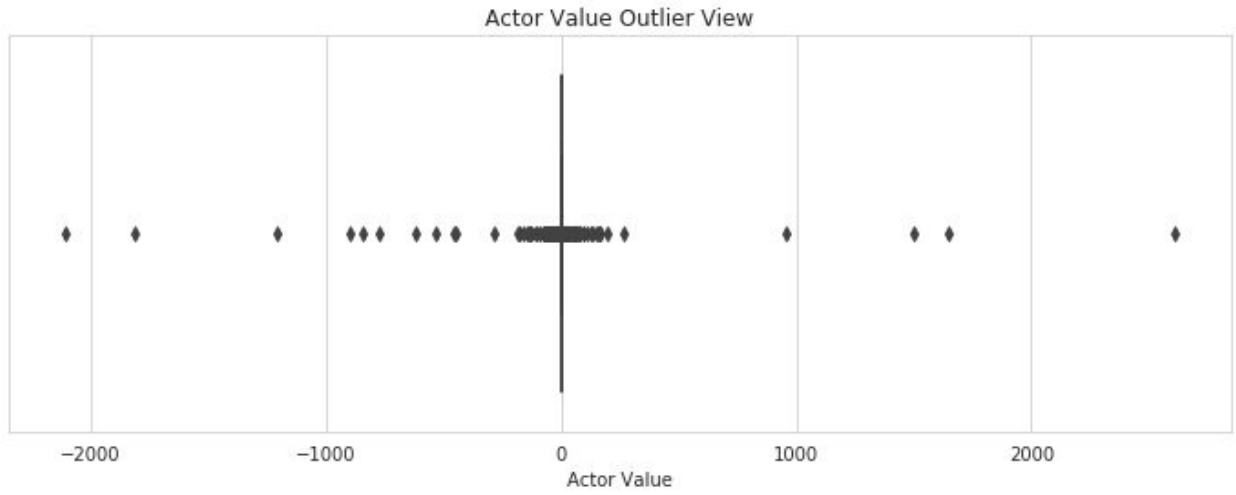
Capstone 1: In-Depth Analysis

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 needed for building a regressor model was put into data 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 an attribute of a movie, not of an actor. For example, the movie ratings were merged with the number of ratings per viewer, then converted into a true Batesian average score. After this was done, the scores were grouped by each actor and averaged. Then, 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. Each movie was categorized with a combination of different genres. I sifted out the individual genres and performed the same procedure of grouping them by actors and taking their averages, as was done with the ratings. Finally, after adjusting each for inflation, I converted the movie budget and revenue values to a ratio that represented the relationship between the amount of money invested in a movie to the amount that was either returned or lost at the end of its run. These values were assigned to the actors in the movies, while being scaled to reflect the importance of the actor to the movie, as defined by the actor's rank in the billing order. 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.

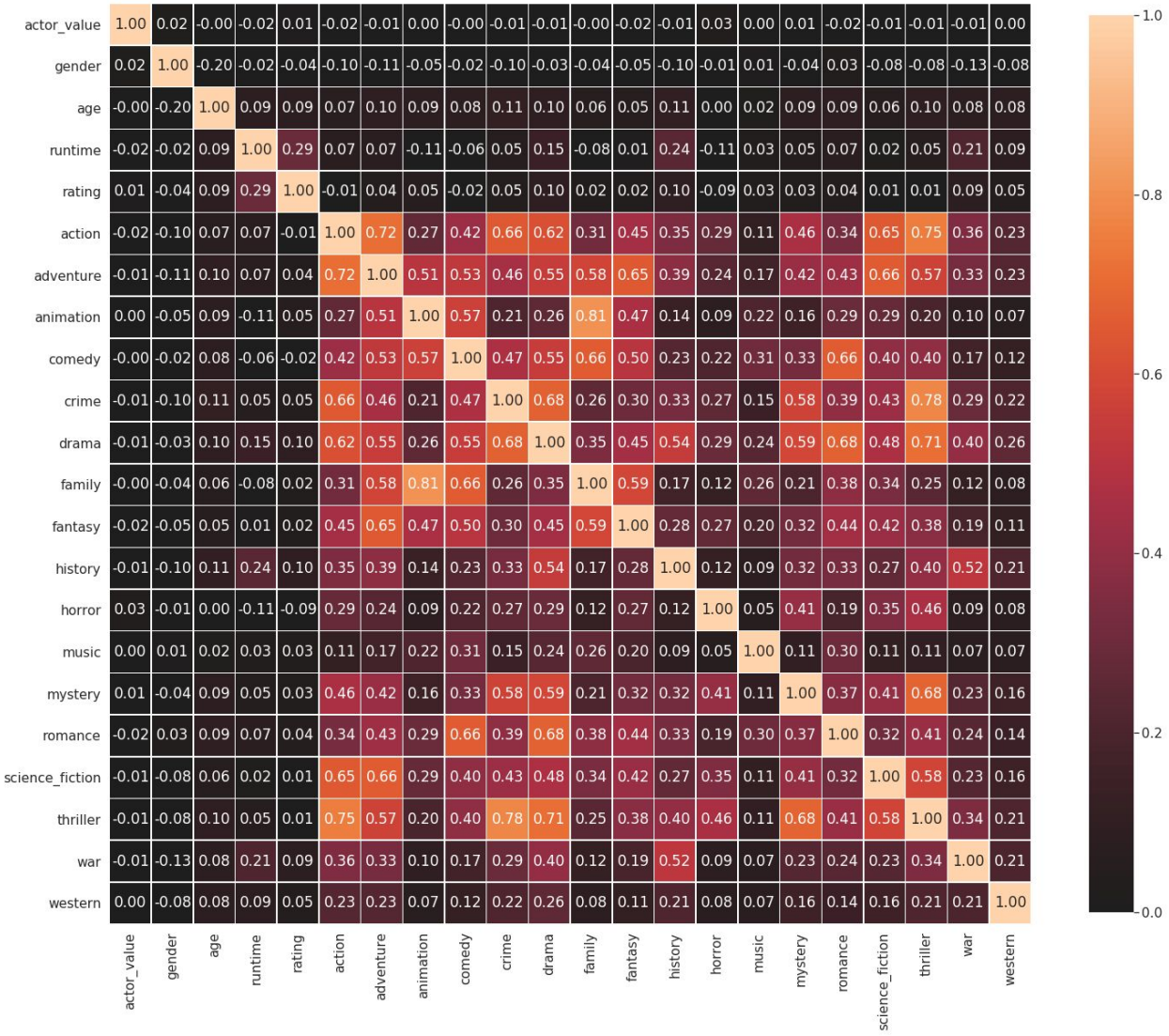
The decision of which model to try would need to take into account the sensitivity of the various regressors to the characteristics of my dataset. For one, it contained outliers from the vast ranges of film budgets and revenue.



Multiple Linear Regression would be sensitive to these outliers. Random Forest Regression would have no problem with this requirement, as feature scaling would not be required.

Some of my data was non-linear, too. This was a result of the transformation of the monetary values by using a ratio of two original variables. Support Vector Regression was considered to help with this. Multiple Linear Regression would have trouble dealing with this characteristic.

Worst of all, because of the engineered actor features, the data was noisier than I had hoped. This could be seen in the lack of correlations between the target variable and any of the predictors. I was concerned that the engineered target was complete noise.



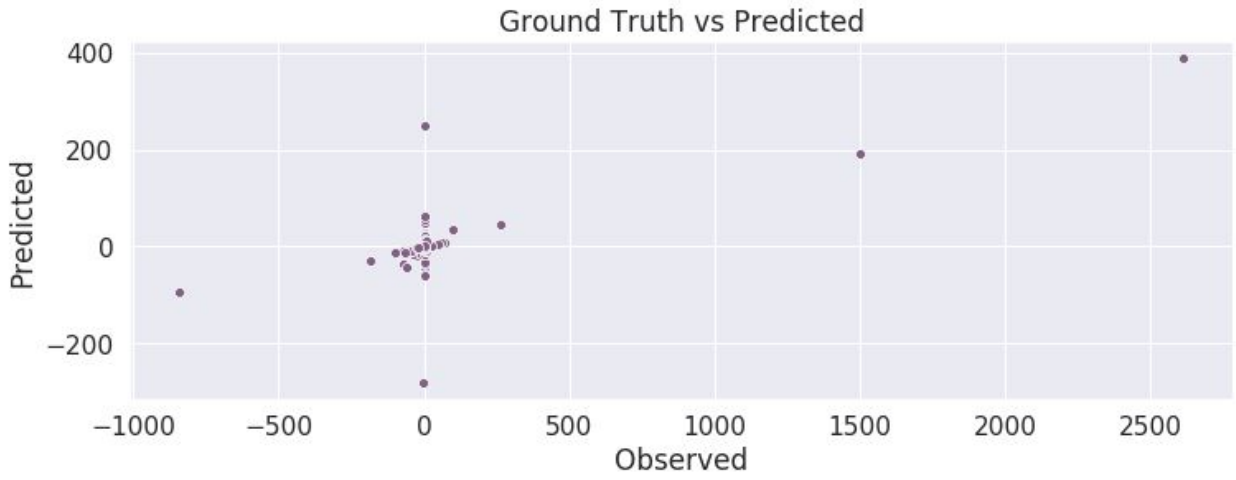
Because of this, I felt that I wouldn't have to worry about overfitting. This eliminated the need for corrections to overfitting provided by regularization models, such as Ridge, Lasso, or Elastic Net regression techniques. Also, Gradient Boosting Regression may have had trouble fitting on noisy data. A properly tuned Random Forest Regressor, on the other hand, would be less sensitive to noise. In the end, I chose the Random Forest Regressor as my model. I was

determined to ensure that enough effort was put into the tuning of the hyperparameters to obtain an accurate and generalizable model.

After observing the lack of correlations between the target variable and the independent variables, I felt that the target values needed to undergo a transformation. Even though I chose to train a Random Forest Regressor, I wanted to use the Statsmodels Ordinary Least Squares (OLS) summary results to see the effects of any transformations that I made to the data.

After running the OLS model on the training data, I observed that the horror genre count was the only statistically significant feature. From the Durbin-Watson value, I could see that the variance of the errors implied their homoscedasticity. Also, the Jarque-Bera value showed that the errors were not normally distributed.

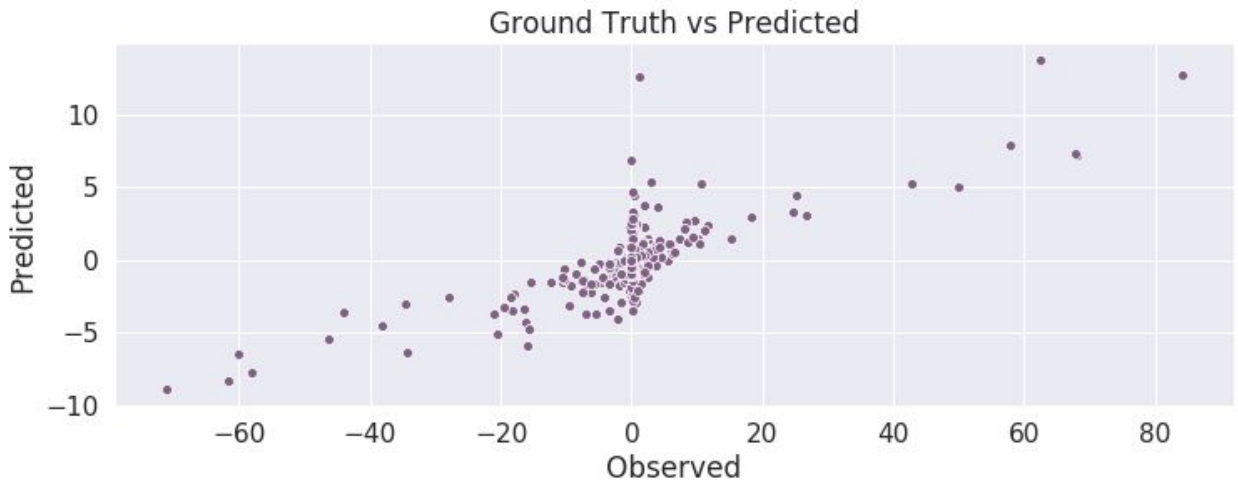
With this information in mind, I turned my attention to the Random Forest Regressor model. I fit the out of the box model on the dataset, without modifications, to get a baseline root mean squared error (RMSE), the metric that I would use for model evaluation.



The score for the model was 40.57. Using a crude and not very scientific reference value to gauge this error, I observed that the target standard deviation was 41.54. Seeing that they were in the same neighborhood, I felt that I could proceed with the framework of transforming the target variable to give it a more manageable distribution, then finding the best model to minimize the error. In the end, I hoped to produce an accurate and generalizable Random Forest Regressor model. I hypothesized that I would succeed.

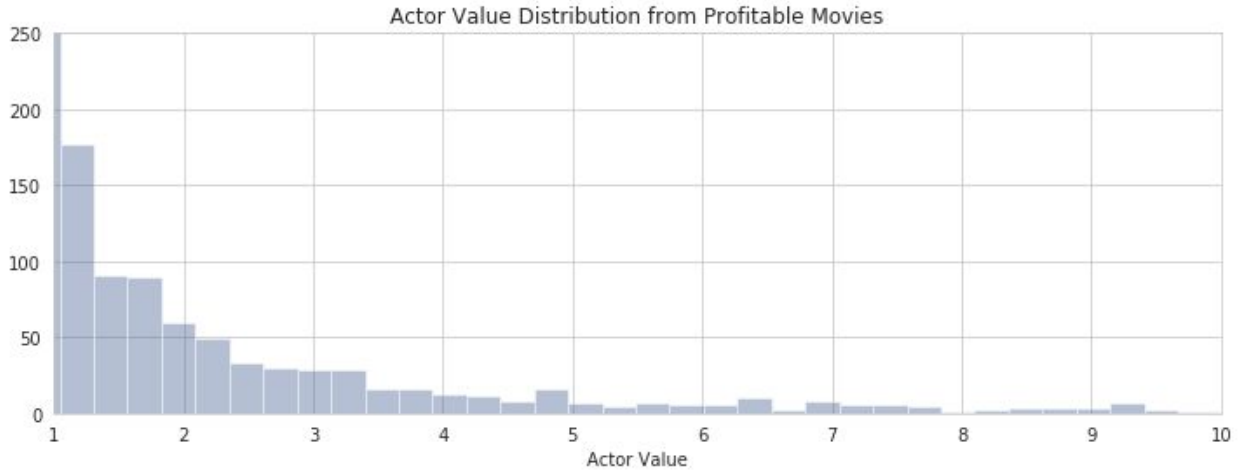
The first transformation that I made to the data was to drop several outliers. I chose to define the outliers by the criterion that their z-scores were at least 3 standard deviations away from the mean of the target values. This only dropped 31 observations out of nearly 15,000. This small change dropped the probability of the null hypothesis related to the significance of the F-statistic

from 0.1% to 4 e-24. The data ceased to be random noise, although it had a long way to go before becoming useful for prediction.



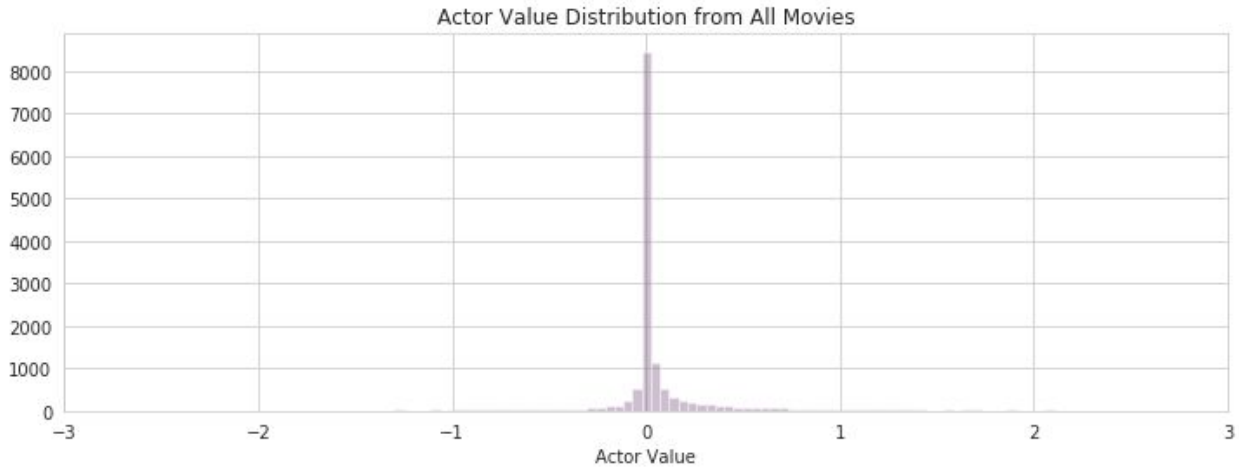
There were now 6 significant features out of 21, but the horror genre was no longer included. I suspect that Katie Featherston was dropped as an observation. She starred in 5 Paranormal Activities horror films, all of which had huge returns on investment figures.

While dropping some outliers resulted in more manageable data, what was needed was a total transformation of the entire target. Having built the dataset from API requests and feature engineered the target variable myself, I knew how it was constructed. To use the same function, $ActorValue \propto \frac{REVENUE}{BUDGET}$, across the whole range of profit values (positive and negative), would have created two very different distributions for each type of value. The positive values would have been multiples of the budgets, beginning at 1 and extended to infinity.



The negative values would have been converted into percentages of the budgets instead of multiples of them, being bound between 0 and 1.

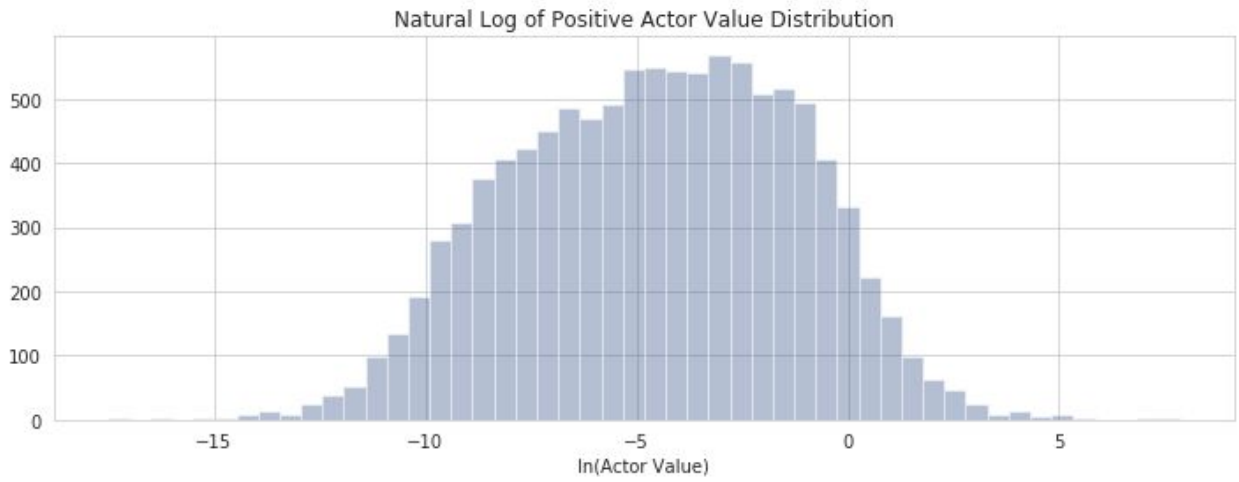
Instead of engineering the target this way, I had created the positive target values from a slightly different function, $ActorValue \propto \frac{REVENUE-BUDGET}{BUDGET}$, than the one I used to make the negative target variables, $ActorValue \propto \frac{REVENUE-BUDGET}{REVENUE}$. I did so with the intent of having all of the values being represented by multiples of the profits associated with the actors' movies. For each profitable movie, the actors in it would be given values that were positive multiples of the profit associated with that movie. To make the distribution symmetric about zero, the actors associated with movies that lost money would be given values that were multiples of their respective movies' negative profits, i.e. their losses.

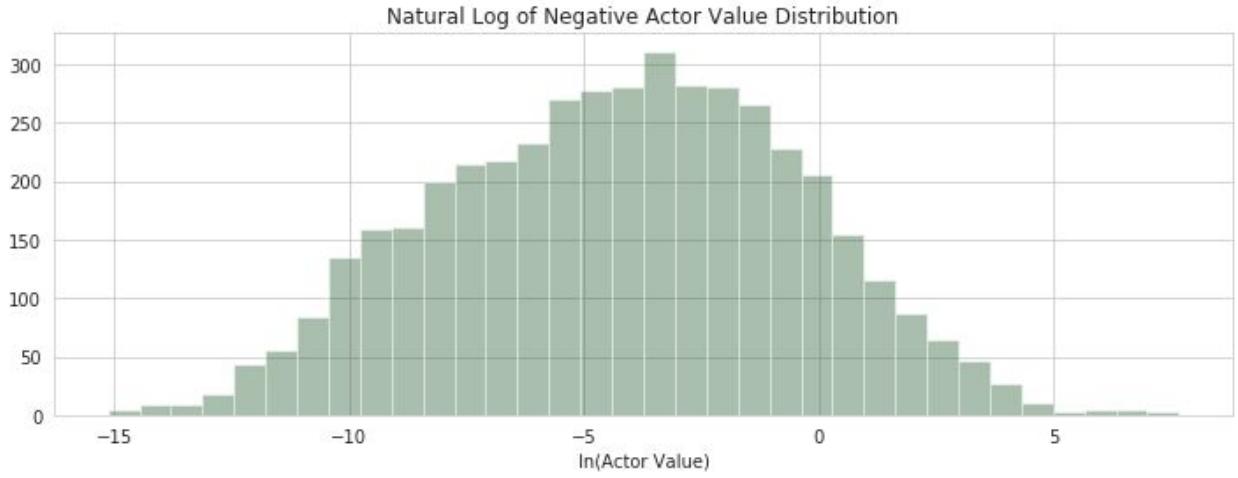


I felt that this was a good way to compare apples to apples, so to speak.

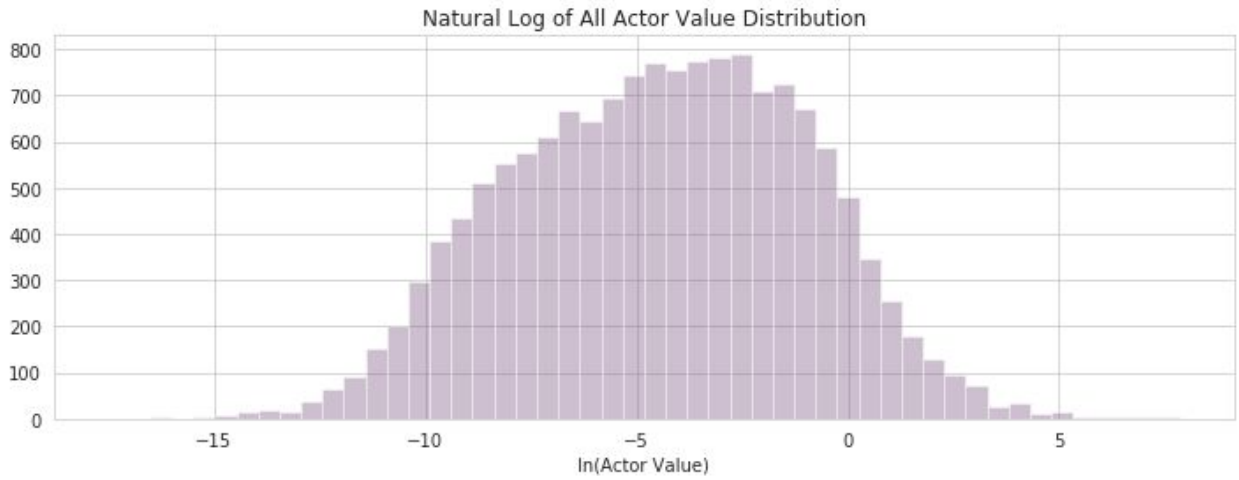
With this knowledge of the symmetry of the target in hand, I showed that taking the natural log of the positive target values resulted in a similar distribution to taking the natural log of the absolute value of the negative target values

.





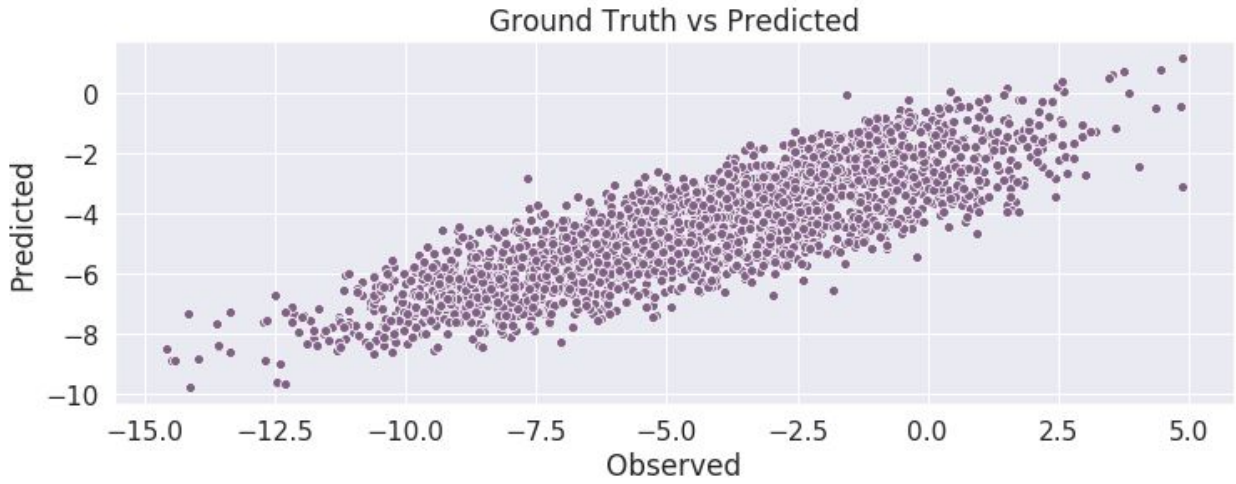
To be clear, their distributions were similar in the sense that the results of fitting the models on each of them were similar. In fact, they were so similar that I combined the two back together after transforming each with the natural log function. This resulted in what looked like an average of the original two targets.



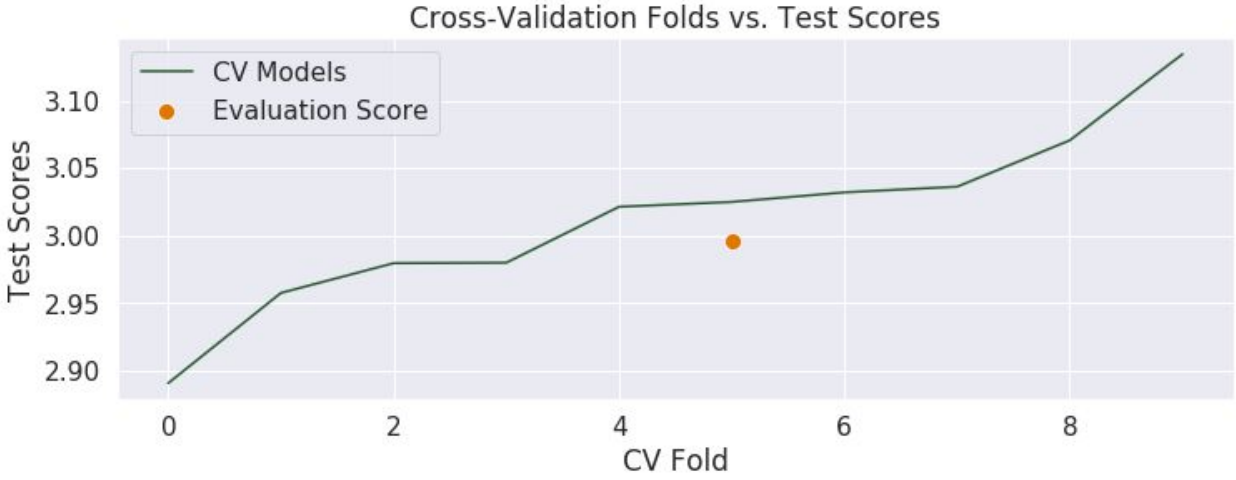
I took this re-engineered target and removed the outliers for good measure.

The first thing that I noticed about the new target were the small but significant correlations it had with all of the predictors. Eight of them had correlation coefficients in the twenties. Considering where the correlations were originally, that was an accomplishment. The F-statistic probability was zero. The skewness was near zero, meaning that the data were symmetric about the mean. Fourteen of the features were statistically significant. The ones that weren't were all movie genres. I decided to make this my final dataset to train the regression models for prediction.

I chose to use a randomized grid search over using an exhaustive grid search. I started with a broad and sparse grid. I felt that after each iteration I could tune the parameters by hand through either expanding or narrowing their ranges for the next grid as the results dictated. After running eight grids, while comparing training and test metrics to detect possible overfitting and observing the top models for each run, I settled on a final set of hyperparameters.



Then, I ran the model through a 10-fold cross-validation to get a range of metrics to compare with the model's performance on the test data.



The model performed equally well on the holdout set. It gave an RMSE of 2.996 on the test data, while the mean of the cross-validation scores was 3.013. The range of the cross-validation scores was [2.891, 3.134], landing the test score right in the middle. In the end, I produced an accurate and generalizable Random Forest Regressor model on a sticky dataset.