Predicting IMDb Film Ratings

Authors: Jessie Wang (yw696), Anil Vadali(arv42)

Background

In the United States alone, there are over 600 movies made each year, however only a select few of these movies are very successful. Moreover, among these select few, an even fewer number of films are scored very well by the infamous IMDb, or the Internet Movie Database, film rating system and make it onto the IMDb's Top 250 movie list. The IMDb rating consists of scores ranging from 0 to 10 by increments of 0.1. The score is calculated by the database utilizing a disclosed weighted average model based on the number of reviewers and the corresponding reviews.

With the emergence of new technology companies such as Google starting to make movies, and with Amazon Studios and Netflix winning Oscars last year, it has become extremely important to understand what makes a good movie to survive in this newly competitive film industry. Determining which movies to make and not to make, casting the optimal actors in certain roles, and predicting what consumers and critics will enjoy have been crucial decisions in creating a new movie and has become somewhat a science in recent years. As a result, this project hopes to accomplish the following goal: **predict the IMDB rating score of a certain film based on certain features and aspects.**

Data

Dataset Description and Feature Engineering

The dataset that we utilize for this project was originally published on Kaggle under the name "TMDB 5000 Movie Dataset" by *The Movie Database* and was last updated in October 2017. The original dataset consisted of 4803 data entries and 25 total features from two different sources: 22 features from the tmdb_5000_movies table and 3 features in tmdb_5000_credits table. Originally, the data quality was tremendous and few missing data. Overall, the data had 5 total quantitative fields, and 20 total categorial fields describing basic attributes of each film ranging from budget to runtime to production companies. Ultimately, this data was combined and further modified as follows.

Immediately, we realized that some features were not going to be helpful during our analysis and removed them from our dataset. Specifically, we removed features such as homepage, movie ID, original language, release status, spoken languages, crew, and overview. We believed that homepage and movie ID were irrelevant for the purpose of prediction. Additionally, we discarded original language and spoken languages because the majority of movies within our dataset are English and the sample for non-English movies was not big enough to give reasonable insights into the data. Release date was discarded for a similar reason as most films in the data were already released. We didn't believe crew would play a large role in the success of the film. Finally, though overview category provided a summary paragraph of the film, which may be useful, but would require advanced Natural Language Processing techniques.

Additionally, to handle categorical features properly, we introduced multiple new fields by utilizing feature characteristics, feature frequency, and binary encoding in many cases to enumerate these types of features. Many categorical features such as cast, production companies, genres and keywords were encoded as lists. In the case where the state space of a feature was relatively such as in the case of genres which only had 19 possible values, we listed each value as its own feature and encoded the value as 1 if a certain film had that value and 0 if it did not. In the case where the state space was much larger such as in the case of cast and keywords, we first ended up taking a predetermined subset of the list provided. For the cast field, which sorted actors/actresses by role importance, we considered the first six entries only, and for the keywords field, which provided one-word descriptions of the film, we considered the first five entries. Finally, we assigned each of these a certain weight based on the frequency of each individual value (actor or keyword) over the entire dataset. The main idea behind this approach was to see if casting more popular or recurring actors/actresses or types of movies would correspond to higher ratings. To handle the Production Companies field, we took a different approach, after researching and finding that 6 studios release a substantial number of movies every year: Walt Disney Studios, Warner Bros., Fox Entertainment, Universal Studios, Sony Pictures Group, and Paramount Motion Group. As a result, we also decided to use a binary encoding in this case where if a film was produced by one of these six groups, a 1 was encoded, otherwise a 0 was encoded. This premise behind this is to understand whether a movie needs to produced by a major production company in order to be successful. Finally in the case of text based features, such as title and tagline, we created fields that take the length of each field to see if text features can influence the success of a movie.

After preprocessing, we include only the following features: budget, popularity, revenue, runtime, status, vote count, rating, release date year, release date month, frequency of the top 6 actors, sum of the frequency of actors, frequency of top 5 keywords, sum of the frequency of keywords, title length, tagline length, major production company indicator, title change indicator, and 19 genre indicator variables.

Missing Data/Inappropriate Data

Though the data provided was mostly in great quality, there were also some issues we had to deal with during pre-processing that could impact the overall performance of our model such as NULL or missing values. In our case, we ended dropping the films that ended up having at least one NULL in any of its columns. This is due to the fact that the upon observation during the initial data analysis, many of the rows that ended up having missing values would have been deleted in the first place due to not meeting some of the criteria in the first place such as being a foreign film. In many cases, the reason these films ended up having missing entries was because the database was not able to find the appropriate information to complete the profile and were very obscure.

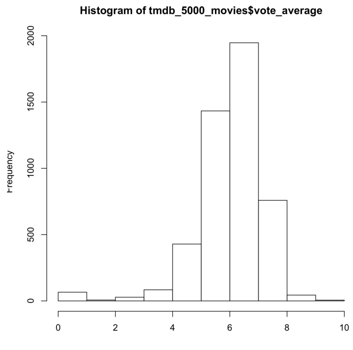
Additionally, another issue of the dataset was inappropriate data values for certain fields such as budget, revenue, and runtime. More specifically, we noticed many films ended up having a value of $0 or $10 for the budget while having a large value for the revenue. The opposite situation also occurred quite often as well where the revenue was abnormally small being $0 or $1, while the budget was extremely large. Moreover, there were also issues such as the runtime being 0 minutes total as well. Though technically possible, we assumed that these instances were errors that may have occurred during data extraction process by the original publisher of the source. As a result, we ended up replacing budget and revenue values under $10,000 with the mean of the respective columns to preserve the information overall.

Initial Data Analysis

Initial Data Visualization

Data Visualization plays an incredibly important role in identifying which features will play an important role in the analysis and how different features will relate and interact with each other. This is very useful in creating an accurate model for predicting IMDb rating at the end. To start off, we wanted to observe and understand how the final overall ratings were distributed.

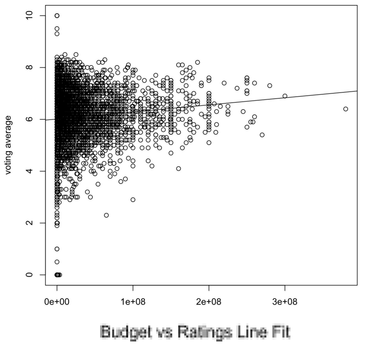
Histogram of average rating: The histogram gives us a visualization representation of the voting average range. As we can see, the histogram gives a bell shaped data, which means the vote_average of all movies are pretty close to normal distribution, slightly skewed right. This indicates that relatively few movies should be predicted as very low scores or as very high scores and most predicted scores should fall within the 5 to 7 range.

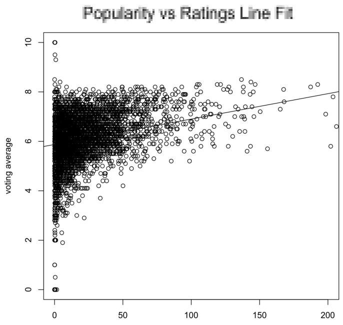


Ratings Histogram

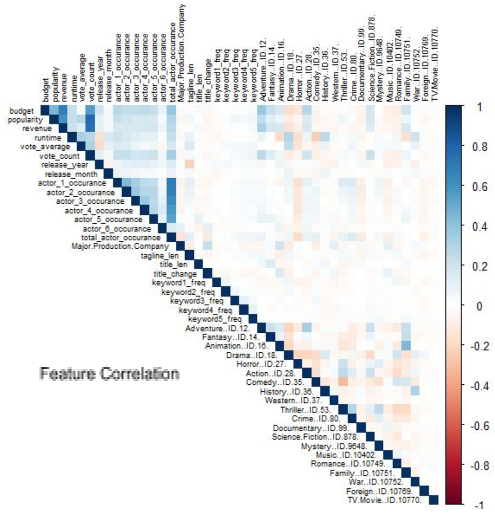
Among the features that could play a large role in determining the success of a certain film, we, as a group, initially predicted that the budget and the popularity fields would be the most significant. Specifically, we imagined that budget would play a large role due to the fact that many times movies that spend a lot, whether it be on getting a top-notch director, the best actors, or even developing unbelievable special effects or settings, tend to do very well both in the box office and often times in the reviews. Additionally, we originally believed that the popularity feature would play a large role in determining the overall movie rating score since a movie that is often times more popular, is generally, admired more. As a result, to test our hypotheses, we wanted to see if there were any general trends among these variables and the final rating score.

<u>Line fit between rating and budget</u>: The left graph is the scatter plot of vote_average vs. budget (vote_average as the y axis and budget as the x_axis), which give us a general idea of the data distribution. The right graph is an attempt to fit a linear model, which shows slight correlation between vote_average and budget.





<u>Line fit between popularity and rating</u>: The graph on the left is the scatter plot of the popularity ratings for each movie in the dataset we analyzed and it is compared against what rating the movie ended up getting (popularity score on the x-axis vs. voting average on the y-axis). As a result, we wanted to observe any trends between these two fields. Though there appears to be a positive linear trend between the two variable, we later realized that the popularity field is not very significant in the prediction of the overall score based on the analysis described below.



Ultimately, rather than trying to find each important feature individually, we decided to find the relationship between all features through the use of a Correlation Matrix.

<u>Correlation Matrix of Features</u>

The correlation matrix did indeed confirm our original hypothesis that there was a slight trend between the average rating and both the popularity and budget fields. However, the immediate conclusion we drew from the plot itself was the overall lack of strong correlation, either positive or negative. This indicated that dimensionality reduction was necessary in order to accurately predict the the final ratings accurately, otherwise, we would be creating additional noise and placing unnecessary weights on features that do correlate strongly with ratings. However, overall, we can see that among the features that correlate the most with ratings(5th row from the top) include: actor occurrences, runtime, popularity, budget, release year and month, and genre indicators such as Drama, Animation, History, and War.

Dimension Reduction

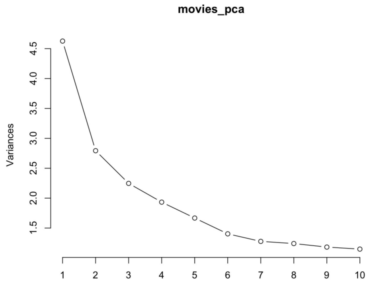
Knowing that our dataset contains too many features to adequately predict the average rating without adding extraneous noise, our main goal was to reduce the dimensions and keep only the relevant and most significant features. To find these dimensions, we initially performed Ordinary Least Squares Regression on the data and observed the summary statistics. Within the summary statistics, we can observe which variables that have a significant relationship with the average rating field, based on the significance codes provided. The significance codes indicate the strength of the p-values: a small p-value illustrates that it is unlikely we observe a relationship between the predictor and response variable purely due to chance and we can reject the null hypothesis that the relationship is due to chance in the significance test. The significance codes are expressed from one-star to three stars with three stars representing a highest range of probability for relationship. Utilizing this information, we reduce the number of features to only the features that display some significance code. As a result, we have determined 17 features that most impact the value of the average rating for the films: Budget, Popularity, Revenue, Vote Count, Release Year, Frequency of the Main Actor/Actress, Length of the Tagline, Frequency of the Second Keyword, and whether the film is Animation, Drama, Action, Western, Crime, Documentary, Science Fiction, War, and/or Foreign. Overall, the average rating is most strongly related to Budget, Popularity, Revenue, Runtime, Vote Count, Release Year, and whether it the film is a Animation, Drama, or Documentary due to the fact that these fields have three significance stars. Overall, we utilize only these 17 features for the following models.

Unsupervised Methods

To better understand our dataset, we want to know 1) if our high dimensional data lies on some intrinsic lower dimension space. 2) If there is distinct underlying pattern between "Good" movies with high ratings and "Not so good" movies with relatively low ratings.

PCA

Based on the decreasing variance explained with respect to number of dimensions, we can conclude that our high dimensional data lies in a much lower, 10 dimensional subspace.

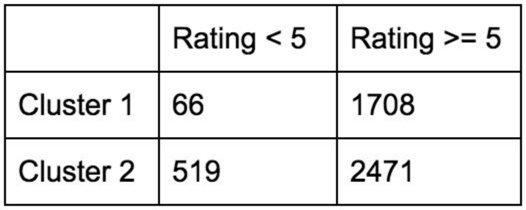


Feature Variance Plot for PCA

K-Means Clustering

Without learning the relation between feature space and ratings from the training data, is it possible to split data into two categories ('Good' and 'Not so Good') based on the pairwise distances? With K-Means clustering, we can cluster data into 2 groups and compare the clustered results with true value. We will use rating >= 5 as indicator of whether it's a good movie.

As we can see, the clusters created by K-Means only has a 0.533 correct classification rate. This means there exist many points with similar features (so the distance of these points are really small) but have very different ratings. This implies we would need more advanced data analysis models for prediction.



	Rating < 5	Rating >= 5
Cluster 1	66	1708
Cluster 2	519	2471

Cross-Validation Split: For cross-validation, we decide to separate the entire dataset into 80% training and 20% validation data. We will also used k-fold for choosing lambda in both ridge and lasso regression and choosing threshold for classification.

Models

Since the output of our problem, IMDb film ratings, is real-valued ranging from 0 to 10, we believed that utilizing multiple different regression techniques to predict the final outcome would be the best approach to get the most accurate answer. Specifically, we utilized 6 different regression methods: Ordinary Least Squares(OLS) utilizing all features, OLS utilizing reduced features, LASSO Regression (OLS with L1 Regularizer), Ridge Regression (OLS with L2 regularizer), RBF-kernelized Support Vector Machines, and Regression Trees and Random Forests. We normalized the data for the first 4 regression models to avoid model being dominated by features with large numbers.

Additionally, we defined a different interpretation of this problem and fit a classification algorithm (Logistic Regression) and also tried to utilize unsupervised learning methods in a different setting. The basis behind this is to understand how a problem could be interpreted and approached based on a different perspective.

Regression

Measure of Performance: Throughout this analysis, we will be utilizing Mean Square Error (MSE) as the measure of performance for each of the models. Mean Squared Error is defined as the sum of the difference between each prediction and corresponding true value squared all over the number of predictions made total. In other words:

$$\frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i})^2$$

Ordinary Least Squares Regression:

The first regression model that we developed and tested was Ordinary Least Squares Regression without any regularizers, the most basic form of regression. The goal of Ordinary Least Squares is to choose a *w* such that it minimizes the sum of squared residuals, or in other words:

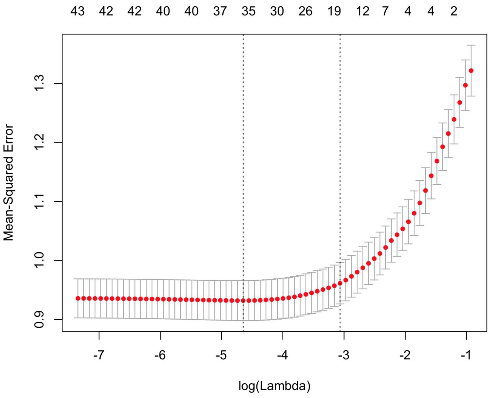
$$\textit{minimize } ||y - Xw||^2$$

As mentioned previously, Ordinary Least Squares was performed in order to identify the most significant features in determining the average rating. However, we also hoped to observe how well the model did utilizing all features in determining the rating predictions. Overall, the Ordinary Least Squares Method utilizing all features ended up with a mean square error (MSE) of 0.8997. We plan to utilize this result as a baseline score for the rest of the models as it is the most naive approach.

After reducing the number of features considered in our problem to the 17 features discussed in the Dimension Reduction section above, we ran the performed OLS with no regularizers once again on this reduced feature space. This time all features were very significant in determining the rating predictions based on the fact that each feature ended up having a very small p-value. The Ordinary Least Squares Method utilizing the 17 most significant features resulted in a MSE of 0.8874.

Lasso Regression/Regularization:

We decided to explore utilizing regularizers to identify and address any weak and discerning characteristics within our model. Specifically, we approached Lasso Regularization, or the



L-1 regularization, in the case that our model still incorporates too many features even after the dimension reduction in the previous section. One of the main benefits of Lasso Regularization is it can help reduce the effect of any extraneous or unhelpful features and further decrease the chance of overfitting to the training data. Additionally, since much of our data is binary encoded as either 1 or 0, it seems natural to utilize a regularization technique that encourages sparsity. Sparsity is encouraged in many different situations as it makes the solution much easier to interpret. Lasso Regularization essentially adds a penalty equivalent to the absolute value of the magnitude of the coefficients, and can be summarized as follows:

Cross-Validation Plot for Lasso Lambda

$$\textit{minimize } ||y - Xw||^2 + \lambda||w||_1$$

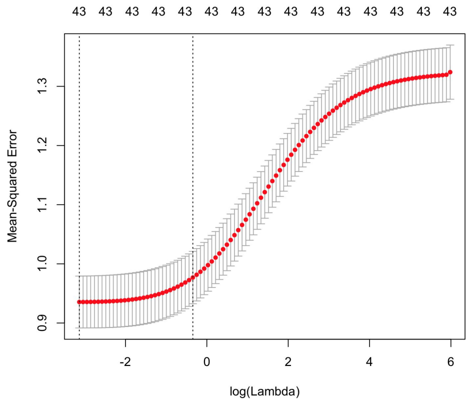
To find the optimal lambda value (0.047) that minimizes the Lasso Objective Function above we performed cross-validation. Ultimately, the overall MSE using Lasso Regularization combined with OLS with reduced feature space ended up being 0.8726, which is slightly better that the baseline score.

Ridge Regression/Regularization

We also decided to try to Ridge Regression, or L-2 regularization, as well, even though we did not believe strongly that it would improve the score greatly compared to the Lasso Regression. The main reason opted to test Ridge Regression on our data is because it has the high possibility of reducing the chance of overfitting the model. However, the downside is that it does not provide the advantages of Lasso Regression, which are extremely important for our problem. Ridge Regularization essentially adds a penalty equivalent to the square of the magnitude of the coefficients, and can be summarized as follows:

$$\textit{minimize } ||y - Xw||^2 + \lambda||w||^2$$

Again, the optimal lambda value that minimizes the Ridge Objective Function above was identified by performing cross-validation. The optimal lambda value was 0.7088. The MSE using Ridge Regularization combined with OLS with reduced feature space ended up being 0.9137, which confirms our assumptions that this model may not perform very well. In fact, the model performed worse than our baseline model.

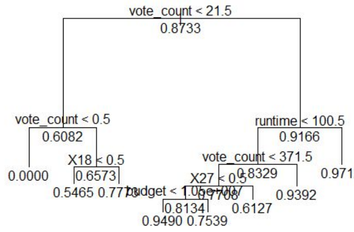


Cross Validation Plot for Ridge Lambda

Regression Decision Trees and Random Forests

Another model that we constructed and utilized to predict IMDb ratings for our problem was a Regression Decision Tree. We further extended this model to Random Forests, which we will touch on in the next paragraph. A Decision Tree is essentially a nonlinear model that has many advantages including being easy to interpret, being able to handle mixed data, making predictions quickly by looking up values within the tree, and capturing interactions between predictors. This makes it a great model to test on our data. A Regression Tree has both predictors, X, and response variables, y, that are real numbers and functions, at a high-level, in a two-step process. The first step is to divide the predictor space X into J non-overlapping regions $R_1,..,R_j$. Additionally, every observation in a region R_j must have the same

Regression Tree Model



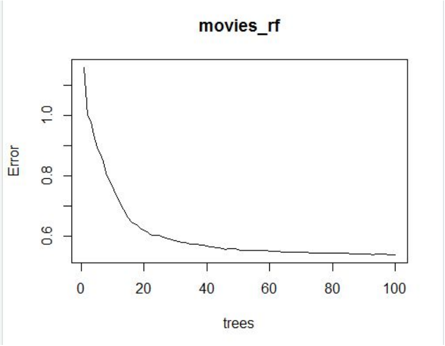
prediction. The catch is that we would like the find regions such that they minimize the Residual Sum of Squares error given by:

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

To do this optimally, we must go through a computationally intensive procedure known as Recursive Binary Splitting as well as Tree Pruning if the model starts to overfit. More information on the specifics of these algorithms can be found in the ORIE 4741 Decision Trees Lecture presented by Professor Sumanta Basu. However, we utilized an R function that did much of this work for us already. As a result, the resulting regression tree output when utilized on our data is illustrated to the right. The MSE for when using a single Regression Tree ended up being 0.6818.

One of the biggest deficiencies of the Decision Tree model is that it has the possibility of overfitting very easily. As a result, we can extend our Regression Tree model to a Random Forest in order to solve this problem. A Random Forest is aggregation method that creates numerous different decision trees upon initialization and training time, and predicts the response based on the average prediction of the trees used to train. The process is very similar to Bagging and the averaging leads to a substantial decline in variance. Again, please refer to Professor Sumanta Basu's lecture on Decision Trees to learn more about the specific algorithm.

In our Random Forest model, we utilized an aggregation of 100 decision trees to predict the final average ratings for each film. The graph on the right shows a plot of decreasing error with increasing number of trees. The MSE for utilizing a Random Forest Model is 0.5176, which is a dramatic improvement over our baseline models. However, this is expected as Professor Basu states in his lecture: When there are a small number of predictors that are strongly and we use a large number of decision trees, this will give the largest reduction in test MSE.



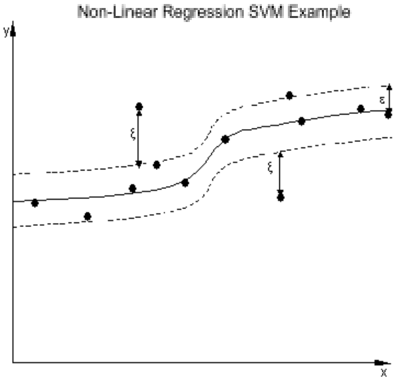
Error vs. Number of Trees for Random Forests

Support Vector Machines(SVM)

As we have learned in class, Support Vector Machines are prominently used for classification problems in determining the maximum-margin hyperplane that separates the classes while minimizing the overall loss function. However, SVMs can also be extended for regression problems with a few modifications. In the case of regression, the SVM tries to identify a maximum-margin around a line of best fit. Similar to the Classification case, the points within a certain threshold are within the safety margin. As a result, it ends up allowing some errors; the trade off is it reduces the severity of some mistakes. Overall, the goal of the SVM is to find a maximum margins of the line of best fit such that the error is minimized. The loss function that is minimized still remains:

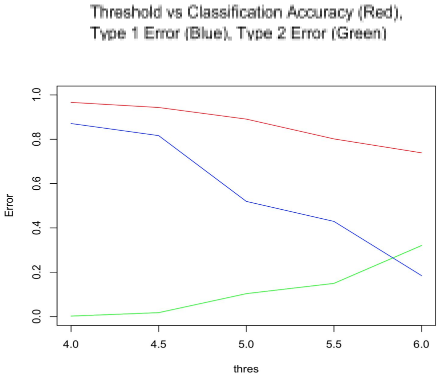
$$minimize \sum_{i=1}^{n} l_{hinge}(x; y; w) + \lambda \|w\|^2$$

Non-Linear Regression SVM Example



For more information with regards to exact parameters of the SVM regression model, please refer to Sources 4,5,6,7 in the bibliography which describe the exact process. Additionally, for our problem we attempted to utilize a linear SVM as well as a kernelized SVM. A kernelized SVM is used when the data needs to be transformed to a higher dimension feature space in order for SVM to find a proper fit. In our case, we are utilizing the Radial Basis Function as way to transform the data. The MSE for Linear SVM for ended up being 0.8566, while the MSE for the RBF SVM was 0.8362. This illustrates that the SVM performed better than the OLS models, however it was not good as the Decision Tree models. Additionally, it indicates that our data may not be completely linearly separable due to the fact that the kernelized version had a lower error.

Classification



The results from Regression may seem well and good, but to many people the difference between a score of 8.8 and 8.9 does not make much of a difference on whether they will watch the movie, or for producers if they will produce the movie. Therefore, we would like to define an adequate threshold that can determine whether a movie is “good” or it is “bad” using a classification method.

The classification method we ended up utilizing was Logistic Regression. The goal for classification is to choose a reasonable threshold rather with relatively low classification error. Additionally, for recommending movies to people or producers, we probably have higher tolerance of type II error (false negative) rather than type I error (false positive) since if we predict a potential movie as “good” to a producer but ended up being not so good, the producer loses reputation. We utilized cross validation on different threshold and the results are shown in the plot. Red line indicates classification accuracy, blue line indicates type I error and green line indicates type II error. From the plot, we find a threshold in between 5 and 5.5 as an ideal threshold.

Results and Conclusion

Method	**MSE**	**Method**	**MSE**
OLS	0.8997	CART	0.6818
OLS with reduced feature	0.8874	Random Forest	**0.5176**
Ridge	0.9137	Linear SVM	0.8566
LASSO	0.8726	RBF SVM	0.8362

This project focused on developing a method to accurately predict IMDb movie ratings based significant film attributes. Overall, we determined that the random forest approach has proved to be the most formidable out of the 6 regression models that we ended up developing based on a MSE of 0.5176. We believe that the low MSE for this approach provides a strong indication that our predictions are credible. Specifically, I believe that producers and film companies would definitely consider 0.5 error to be useful as our estimate predicts in the ballpark of the true value. Additionally, as mentioned previously, most producers or viewers do not see much of a difference between close ratings such as 5.6 and 5.8. As a result, we were able to develop an adequate threshold based on classification methods that is able to generalize predicted ratings as either a ‘good’ movie or a ‘bad movie’, which will be extremely helpful for producers in identifying whether a certain movie is worth taking a shot on. Besides the Random Forest, I believe that utilizing models SVMs and Regression Trees could possibly useful as well in predicting accurate movie ratings; however the main concern falls under the fact these models have a higher chance of overfitting the data overall. Finally, we were able to identify 17 key features that play a large role in determining the final rating of film, with the most significant features being budget, release year, user votes, and revenue.

Though our model seems to be accurate in many ways, there are always methods to improve the current procedure. One future addition that could be very useful would be to perform quantile regression on the data. This would be helpful for producers or film companies who would like to make films of the highest qualities by providing them information on what types of features differentiate a 9.0 rating movie from a 7.0 rating movie, and whether it is worth risk and the cost to pursue that goal.

References

1. ORIE 4741 Lectures by Professor Madeleine Udell and Professor Sumantha Basu
2. P-Value Theory: https://rstudio-pubs-static.s3.amazonaws.com/119859_a290e183ff2f46b2858db66c3bc9ed3a.html
3. Ridge vs Lasso Regression Theory: https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/
4. SVM Resource 1: https://www.mathworks.com/help/stats/understanding-support-vector-machine-regression.html#buyrzdd
5. SVM Resource 2: http://kernelsvm.tripod.com/
6. SVM Resource 3: http://www.saedsayad.com/support_vector_machine_reg.htm
7. SVM Resource 4: http://www.cs.cornell.edu/courses/cs4780/2017sp/lectures/lecturenote09.html
8. Regression Trees Source: http://www.stat.cmu.edu/~cshalizi/350-2006/lecture-10.pdf
9. Major Film Studio Source: “Major Film Studio.” *Wikipedia*, Wikimedia Foundation, 2 Dec. 2017, en.wikipedia.org/wiki/Major_film_studio.