Final Report- Movie Box Office Model

Overview

Using a dataset of movies an attempt was made to build a model to predict box office success based on the people involved in the production of the film (actors, director, screenwriter, editor, and production company), specifics of the story (genre, duration, keywords), and user ratings. The ultimate results showed that these features are unfortunately not good predictors for box office success.

Data Sets

Kaggle - The Movies Dataset - (cast, crew, genres, user ratings)
https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset

The Movie DB - (revenue, budget) https://www.themoviedb.org/

Box Office Mojo - (Domestic Box Office sales) https://www.boxofficemojo.com/

Data Preparation

Feature Engineering:

The Kaggle Datasets contained several columns of unclean data, each entry of cast, crew, and genres were themselves tables containing columns of unique identifiers, names, etc, and could not therefore be used as model predictors as given. Multiple methods were used to translate each of these parameters for each film into usable data. The crux of each method used was sorting the movie data set by release date and the unique ids given to each actor, crew member, and genre, which provided a way to isolate films associated with that id.

Average Rating

Each of the films in the data set has a user review rating, and averaging these past ratings for a given id (actor, screenwriter, etc) can give an indication of overall quality for that id. Further averaging over all ids in a set (actors, screenwriters, etc) produces an average rating for that set.

Vote Average Rating

When averaging over films for a particular id, one film could skew the result. Each film in the dataset also has a feature for how many users submitted a rating, which can be used to calculate a ranked average, multiplying each rating by the number of votes for that rating, then dividing by all votes. Like before, this ranked average produces a rating per id in each set, which is then averaged over all ids in a given set.

Historical Average Rating

There is a saying in the film industry that a person is only as good as their last project, so an alternative way to rank the ratings for past films for a particular id is to treat the more recent

film's rating as more accurate than older films in the list. This can be achieved by using a linear ranking, taking the time span between each film in the list and the film under review, and dividing by the time span between the most recent film in the id's list and the film under review. This produces a ranking from 1 for the most recent film found, towards 0 for each older film in the id's list of movies. Again, this ranked rating produces a rating per id in each set (actor, director, etc), which is then averaged over all ids in a given set (actors, directors, etc).

Average Rating for All Movies

The previous ratings all relied on previous movies in the data set for each id to generate a rating for a given movie. This led to many films, especially the earliest films in the data set, missing the engineered features. Also, perhaps the best way to assess the quality of a given film professional on a project is not to look at the projects they've done thus far, but their entire history. This feature found all films in the data set connected with a particular id (actor, director, etc) and averaged the films' ratings over the id's entire history, then averaged over all ids in a given set (actors, directors) to estimate the quality of that set for a given film.

Vote Average Rating for All Movies

Like the previous feature, this one uses all the films in the data set connected with an id to assess the quality of that film professional, though like the Vote Average feature listed above, the films for each id are used in a ranked average based on the number of users who contributed to each film's rating.

Ranked Averages (Actors Only)

Since each film listed the cast in order of importance of the role they played in the film, the rating for the cast for a film can be calculated with a ranked average, weighting each cast member's rating by their position in the credits. Since this is a calculation for the cast, it is applied on top of each of the previous methods used to calculate each cast member's rating.

Ratings Total (Actors Only)

The cast list for any given film is much longer than the list of people performing a specific job in the crew. As such, any method of averaging the actors' ratings could be skewed by a cast member with a low rating. To have a measure of a cast's rating not biased by this, this feature does not average the actors' ratings, but instead uses the total sum of those ratings. Totalling the casts' individual ratings is applied to all of the methods described above for calculating an individual cast member's rating.

Top 3 Ratings Total (Actors Only)

Since many film stories are led by very few actors, the actors lower in the cast list likely have less effect on a film's success. To capture this in a feature, this measure sums only the ratings of the top three actors listed in a cast. This method for calculating a cast's quality is applied after each of the methods described above for calculating an individual cast member's quality.

In total, 42 separate features were engineered based on the cast, crew, genres, and keywords listed in the dataset for each film. These are summarized in the following tables:

	Average Rating	Vote Average	Historic Average	All Movie Average	All Movie Vote Average
Actors	X	X	X	X	X
Directors	X	X	X	X	X
Screenwriters	X	X	X	X	X
Editors	X	X	X	X	X
Genres	X				
Keywords	X				

Additional Actor Features

	Total	Top 3	Ranked Average	Ranked Total
Average Rating	X	X	X	X
Vote Average	X	X	X	X
Historic Average	X	X	X	X
All Movie Average	X	X	X	X
All Movie Vote Average	X	X	X	X

Separating by Production Company

One final measure of feature extraction was to take all of the production companies listed for a film, creating duplicate entries for each film which differed only by this column. This was used in exploration in two different ways, firstly using only the first listed (likely the production company in charge) and dropping all other entries, and secondly keeping all duplicates and looking at how each production company performed. Both were used in data exploration, but only the first is used in modeling.

Gathering Film Performance Data:

Since the ultimate goal was to create a model that predicted a film's performance, i.e. how much money the film made in comparison to how much was spent in its production, reliable information regarding both the film's budget and revenue were required. These were not initially present in the data set, but luckily a movie id common to this data set and The Movie Database was provided for each movie. By creating an account with this database, and using its api, a total of 10465 budgets and 6841 revenues were filled in for the 45432 movies listed in the set.

As the total number of revenues was much less than the number of budgets an alternative approach was used to gather more information. The website BoxOfficeMojo.com tracks the ticket sales for films, and further separates them into Domestic and International sales as well. Box Office Mojo has more Domestic records than International, so the former was chosen as a

measure of success. Unfortunately, the number of records provided by this method did not exceed the number of revenues listed in The Movie Database, but with 6395 records it was at least comparable in size and provided multiple outputs to use for the machine learning models.

Accounting for inflation:

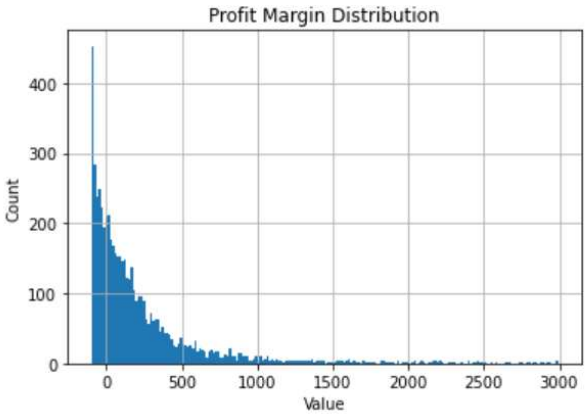
To ensure that the earning outputs for the machine learning models were not affected by inflation, a common problem with lists ranking top earning films, the revenue and Domestic earnings were divided by the budget for each film. This created a profit ratio, with underperforming movies resulting in values less than one and profitable movies resulting in values greater than one.

Modeling

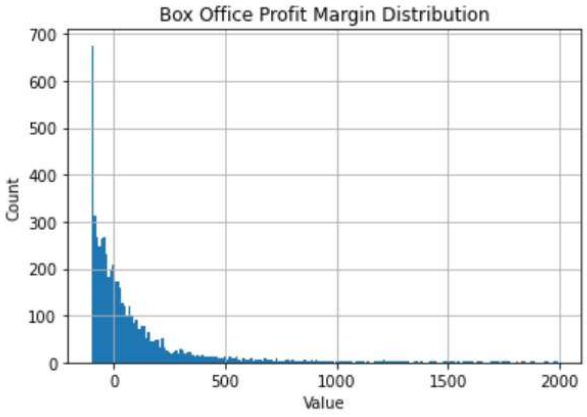
Independent Variable Selection:

There were three variables for consideration for the output of the supervised learning models:

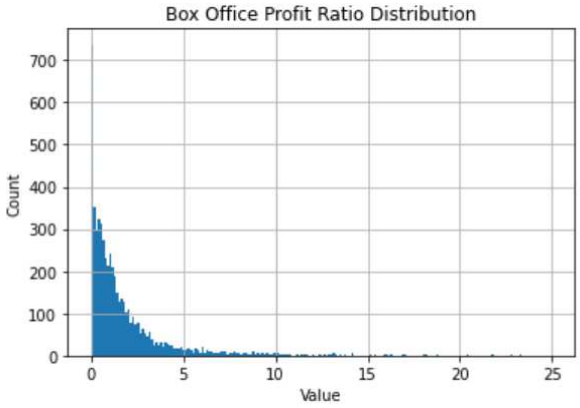
$$Profit\ Margin\ =\ \frac{(Revenue - Budget)}{Budget}$$



$$Box\ Office\ Profit\ Margin\ =\ \frac{(Domestic\ Box\ Office\ Earnings - Budget)}{Budget}$$

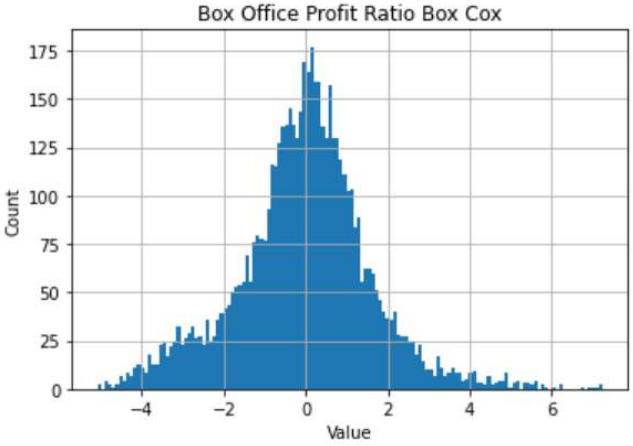


$$Box\ Office\ Profit\ Ratio\ =\ \frac{Domestic\ Box\ Office\ Earnings}{Budget}$$



All three of these features are highly skewed toward the high end, with most of the data grouped around the lower values and with a hard stop at the minimum value. These distributions would not perform well in regression based models due to the high level of skew, and may suffer in other model types as predicting the median would perform well for most cases. To increase the range of values to make a well performing model more powerful, the distributions can be transformed using a box cox method. However, this method involves taking the logarithm of the distribution, which will not work for the first two cases, since they involve negative values.

The box cox of the third output, Box Office Profit Ratio, produced the following distribution:



Model Performance:

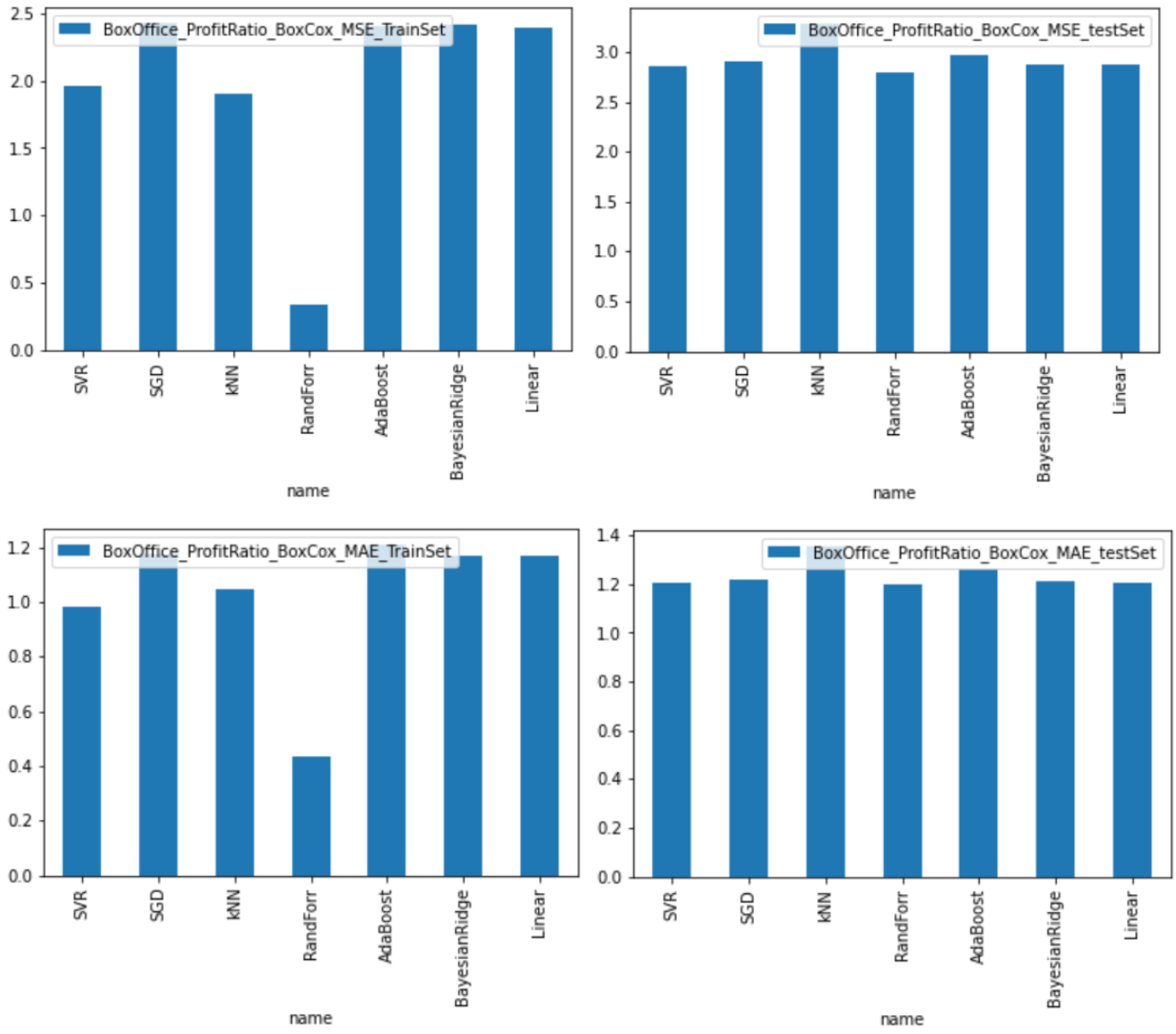
The following models were attempted for regression of the Box Office Profit Ratio Box Cox:

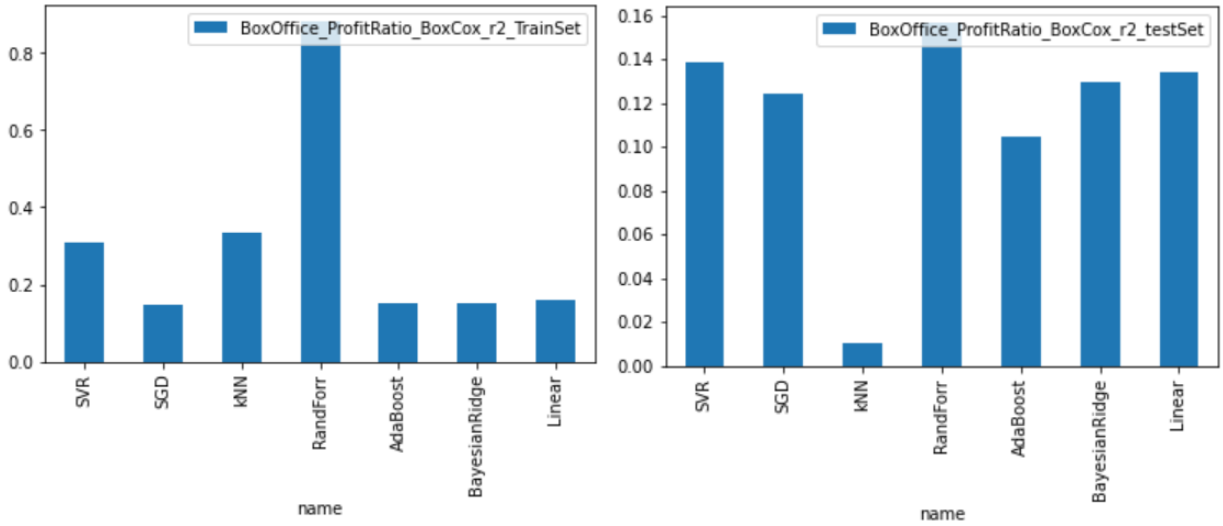
- Epsilon-Support Vector Regression (SVR)
- Stochastic Gradient Descent Regression (SGD)
- K Nearest Neighbors Regressor (kNN)
- Random Forest Regressor (RandForr)

- Ada Boost Regressor (AdaBoost)
- Bayesian Ridge Regressor (BayesianRidge)
- Linear Regression (Linear)

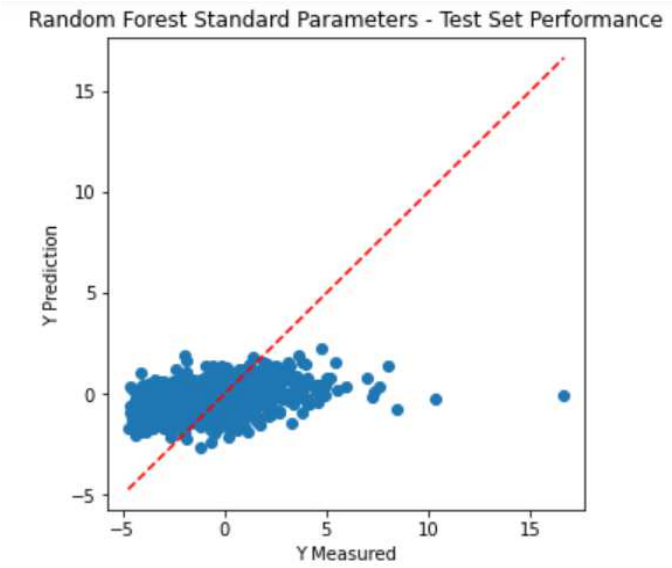
And below are the base models' performances using the following metrics:

- Mean Squared Error (MSE)
- Mean Average Error (MAE)
- R^2 Score (r2)

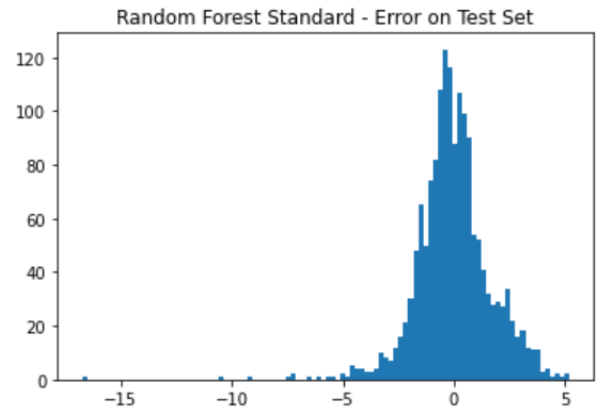




The Random Forest model clearly performed the best on the Training Set, and performed as good as the other models on the Test Set, but no model performed particularly well. Below are the results of the Random Forest Model in more detail:




```
MSE : 2.799383385956045
MAE : 1.2028713138377192
R^2 : 0.15503733708840828
```



Mean: -0.01597923584035411 Standard Dev: 1.6730594878778224

Optimization:

To try to improve this model, Bayesian Optimization was used on the meta parameters of the Random Forest, using k fold cross validation with 10 splits, and reporting the median score for each step.

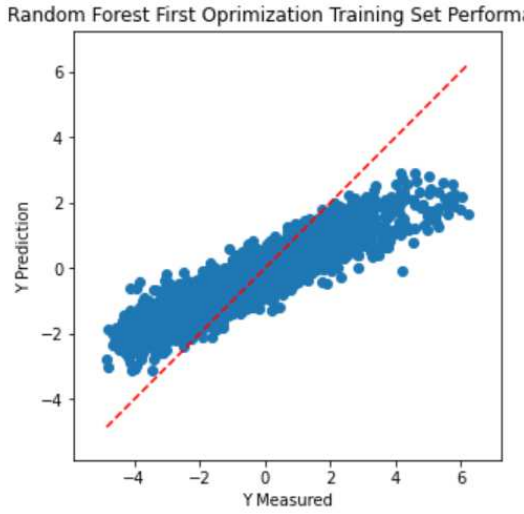
The first optimization used had a wide parameter spread, using the following ranges over 80 iterations:

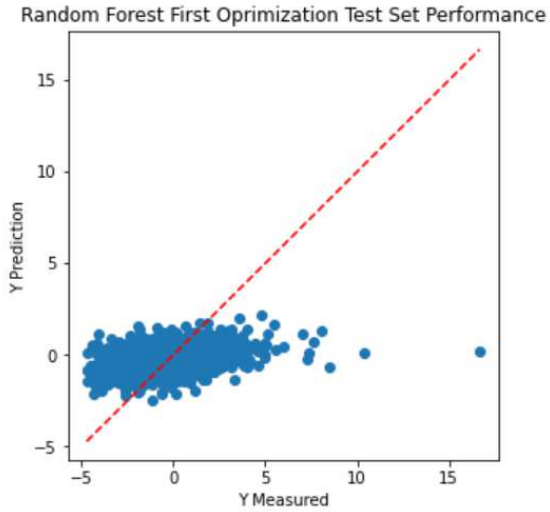
n_estimators:	(90 - 110)
max_depth:	(40 - 60)
min_samples_split:	(2 - 10)
min_samples_leaf:	(1 - 10)
min_weight_fraction_leaf:	(0.0 - 0.5)
min_impurity_decrease:	(0.0 - 0.2)
ccp_alpha:	(0.0 - 0.2)
max_features:	(24 - 44)

The following parameters produced the best results:

n_estimators:	101
max_depth:	51
min_samples_split:	4
min_samples_leaf:	5
min_weight_fraction_leaf:	0.0
min_impurity_decrease:	0.0
ccp_alpha:	0.0
max_features:	25

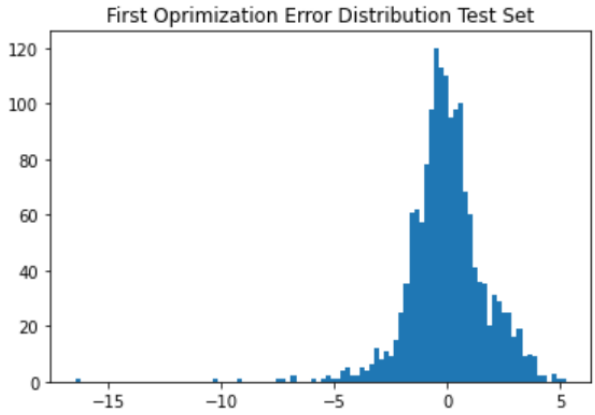
But the results were still not great, as the best score was just 0.167.






```
MSE : 0.8711301020689038
MAE : 0.6688260174110391
R^2 : 0.6936872354035357
```

```
MSE : 2.7615067661006263
MAE : 1.1919977562358146
R^2 : 0.16646997248078932
```




```
Mean: 0.0012127691818250735      Standard Dev: 1.6617777514732646
```

Since many of the optimized parameters did not differ from the default variables for this model, a second optimization was attempted to improve performance. The reasoning behind this was that if the first optimization spent iterations modifying parameters that ultimately performed best at their default values, then these iterations may have been better spent exploring other parameters' values. By limiting the values that could be altered during Bayesian Optimization, another optimization may perform better.

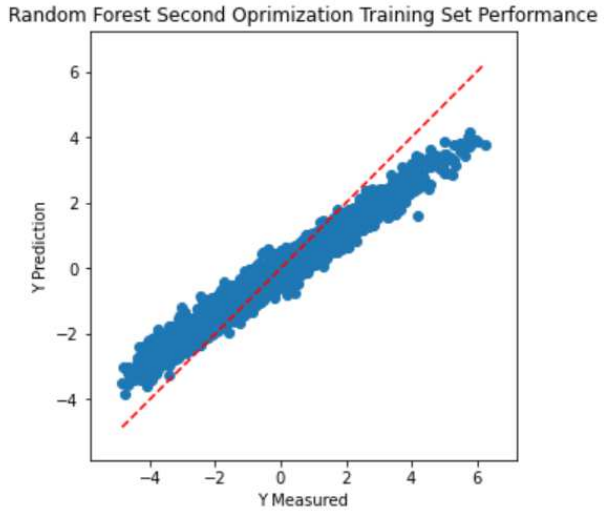
The parameter ranges for the second optimization were as follows:

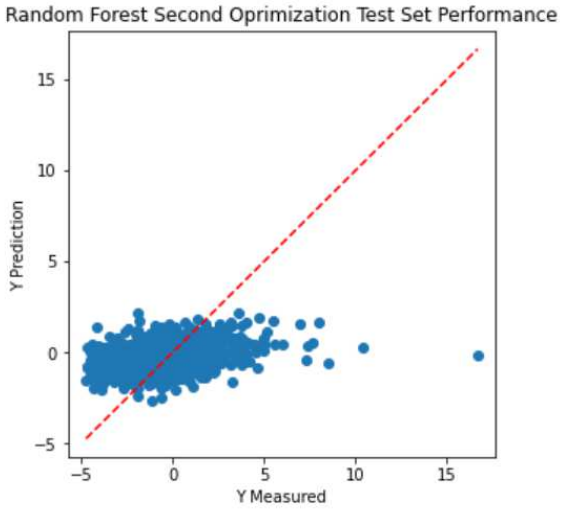
n_estimators:	(125 - 250)
max_depth:	(37 - 55)
max_features:	(10 - 40)

The following parameters produced the best results:

n_estimators:	160
max_depth:	39
max_features :	38

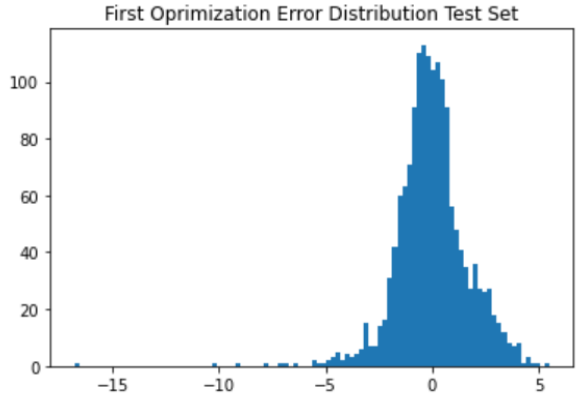
The results were again less than ideal with a best performance score of 0.161.






```
MSE : 0.3343961152603857
MAE : 0.4316499854236619
R^2 : 0.882417335490463
```

```
MSE : 2.762038502829518
MAE : 1.1942630273074142
R^2 : 0.16630947367785076
```



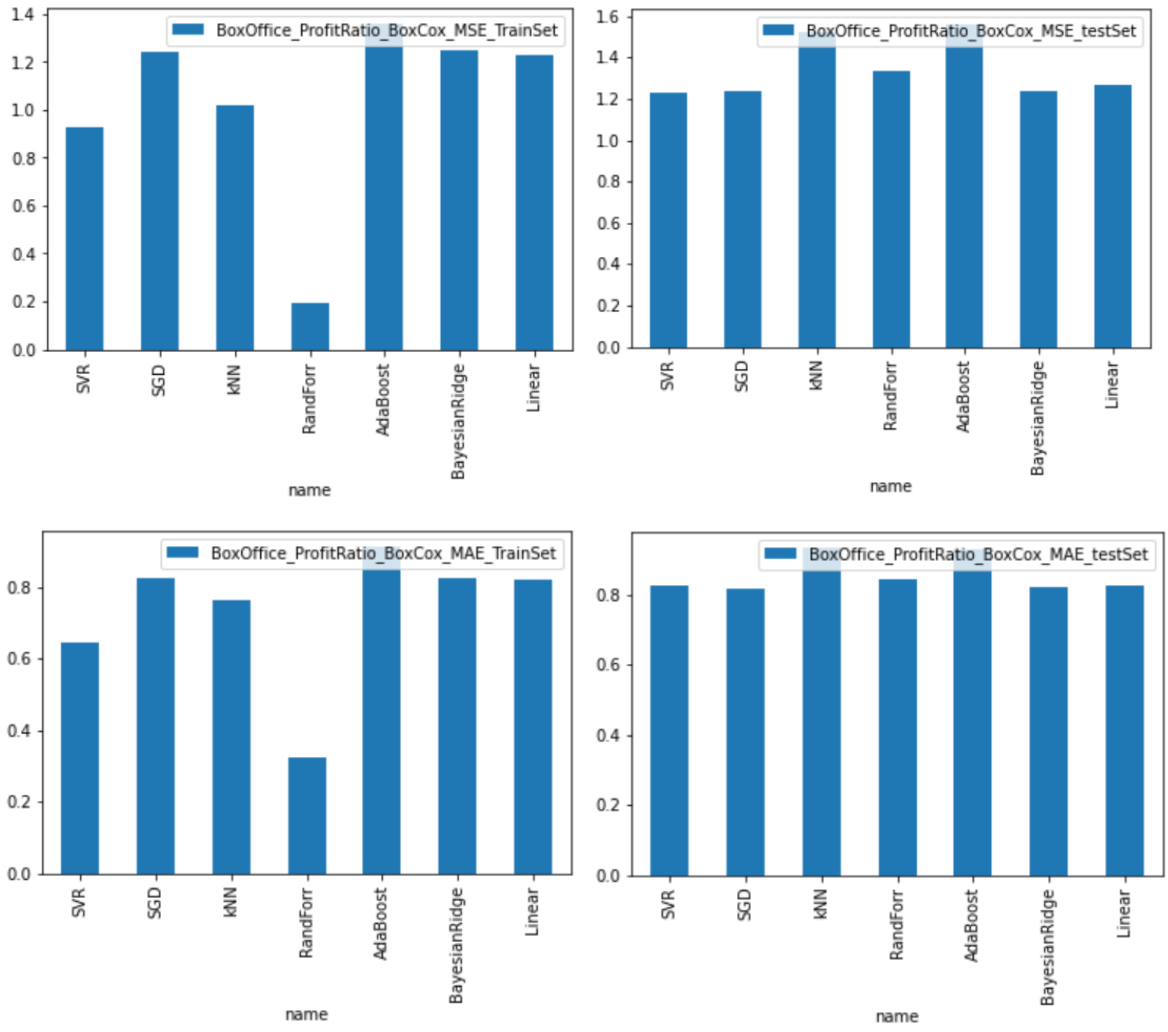

```
Mean: 0.0011202869675978533      Standard Dev: 1.661937799012534
```

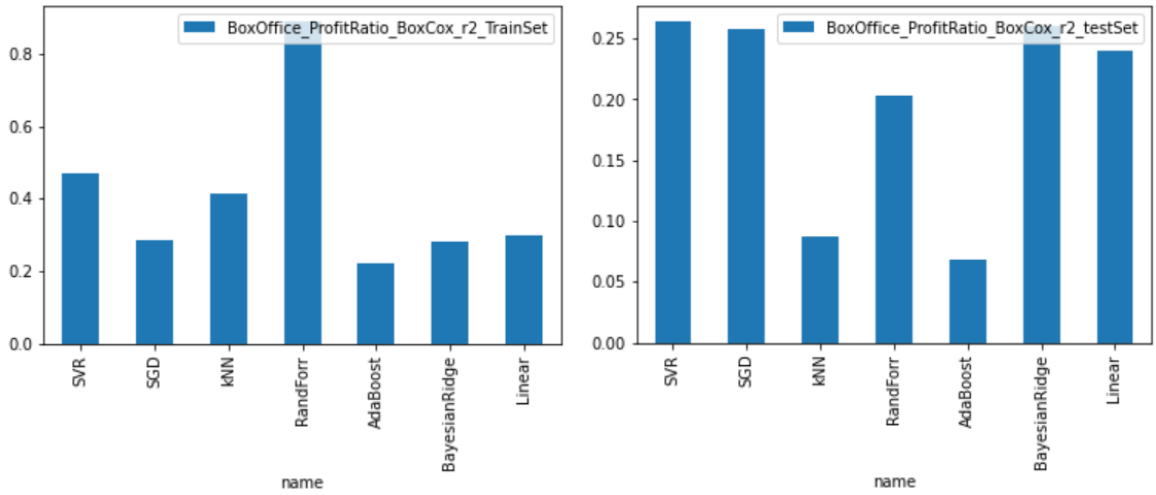
Production Company Data Model Performance:

Adding the Production Company feature was attempted, exploring the possibility that this would improve model performance by limiting the data that could be used to the most popular production companies. A successful model of this type would not be able to extrapolate to every movie, but could encompass a large amount of the market share of the film industry. The

production company used for each film was the first one listed in the credits, and therefore likely the one that contributed the most resources towards the production.

The data set was further limited to the top 60 most frequent production companies, which accounted for roughly 60% of the films listed. The same base regression models and the same metrics described above were used, with the same output of Box Office profit ratio Box Cox. The performance of those models is shown below:





As seen in the previous dataset the Random Forest model performs well on the Training Set, but does not generalize to the Test Set.

Lessons and Insights

Box Cox on the Test Set:

In an effort to avoid data creep in the Test Set, the lambda value used in transforming the Box Office Profit Ratio was calculated from the Training Set, then that value was used to transform the Test set without knowledge of the underlying data. This process was correct, but initially the value was calculated based on the formula below (described in the documentation for the scipy stats boxcox function), and not the function itself.

```
y = (x**lmbda - 1) / lmbda,  for lmbda != 0
    log(x),                  for lmbda = 0
```

The thought was that using this formula would isolate the test set in a way that using the function would not, which was technically true, however, it produced wildly incorrect values. Using this formula resulted in Test Output Values that were fundamentally different from their Training counterparts, resulting in large negative R squared values for the models tested. Clearly the documentation for the scipy box cox function did not describe fully what was happening behind the scenes as using the function for the Test Set (and providing the lambda of the Training set) provided much better results.

Feature Engineering:

The experience of this capstone highlights the adage ‘garbage in, garbage out’. Despite the many methods used to translate the cast, crew, and descriptive data provided for each film into usable data for modeling, they all ultimately depended on the user ratings for each film to predict box office success. There is no clear indication that a popular movie was successful in the box office or that an unpopular movie lost money, so any further calculations based upon this assumption was flawed to begin with.

Also, user ratings are not an unbiased measure of a film's quality. The ratings in the data set are mostly in the 5-8 range, users being too kind to give a low rating and selective about which films got too high of a rating. Perhaps a better measure would have been to use historical Box Office success to predict future success, though this would only have been reasonable for those features that did not consider all movies in the set, and features for new movies tested against the model would have to use the data used to train the model to obtain their features. Regardless, in hindsight the 42 features engineered were based on a false assumption.

Conclusions

Ultimately, the film data gathered from Kaggle Movie Dataset, The Movie Database, and Box Office Mojo cannot be used to accurately model Box Office Success. There are simply not enough usable parameters listed in the dataset to model from, and the ones this capstone engineered and tested did not ultimately perform with a sufficient degree of success.