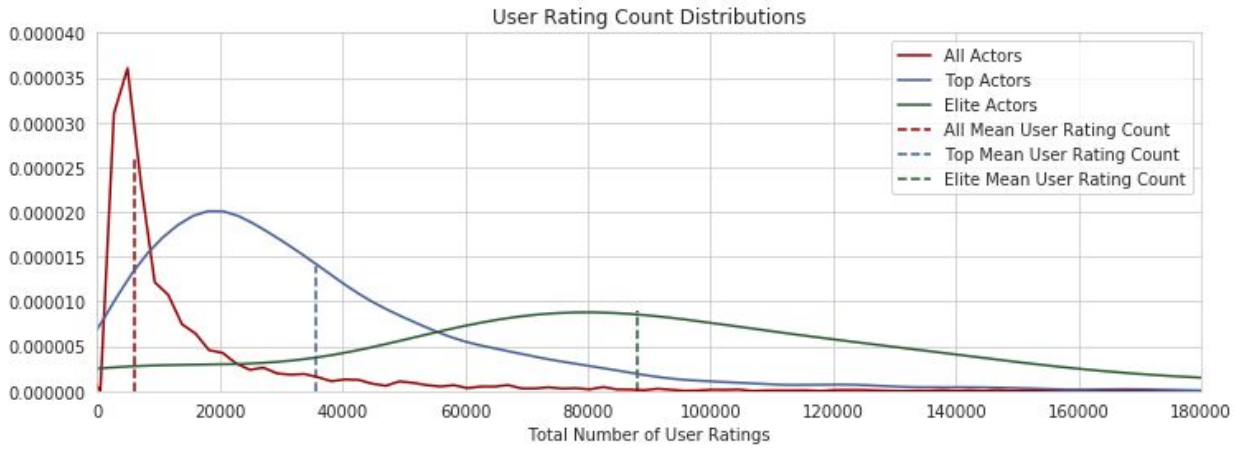
The average runtimes tended to increase with the revenue totals of the actors' movies. I speculated that since budgets had already been shown to positively correlate with revenues, runtimes should as well. This assumed that a longer movie would cost more to make, which seemed logical.

Next I observed the distribution of the average ratings for the actors' movies. These were all ratings conducted on the TMDb website.



These distributions were very similar to the ones observed for the popularity rating. It should be remembered that these ratings were given to the movies, while the popularity ratings were given to the actors. This strengthened the star power theory, by showing a positive correlation between viewer sentiment for a film and sentiment for the stars in those films. The elite distribution seen in this plot had a similar second mode that was observed in the actors' popularity distribution. Was there such a thing as elite star power? The mystery deepened.

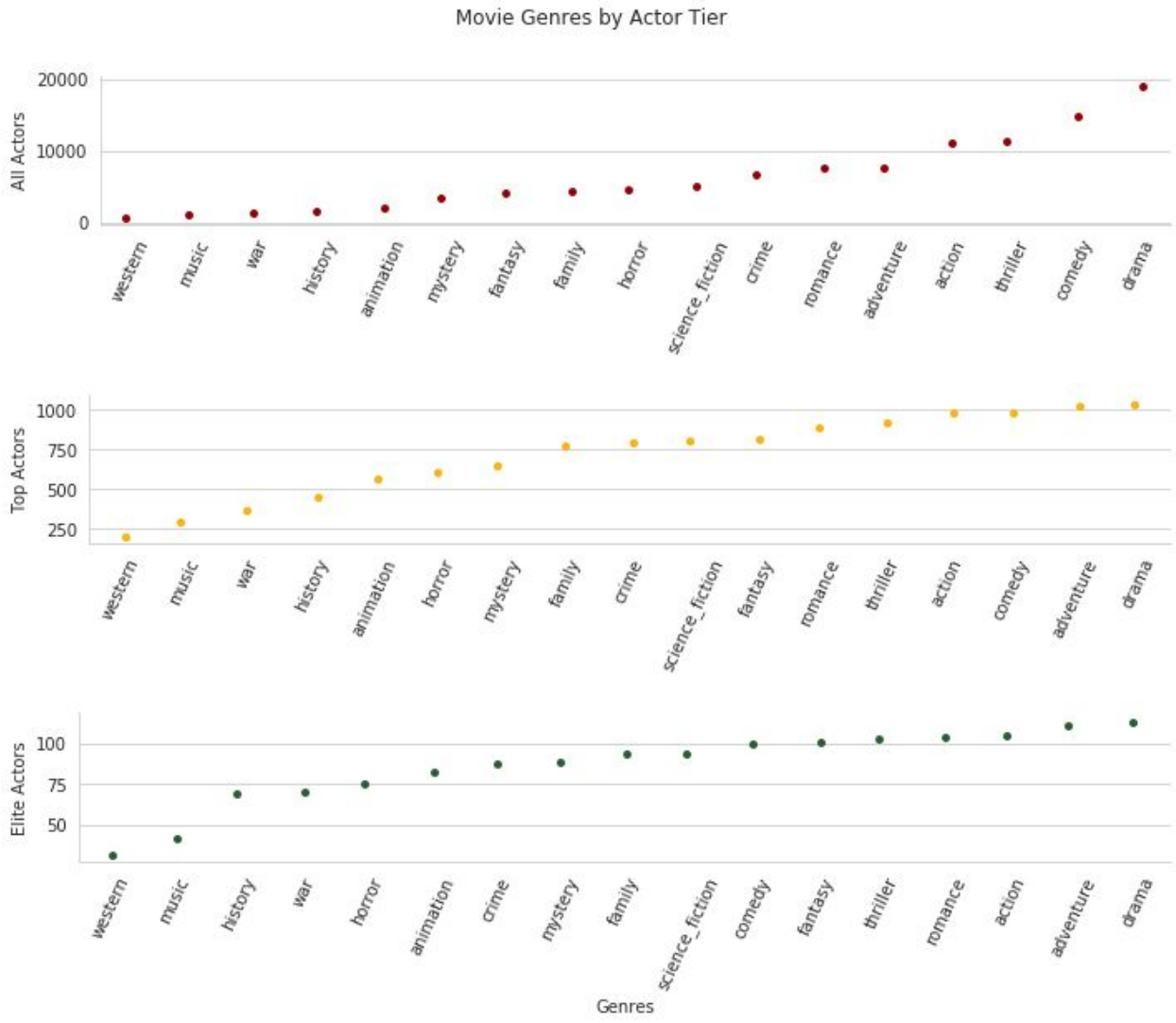
The final distribution that I examined was that of the rating counts. These were the total number of ratings received by all of the movies for each actor. This feature could be thought of as website rating activity related to each actor, regardless of the type of rating the movies received.



These plots appear very similar to the distributions of the total budgets. Even when an actor's movies had low traffic with respect to ratings, the movies of elite actors were still able to show high revenue figures. As higher budget movies would be expected to have bigger promotional

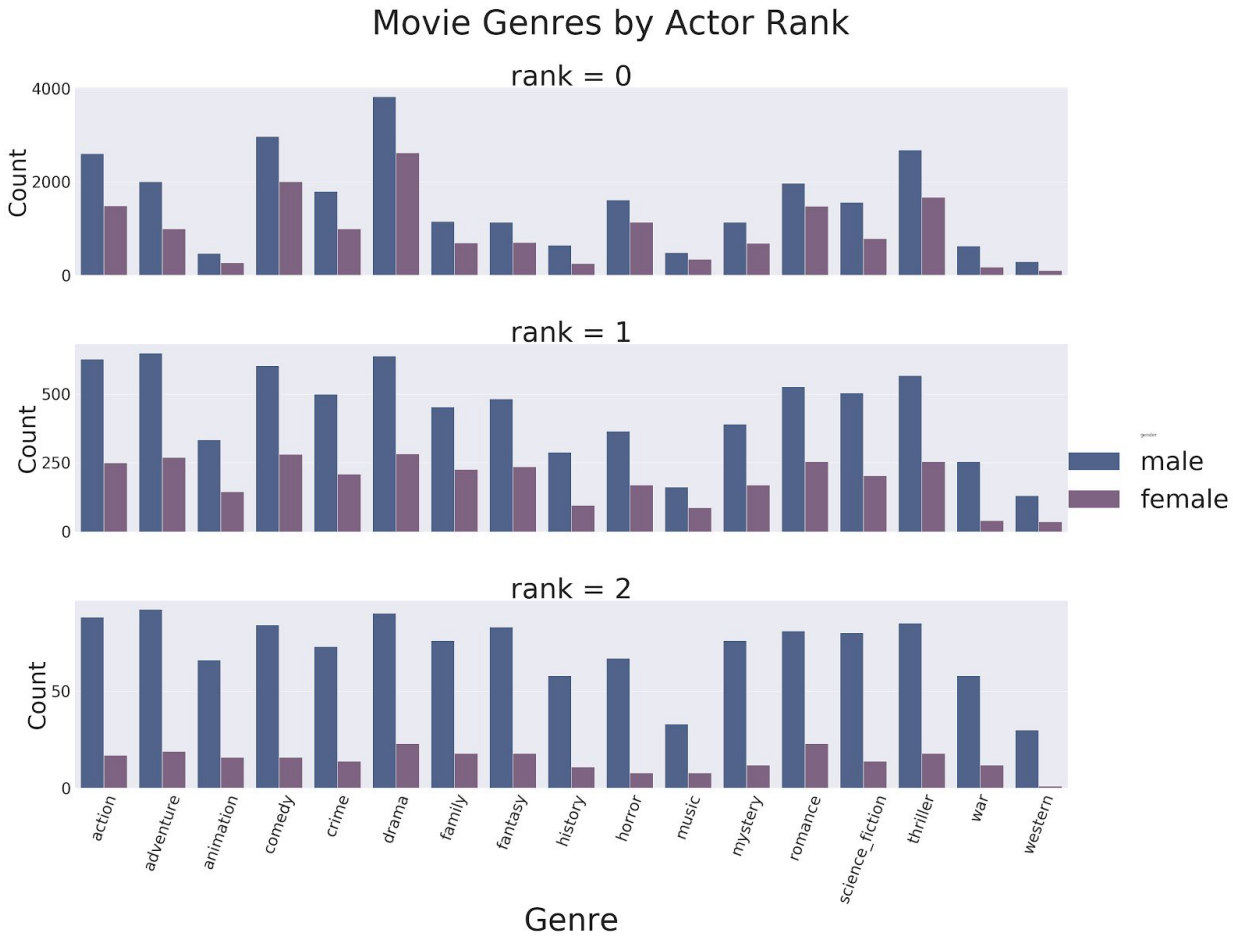
campaigns, it would make sense that these actors would be recognizable enough to moviegoers to drive ticket sales without substantial media promotion.

Next, I observed how the movie genres were distributed among the different tiers of actors.



Dramas were consistently the best genre when it came to tapping greater revenue potential. As the focus proceeded up the ranks to the elite tier of actors, the comedy genre declined in importance. Also at the top tier of actors, adventure movies comprised a larger portion of the genres than they did in the other tiers. Western, music, war, and history genres stayed near the bottom group of earners throughout the tiers.

Finally, I ranked the actors according to their tiers. Rank 2 was for the elite 1%, rank 1 was for the top 10%, and all other actors were assigned to rank 0. Then, I plotted the genres by actor ranks and split the distributions by gender.

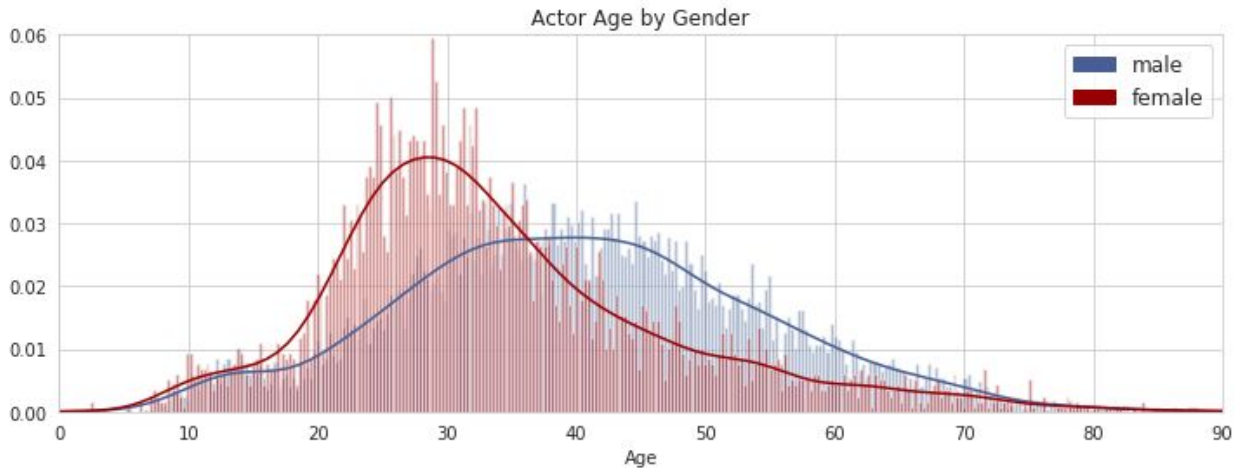


All of the genres were distributed fairly evenly among the elite actors. Rank 0 actors tended to have their roles in the action, comedy, drama, and thriller movies. By the time the actors were filtered down to rank 2, it was quite apparent that the female actors were sparsely represented across all genres. In fact, male actors comprised almost 80% of that entire group. While the female actors accounted for over 40% of the observations in this dataset, they made up only 20% of the actors who were ranked in the top 1% with respect to total movie revenue.

Statistical Analysis

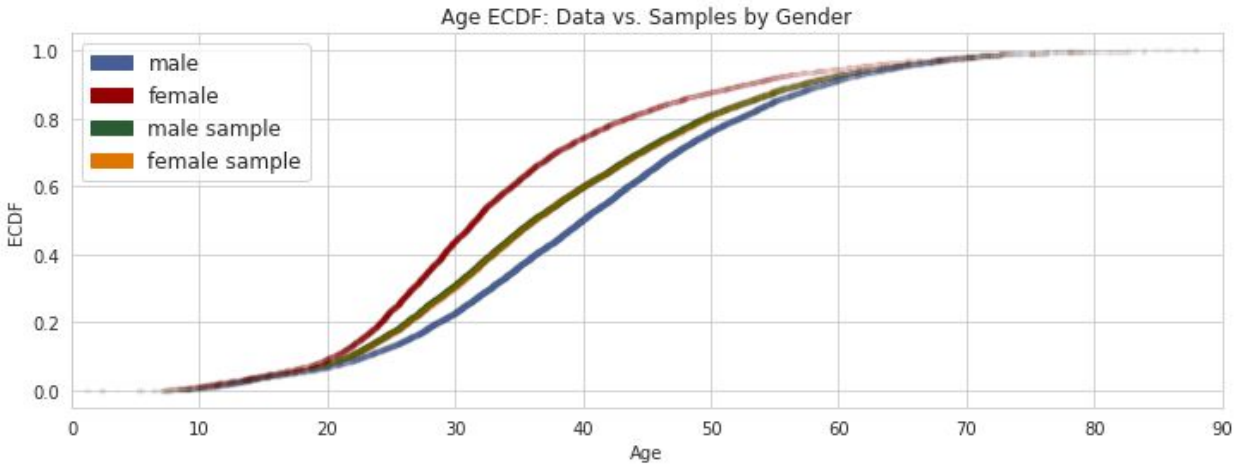
For the statistical analysis of the dataset for this project, I used several different techniques to perform hypothesis testing. I compared distributions of the mean between independent variables and between the target variable and one of the predictors. I applied hypothesis testing to evaluate a correlation coefficient, as well.

The first hypothesis test was performed to examine the difference between the distribution of ages with respect to male and female actors. After separating the data by gender, I observed their distributions.



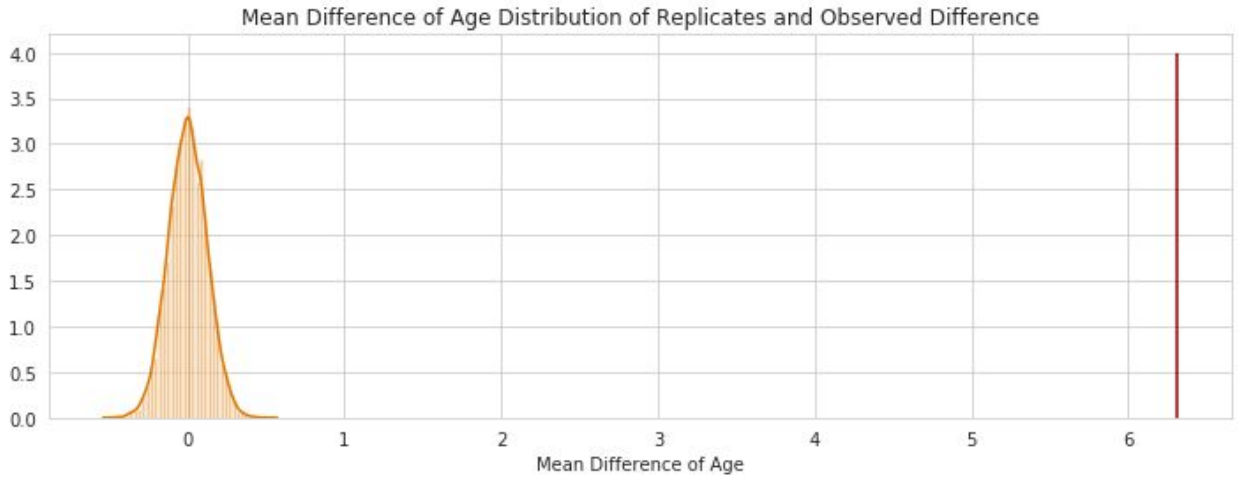
Observationally, it appeared that the average ages of female actors was younger than that of the male actors at the times of the actors' movie releases. Also, there seemed to be a short plateau for the teenage actors, indicating a lack of movie appearances for their age group. This plateau was more apparent for the male actors.

Next, I sought to break the structure of the individual distributions by combining them and reconstructing new samples of each gender through a random selection of the combined data. This selection was done without replacement, ensuring that each actor would be represented in one of the new datasets, while no actors would be observed more than once. Then, I wanted to see the likelihood of observing the statistical similarities between the original data and the new null distributions. First, I generated 10,000 permutation samples of each set that were randomly taken from the combined data. These new samples were of the same sizes as the original samples, respectively. Then, I plotted the Empirical Cumulative Distribution Functions (ECDF) for each and compared them to those of the observed data.



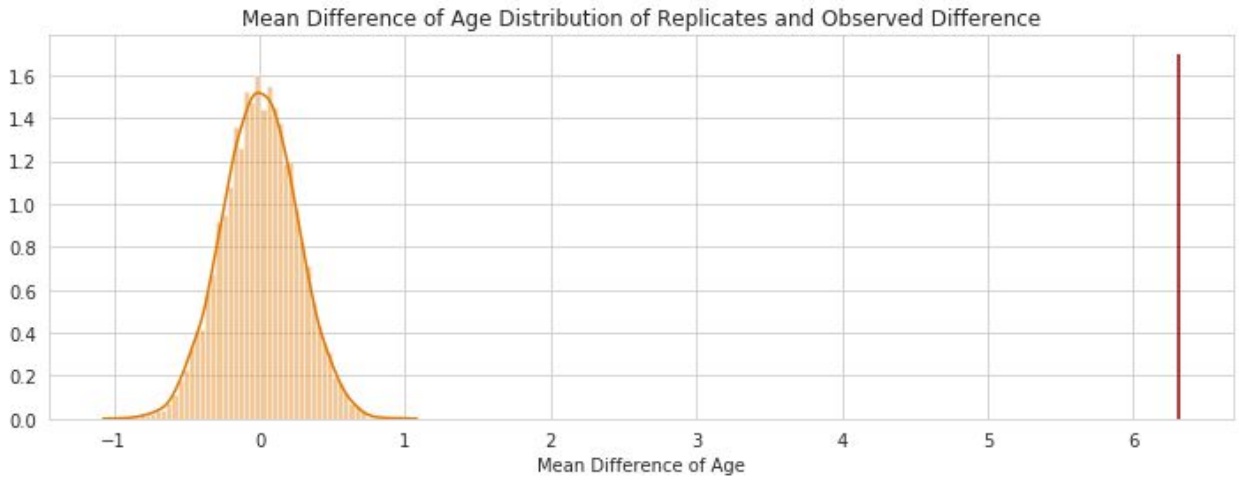
The curves showed a distinct gap between the genders of the observed data, while the curves of the samples overlapped. This indicated that the distributions between the genders were not the same.

Next, I compared the difference of the mean age between the two genders with that of the 10,000 permutation replicates already generated. The mean age gap seen in the observed genders was 6.3 years. The distribution of the mean age difference of the randomly shuffled samples was plotted for comparison.



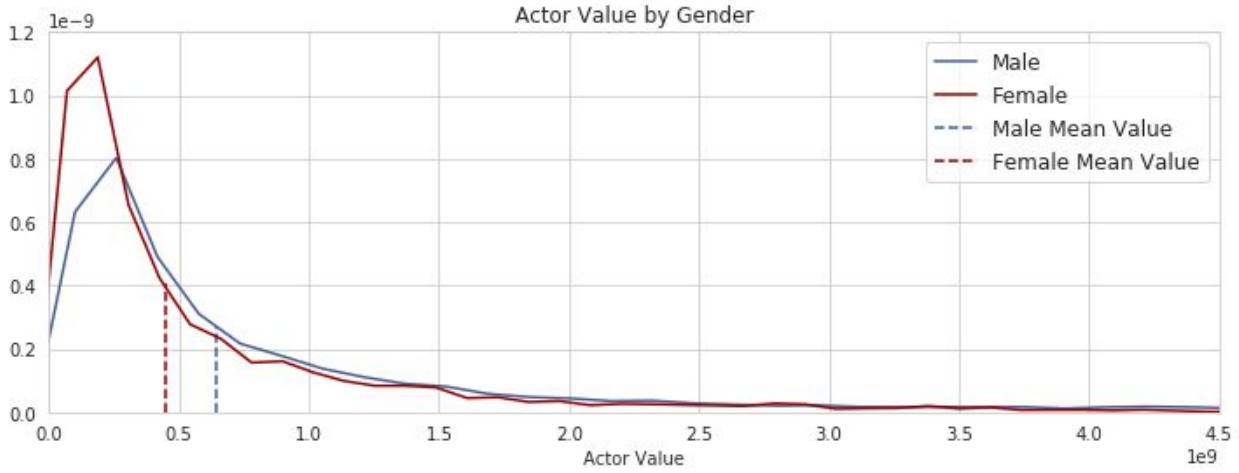
As was observed graphically, the mean age difference was found to be far beyond the distribution of values computed from the permutation samples. The p-value was calculated as zero. This meant that the probability was extremely low of observing a difference of the mean ages by chance as extreme as the one observed in the true data.

For the second hypothesis test, I examined the difference between the mean age of the two genders, without considering their distribution. First, I took the individual gender datasets and shifted the ages of each actor within them to coincide with the mean age of all of the actors. Then, I created 10,000 randomly generated bootstrap replicates of each shifted gender dataset through selection with replacement. This type of selection allowed some actors to be observed multiple times in a new sample, while completely excluding other actors. By doing this, a collection of 10,000 new datasets for each gender was simulated, without the expense of collecting actual data. Then, I calculated the difference of the mean ages of the simulated datasets to compare their distribution to the mean of the original, unshifted data.



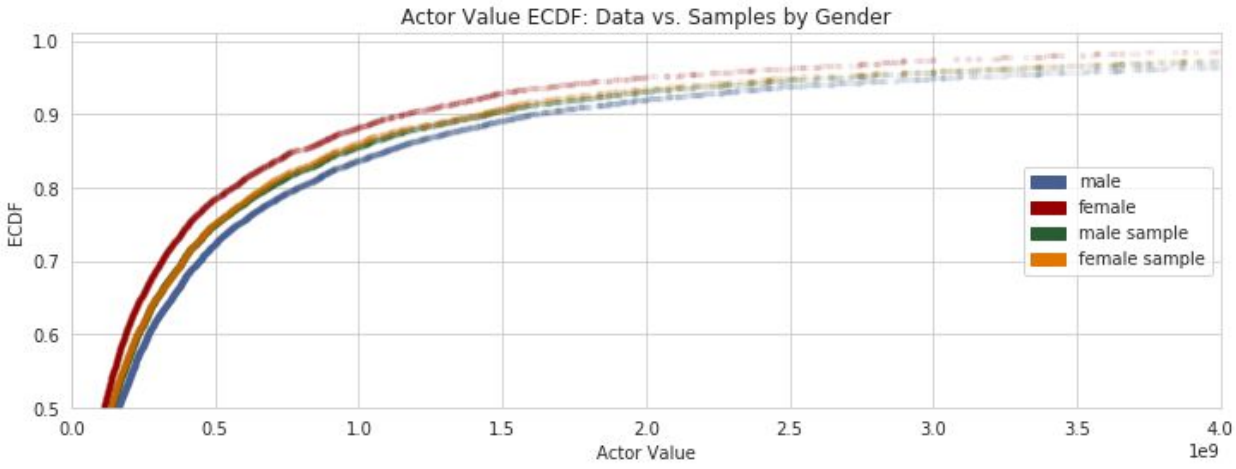
As would be expected, the variance of the difference between the new sample distributions was greater than it was when samples were created with replacement. Yet, the mean difference of age from the true data was still far away from the mean of the new distribution. The p-value was given as zero, once again. This confirmed that regardless of whether or not the two sets came from the same distribution, it was highly unlikely that the difference in mean age between the genders was due to chance.

The next hypothesis test focused on the target variable, actor value. This target represented the total of revenue generated by all of the films over each actor's career. The goal of this test was to determine if there was a significant difference between genders with respect to the target values. First, I plotted the target variable distributions by gender.



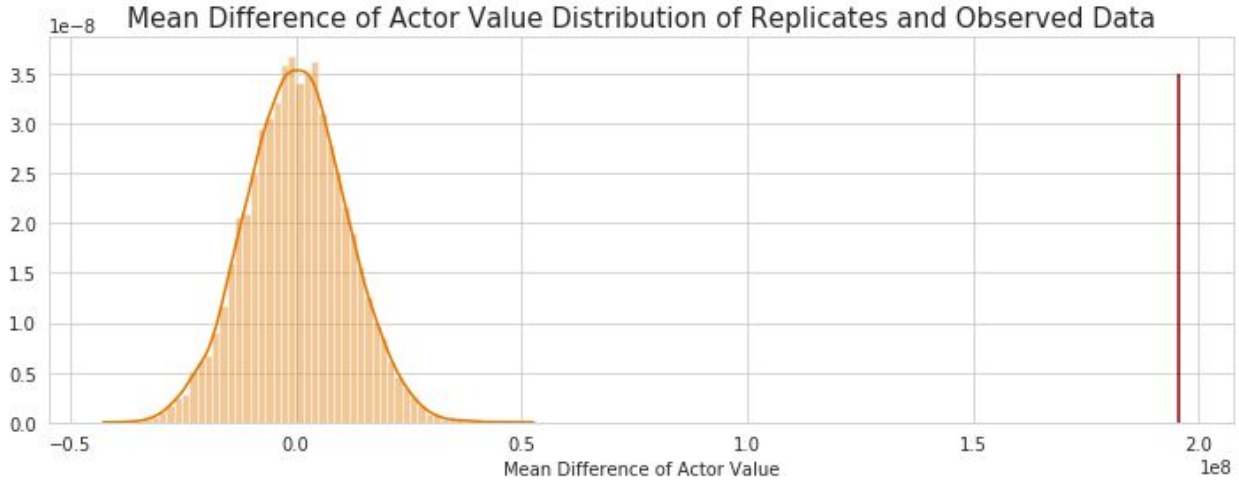
The distribution shapes appeared similar, although the female actors were clustered lower in the target values, while the distribution of the male actors peaked at a greater value.

Again, I drew 10,000 permutation samples for each gender and plotted all four ECDFs. I was looking for an obvious difference between the distribution of the original data and the distribution of the samples generated through combining and shuffling the two genders.



The two curves lined up tightly for the sample data, showing no gender bias when samples were obtained randomly. By observing the true data, it could be seen that a greater portion of the female actors were registered at lower target values than the proportion of male actors. Visually, the two original datasets appear distinctly distributed.

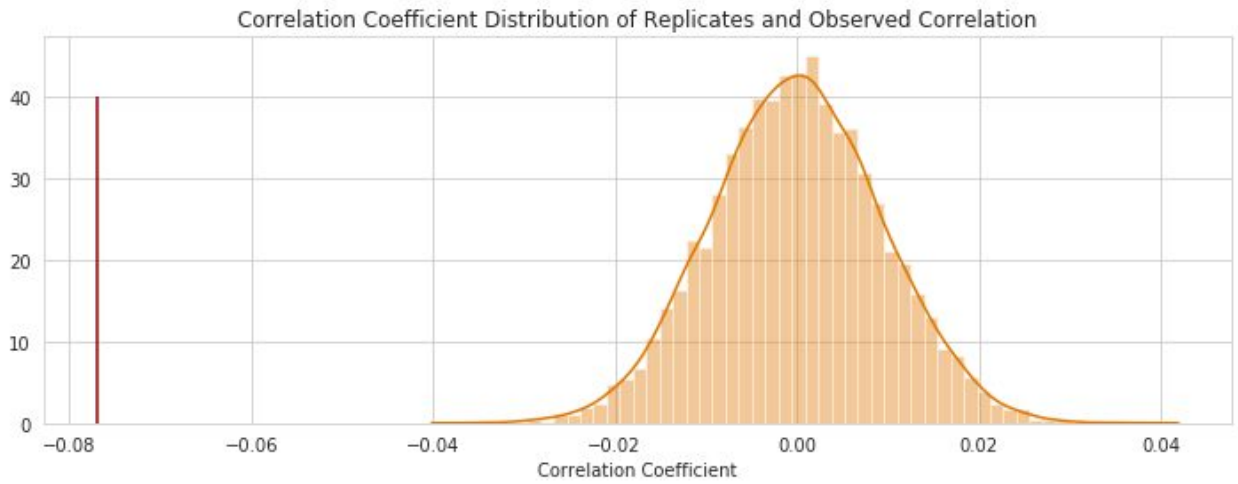
As before, I compared the distribution of the mean difference between the genders from the samples generated to the true average calculated from the data.



The true mean difference in career movie revenue was $195,749,566.18. As could be seen from the plot, this value was far from the distribution obtained from the permutation datasets. The p-value was calculated as zero, again. This meant that we could be extremely confident that

there was a real difference between the distribution of the mean target values when comparing the genders in the original data.

Finally, I applied hypothesis testing to examine the correlation coefficient between the target variable and one of the predictor variables. Once again, I chose to examine gender. I permuted the values of each gender dataset and generated 10,000 permutation replicates of their correlation coefficients. This was the same technique I used when I analyzed the mean difference in ages using bootstrap replicates that were generated by selection with replacement.



The true Pearson correlation coefficient of -0.076862 was not in the neighborhood of those found in the sample distribution. The p-value for this test was zero. This meant that we could be very confident that the actor's gender and their career movie revenues were negatively correlated.

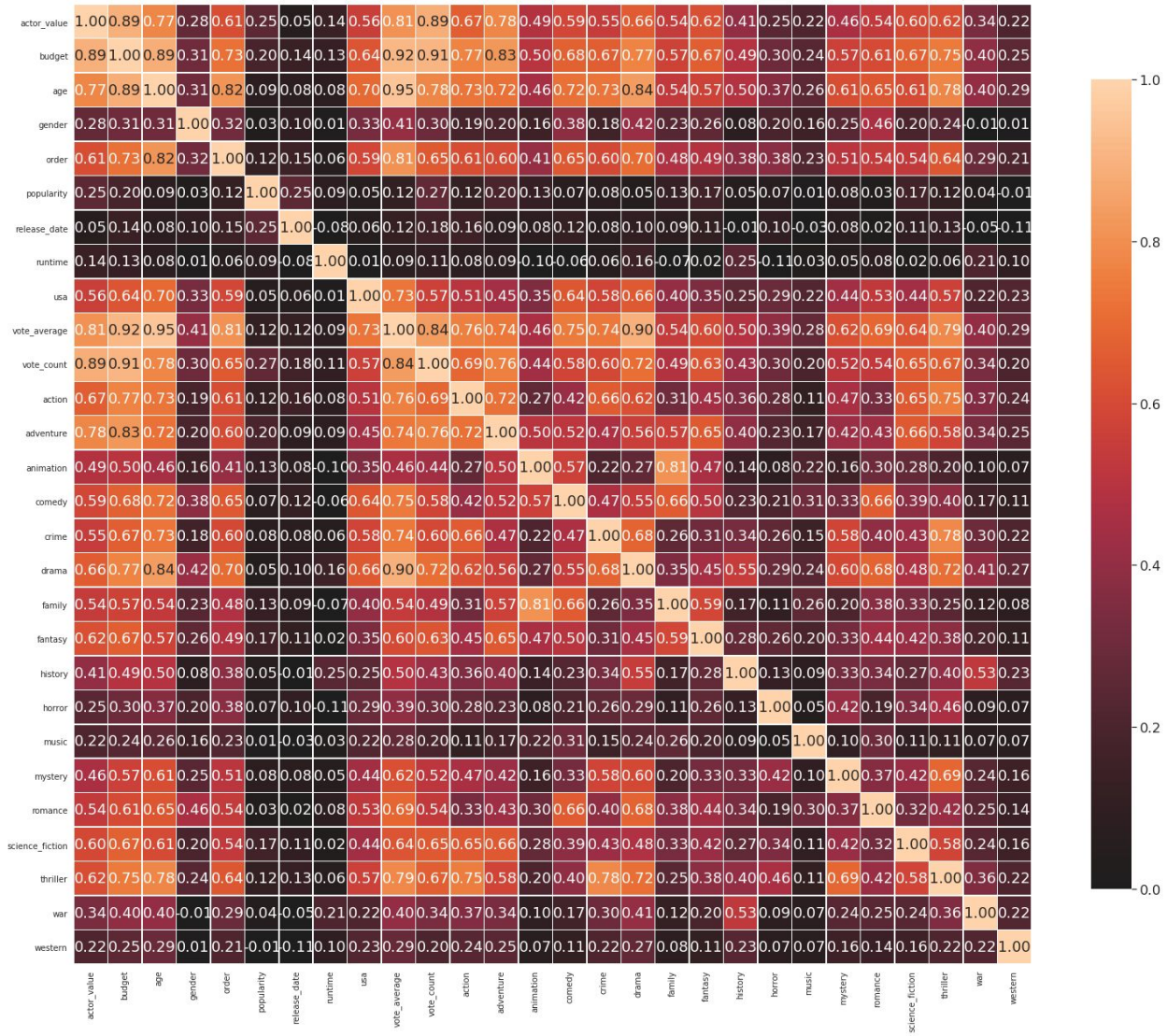
In-Depth Analysis

While most studies of cinema data have been movie focused, with this project, I attempted to investigate the same information with an actor centric view. Having requested the data for this project through the API of a movie database, most of the effort expended for building an accurate and generalizable regressor model was put into data extraction and munging and feature engineering. Several features were created from the transformation and amalgamation of the existing data, as it was received from the TMDb database. This was necessary, because most cinematic metadata available described attributes of a movie, not of an actor. For example, each movie was categorized with a combination of different genres. I sifted out the individual genres for each movie and grouped them by the actors who played in those movies. An actor feature was created for each of the 17 movie genres. The values for these features represented the sums of an actor's appearances under each genre. This would allow algorithms to select similar actors based on their appearances in common styles of film. As for the movie reviews data, average rating and total number of ratings, I converted these values into actor features in a similar manner. In addition, I used the average runtimes for the actors' films as

attributes of the actors themselves. Also, the actor birthdates were aligned with the movie release dates to give values for the ages of the actors on the days the movies came out. Finally, monetary features were created using the revenue and budget numbers of the movies. The resulting features represented the total dollar amounts that were associated with the films which comprised each actor's lifetime work. The monetary feature associated with the movies' revenues became the target variable. Needless to say, this type of feature engineering came with a bit of uncertainty compared to the simple extraction of toy datasets that could have been easily obtained from movie data repositories such as Kaggle, IMDb, TMDb, Rotten Tomatoes, and Movie Lens, among others. I was determined to deliver a new spin on the old movie recommender system trope. I felt that acquiring fresh data with a unique focus that required novel features to obtain success would be well received and greatly appreciated.

The decision of limiting my available tools to only using a linear regression model to make predictions on this dataset defined the goal for this analysis. I had to find the best way to represent my data in order to optimize the chosen model. Linear regression models tended to be less robust to messy data than other regressors, such as random forests and gradient boosters. To choose linear regression over other regression models required diving deeper into the dataset to understand the relation of the target variable to each predictor.

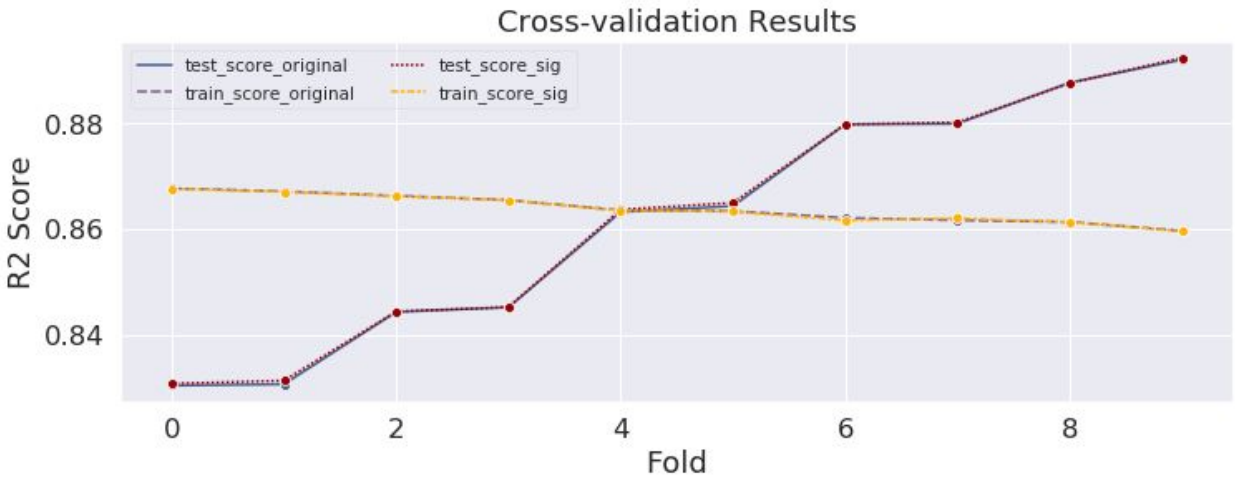
First, I observed the Pearson correlation coefficients between the dependent and each independent variable. I felt that this would give me a good starting point on how to proceed with any data manipulation that may have been required to optimize the model for more accurate predictions.



After observing the many strong correlations in the plot, I felt good about my chances of success with this model. Budget, age, vote average, and vote count all had very strong correlation coefficients with respect to the dependent variable. I decided to proceed with predicting on an unmodified dataset and make any further decisions after analyzing the results.

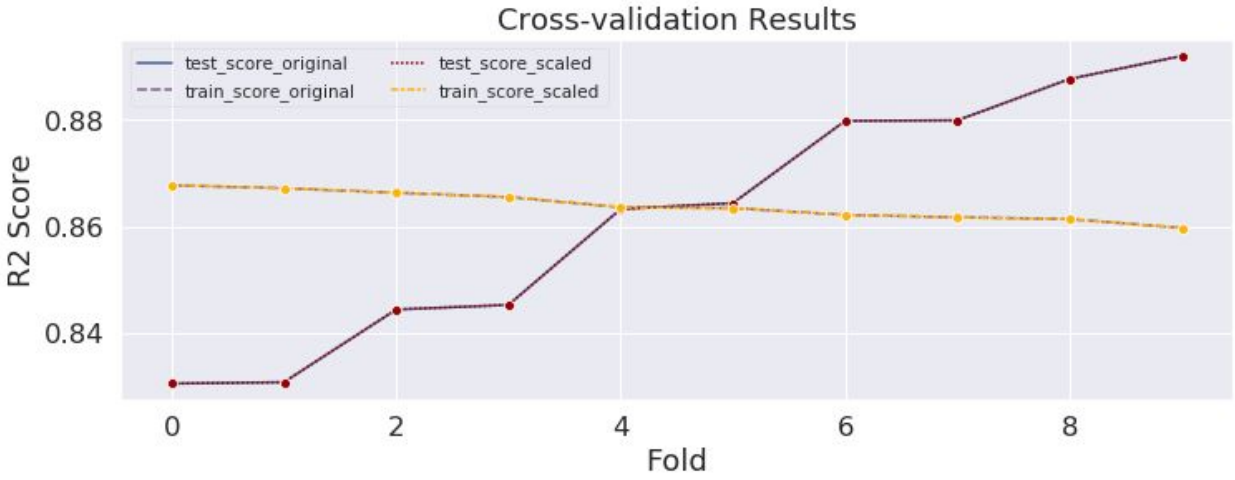
First, I split the data into training and final validation sets to guard against data leakage contributing to model overfitting. To my great delight, the model scored 0.864 for its coefficient of determination on the training data. To be reassured that this score wasn't merely the result of a favorable random split, I ran the model through a 10 fold cross-validation analysis to observe the range of scores that would result from many different random splits. The scores ranged from 0.8304 to 0.8921, with an average score of 0.8618. It appeared that the result from the first split was not extreme.

To see if I could improve on this score, I decided to look for features that the Statsmodels OLS summary report indicated were not significant as predictors of the target variable. These features were all movie genres. There were four of them: action, horror, music, and war. After removing these features, I ran the model through another cross-validation comparison. The scores from this model ranged from 0.8308 to 0.8924, with an average of 0.8621. There was a slight improvement, but not by very much. I plotted both the training and test scores from the cross-validation results on the original data and on the data with the subset of features to observe any differences.

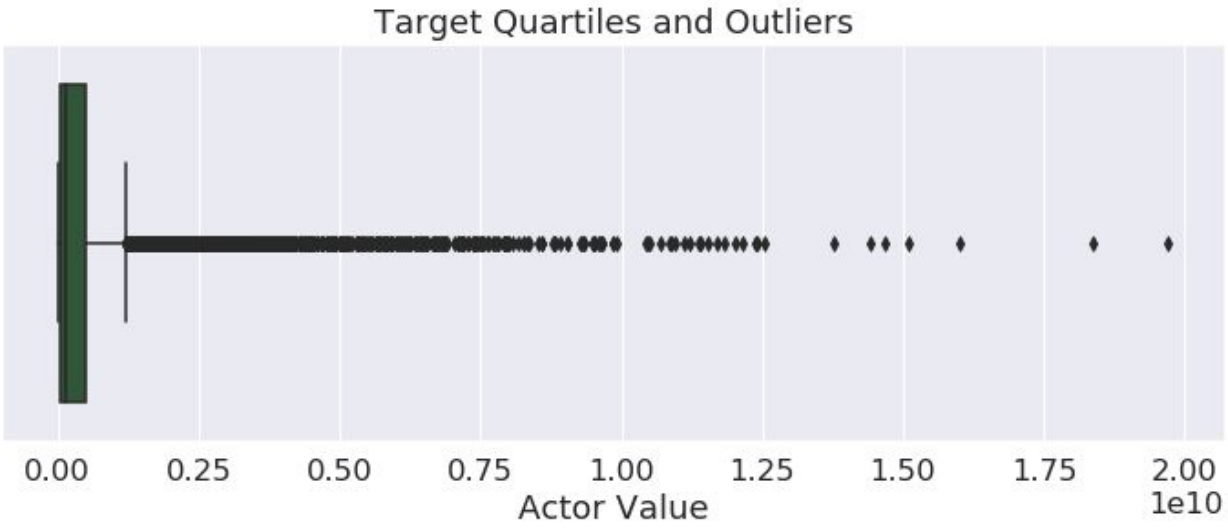


The results from each dataset overlapped so well that it was difficult to distinguish between the two. There was a slight difference that was observable at the 2nd fold.

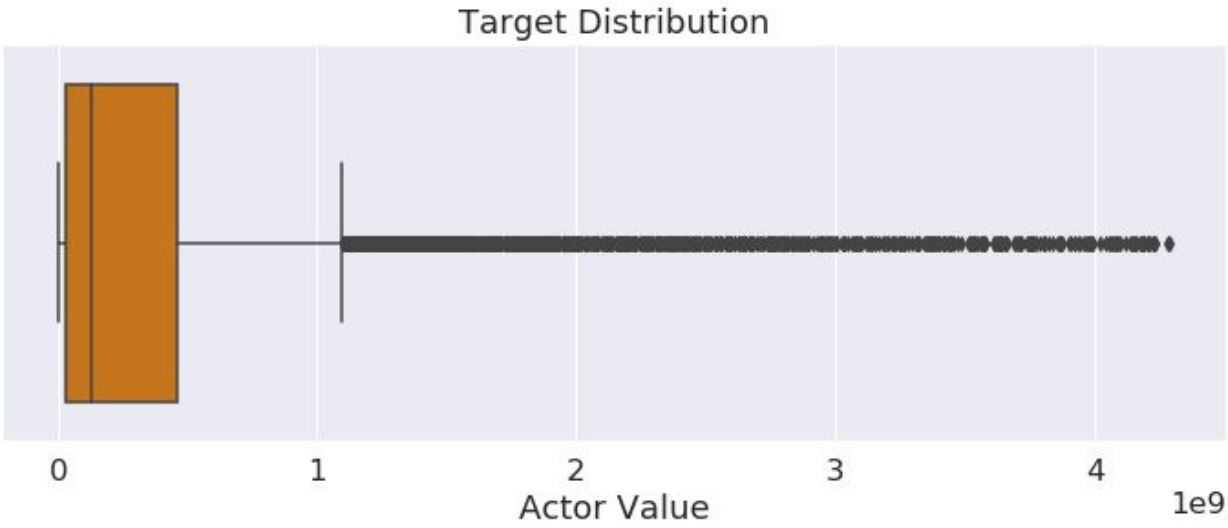
Another method that I knew was commonly used to increase linear regression model accuracy was the scaling of the predictor variables. This involved setting the mean of each feature to zero and giving each a variance of one. The results were exactly the same as they were on the original dataset. Nothing had been gained by scaling the features.



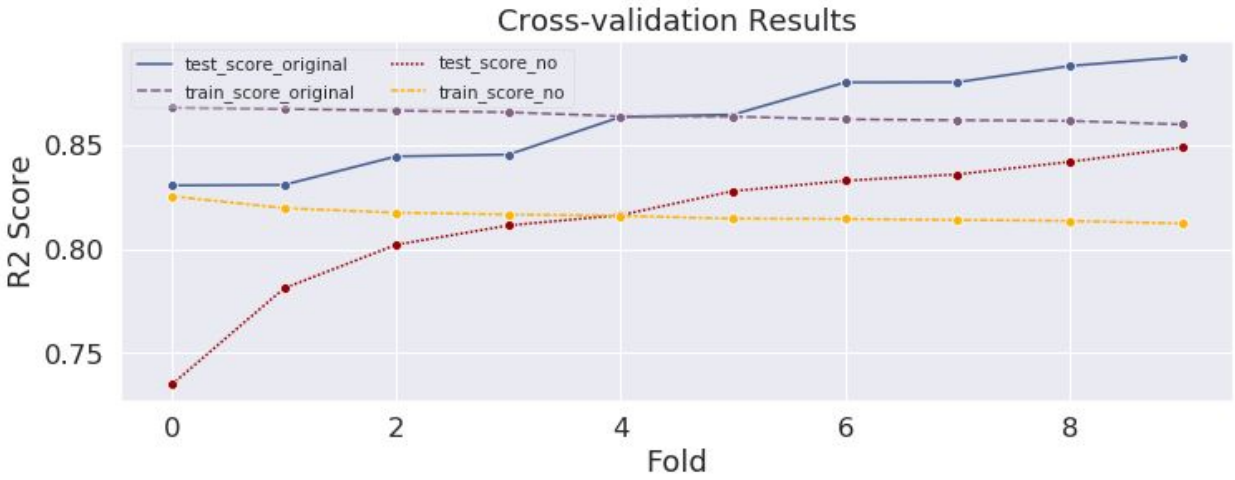
Next, I decided to take into account the possible sensitivity of this particular regressor to the many outliers of the target variable, due to the vast range of the film revenues.



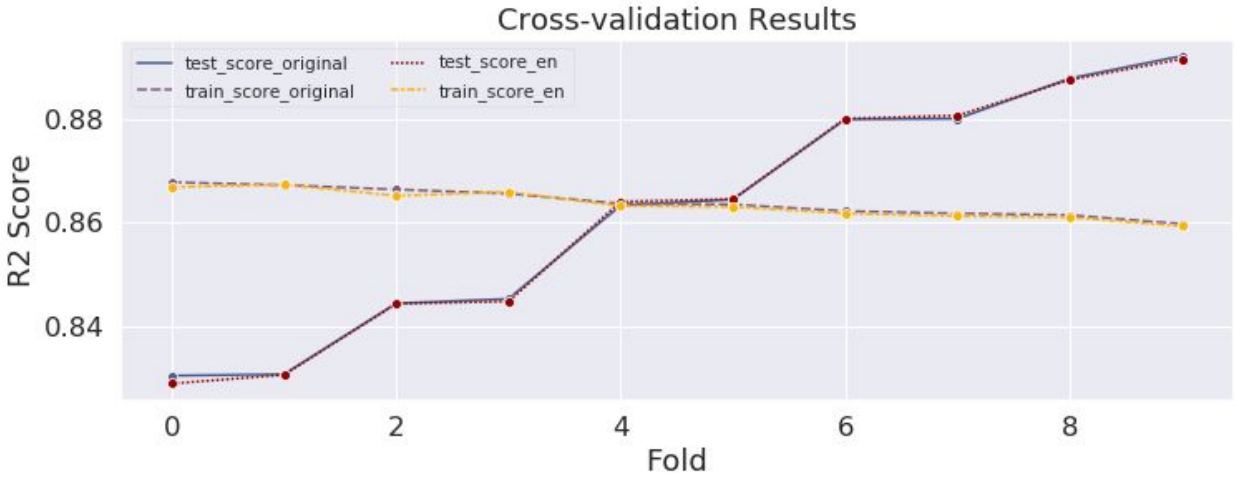
The outliers were very easy to observe in the boxplot of the target variable. I generated another boxplot after removing outliers from the target that had a z score greater than 3.



Actors who had lifetime movie revenue totals slightly over $4 billion were removed. The scores returned were not helpful in improving the model. In fact, the changes made the model less accurate. The cross-validation scores ranged from 0.7349 to 0.8486, with an average of 0.8132.

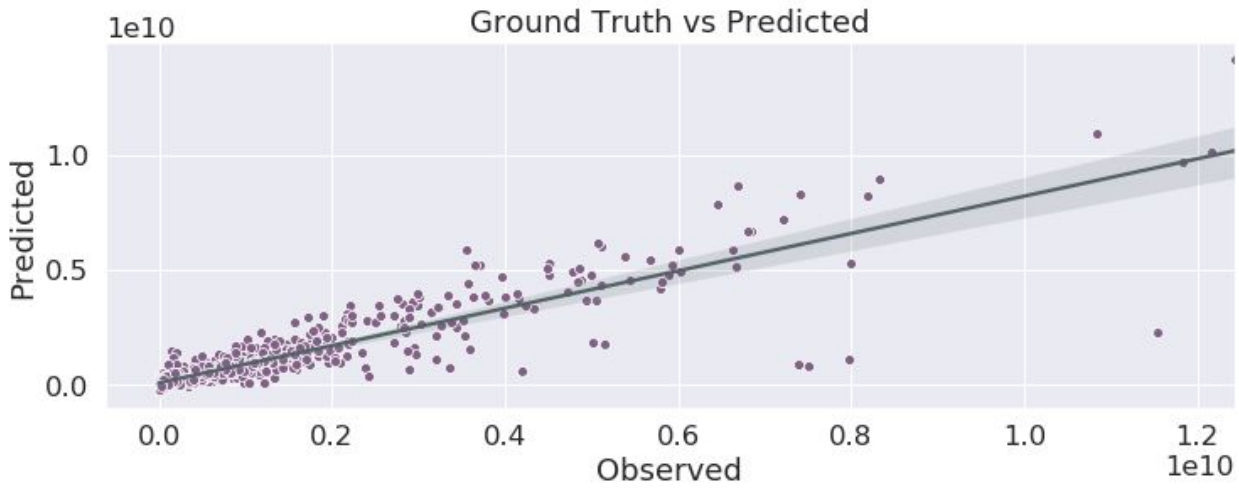


Lastly, I wanted to try using a regularization method on the model. Ridge regularization would penalize the features with large linear regression coefficients. Instead of removing them, as I did earlier, it would keep them small (L2 penalty). This would decrease model complexity, but eliminate the need to remove features. Similarly, Lasso regularization would still enforce the penalty for non-zero coefficients, but would allow for some coefficients to be exactly zeroed out (L1 penalty). I used Elastic Net, a method that combined both Ridge and Lasso regularization techniques. This method had two hyperparameters that could be tuned for model optimization, the L1 ratio and the alpha parameter. The L1 ratio would be tuned to decide how much the regularization performed like Ridge and how much it performed like Lasso. The alpha parameter was used to determine the severity of the penalization. I chose a range of 8 values to try for each hyperparameter. I used an exhaustive 10 fold cross-validation grid search on all 64 model candidates, which resulted in running 640 total fits. The best performing model's hyperparameters were an L1 ratio of 0.5 and an alpha of 0.1. The average score of the best Elastic Net model was 0.8620. It was just slightly lower than the average score of the model that had its features removed through inspection.



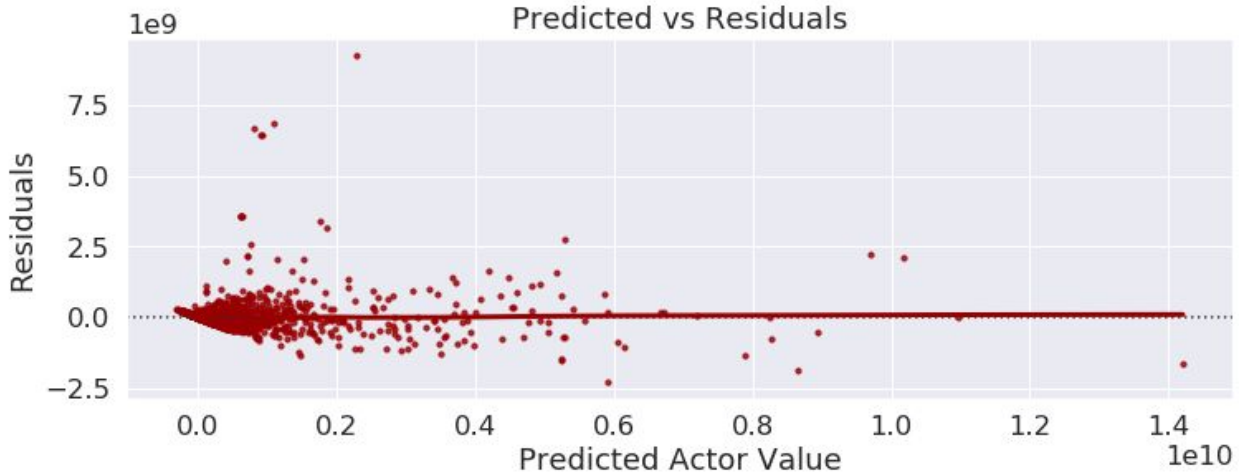
I decided to use the model which had the subset of predictors that I had chosen for the final evaluation on the test data, as it performed the best on the training data.

Before predicting on the holdout data, I wanted to observe the residuals of the best model. The training data were split once more into additional training and test sets. The residuals plotted were the errors calculated from predicting on the new training data and comparing the predictions to the new test data. The first plot I made was of the ground truth against the predictions.



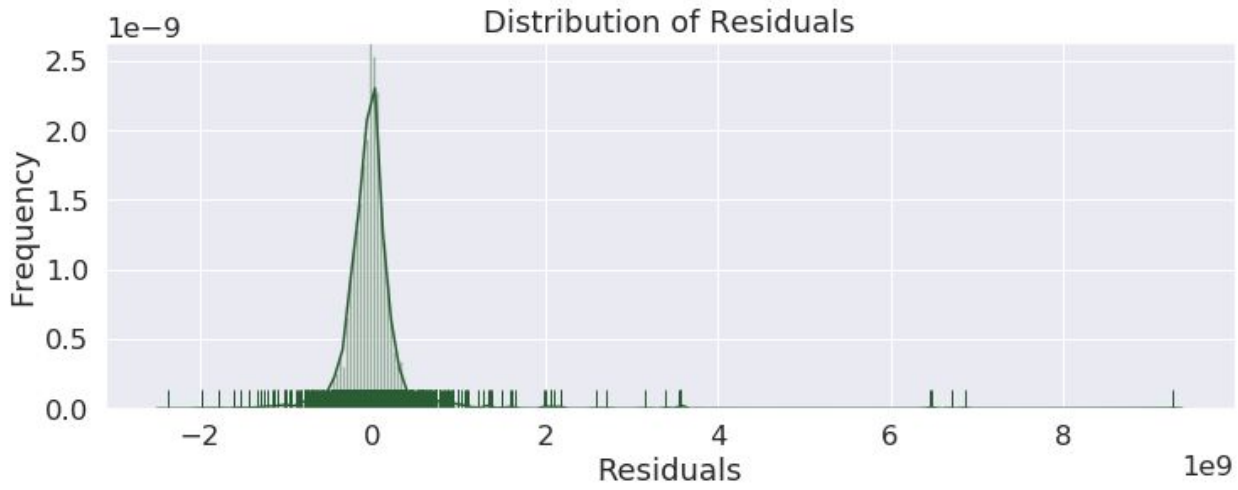
The model performed best when it was not predicting on the outliers. There were a few errors above $7 billion that pulled the fitted line away from the optimal angle of 45 degrees.

Next, I plotted the predictions against the residuals to look for any deviation away from a horizontal line.



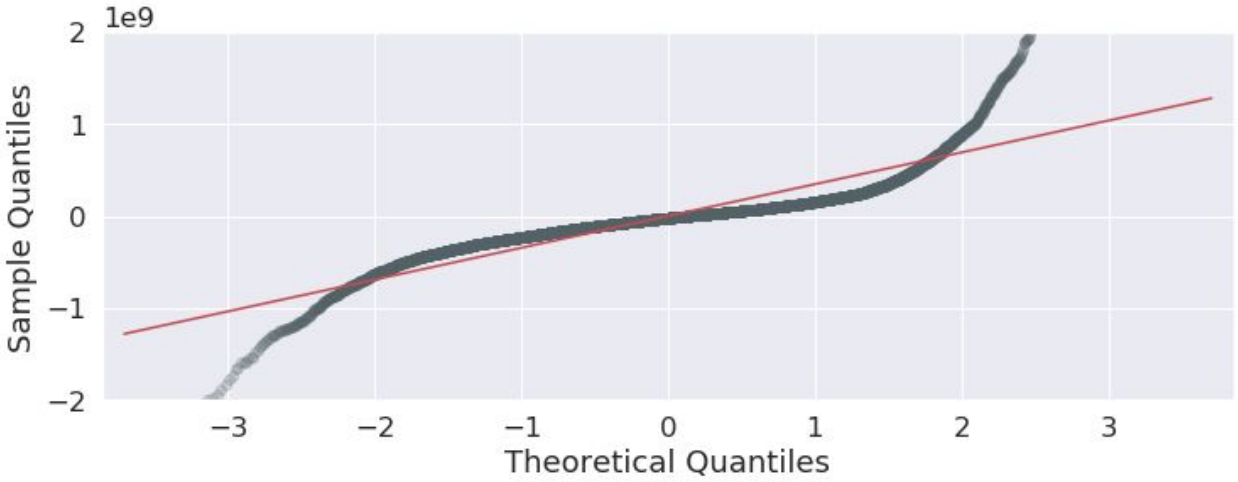
That plot looked good, as well. The fitted line was completely horizontal. This indicated that there was a strong linear relationship between the predator variables and the target variable.

Then, I plotted the distribution of the residuals to observe its form.



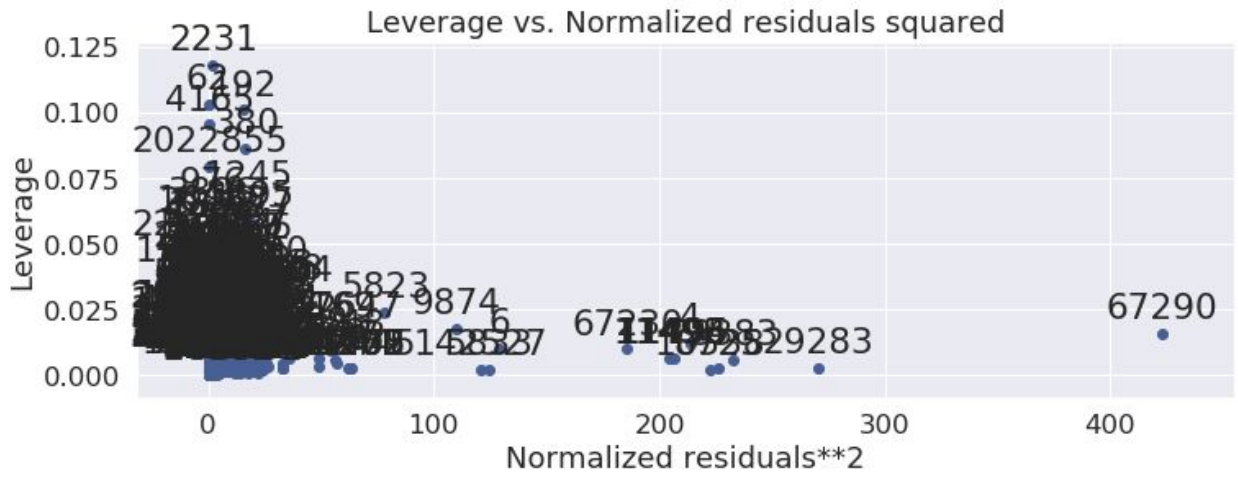
The residuals appeared normally distributed about the origin. A few large errors could be observed in the lower right corner of the plot, as well.

Next, I observed the quantile-quantile plot to look for where the residuals of the model started to deviate from a normal distribution.



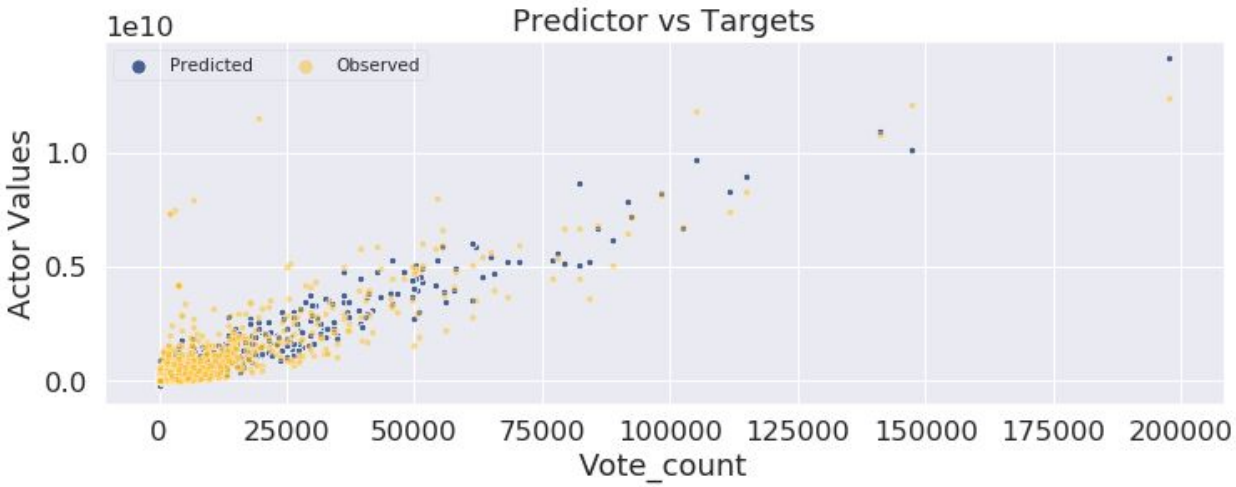
The residuals pulled away from the desired fitting just before hitting $1 billion. These errors accounted for less than 3% of the total number of residuals.

Finally, I created a leverage plot to observe any data points that had strong influence over the predictions of the model.



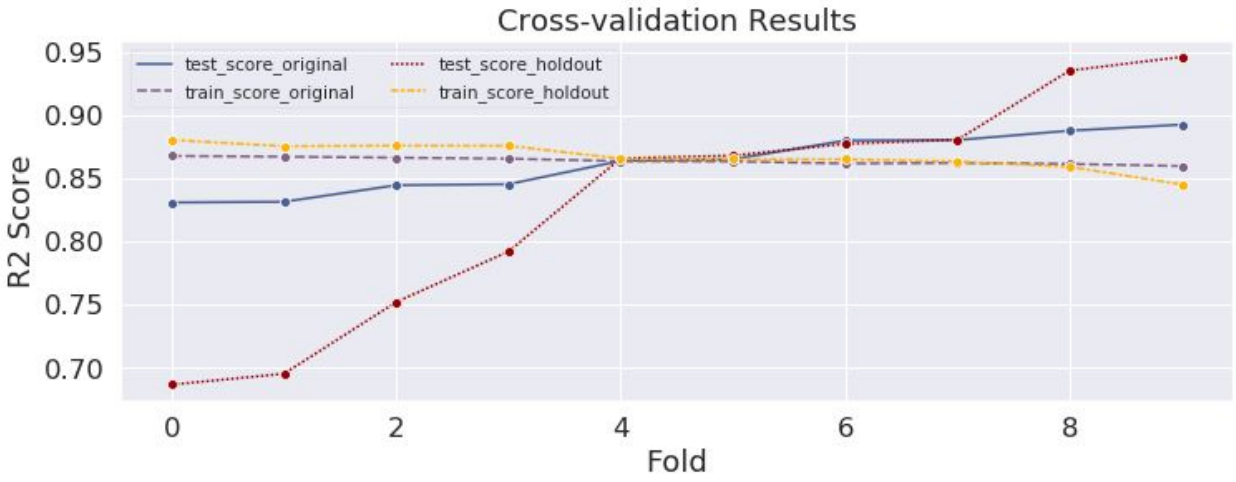
Fortunately, there were no observations that had both a large residual and a high leverage value. Those points would have appeared in the upper right corner of the plot. The actors with the highest leverages were still found near the population cluster. There was one actor who had a noticeably high residual, as could be observed in the lower right corner of the plot. After I referred to my reference dataset, I discovered that this person was Verna Felton. She was the voice of many characters in Disney movies throughout the 50's and 60's. Some of her characteristics that may have thrown off the model included her higher than average age of 66 years. This would have been especially true given her gender, as it was observed earlier that female actors tended to be over 6 years younger than their male colleagues. Another interesting note about the data related to her is that her roles were spread fairly evenly among the billing orders of her movies. Her billing order ranged from 3rd to 8th in 5 films that had over $2 billion in collective revenue. Also, she averaged as the 6th actor in the credits. Despite performing in many high revenue films, her casting did not indicate that she had the star power that was a characteristic of all of the other actors who held the largest career revenue totals.

Before I turned to the final model validation, I wanted to observe the performance of some of the strongest predictors. I plotted a handful of them against their target values as well as the predictions they made on those targets. I will only show the plot for the strongest predictor for the sake of brevity.



The predictions aligned quite nicely with the ground truth under this predictor. The next three strongest predictors showed similar performances.

Finally, after being allowed to predict on the validation data, the average cross-validation score obtained by the best model found was 0.8298.



The linear regression model that performed the best in this dataset had a training score of 0.86 and a test score of 0.83. The model accurately predicted career movie revenue of actors, and its performance was shown to be generalizable to unseen data.

Conclusion

Through the use of a simple linear regression model, I showed that accurate predictions about the value of actors could be made. These predictions could help movie producers, directors, and casting directors optimize their casting choices toward the goal of increasing movie

revenues. The numerous actor attributes created could be used to select the best fitting actors over the wide range of various types of movie productions.

While working my way through this project, I had attempted to engineer more complex features to describe the actors. The one feature that I thought would be very insightful was something I called the return on investment value. This was a value that looked at the relation between the profit seen by a movie's investors and the amount of money that they had invested in that movie. I felt that this feature would spearhead a Moneyball type of search for those "diamond in the rough" actors waiting to be discovered. Unfortunately, using these values to construct the target variable did not result in building any models that could predict accurately. Perhaps, using a multi-output regression model, instead of combining the monetary figures into one variable, would have produced useful results.

Other paths of further exploration to take with this dataset include expanding the range of regression models to include different choices such as random forests and gradient boosting trees. It would be interesting to see what various deep learning models could do with this data, as well.

This dataset is very rich, and has not been fully tapped by this analysis. There is a lot of text data that wasn't incorporated into the linear regression model. Various NLP models may be able to use sentiment analysis techniques on the critic reviews to give more insight into this problem.

Also, only the actors were used in this project. After incorporating all of the individuals involved in making a movie, I would like to see the results of putting this data into a graph for building models that add network analysis to their methods of prediction. It would be interesting to know if any effects could be observed on the chemistry created by groups of movie workers that are not observed at the individual level.