Predicting the IMDB rating of a movie before its release

Kimia Esmaili – 610398193

Dr. Arta Jamshidi

University of Tehran - College of Mathematics, Statistics and Computer Science

Final Project Explanation – Mathematics Laboratory

Feb. 2023

Abstract:

To be honest, the world we live in, relies highly on capital. Hence, industries with greater income and more stable financial flow, have always been a significant part of the society and the focus have always been on maximizing the profit of these fields. I took that concept and applied it to movies for this model. The idea is that artists in the movie industry can utilize this model to predict how well a movie will be received by viewers; thus, I focus on IMDb rating as the target, since it shows the commercial success and popularity of a movie among the norm of the society, unlike Metacritic's rating system or Rotten Tomatoes's Tomatometer which consider the artistic value of a movie and how well it was received by the critics. I used mathematical analysis in my code which would yield a more ample and efficient outcome.

Background:

In its entirety, this project explored a few critical skills required of a data scientist for the coding part:

- Web scraping (requests, HTML, Beautiful Soup)
- EDA (pandas, numpy)
- Linear regression (scikit-learn)
- Data visualization (seaborn, matplotlib)

Step 1: Data acquisition & cleaning

As a quick note, IMDb has an API available to download bulk data, but a primary requirement for this project was to obtain data through web scraping; so, I went along and got the information from IMDb using requests and Beautiful Soup. Requests is the module required to take the webpage and turn it into an object in python. Beautiful Soup takes that object, which is the HTML information behind the webpage, and makes searching and accessing specific information within the HTML text easy. You really need both in order to fully complete the process of web scraping.

On the IMDb page, I used the advanced search feature to access titles between 2000 and 2020. The results spanned thousands of pages and each page held the titles and links to 100 movies. Upon further inspection, I noticed the URL contained the phrase: '*start=1'.* Increasing this start number by 100 would flip through each page. With a helper function, I used requests and Beautiful Soup to pull the links for each page and returned a list of those links.

To utilize that list of movie hyperlinks, I created another function to extract as much data as I could from each page. This function took in a link and returned a dictionary containing the following information: title, IMDb rating, the number of IMDb raters, MPAA rating, genres, directors, writers, top three stars, initial country of the release, original language of the release, release date, budget, opening weekend USA, gross USA, cumulative worldwide gross, production companies, and runtime.

As part of the EDA, some data had to be cleaned. This consisted of turning any numerical value from a string into an integer. Runtime had to be converted into minutes, all of the monetary values needed commas and dollar signs removed, and the release date had to be converted into datetime. Additionally, categories that contained lists needed to be converted from strings into actual python lists (genres, directors, stars, production companies). The retrieval function did most of this cleaning, but after putting the data into a DataFrame, some other cleaning was necessary.

With over 2,000 movies in a DataFrame, I needed to do some more processing to get a functional DataFrame for modeling. This meant dropping movies without information on budget, movies with a budget below $1,000, and movies with a sum of raters under 1,500. In regards to that last requirement, movies with a low number of raters proved to report the more extreme movie ratings (movies leaning towards a perfect 10 or a big goose egg). All in all, I ended up with a DataFrame consisting of over 1,100 movies. Now it's time to start modeling.

Pairplots: Before moving on to the next section, I'd like to mention pairplots. Pairplots is a great visualization tool for exploring relationships within the data and informing where to start for an MVP. It seems like a lot of information, but when you format your DataFrame with the first or last column being the target, it is a lot easier to interpret all of this information. For this pairplot, the plots in the first column show relationships between the independent variables and the target. Although I did not use most of the numerical data, it is obvious that there are linear and exponential relationships, which can easily inform where to start modeling.

Step 2: Models and features

It is important to note that another requirement for this project was the use of linear regression, so the models I experimented with were linear regressions and ridge regressions and R-squared for the error. With such a large number of features available, it took me a bit of time to sort out each feature.

- Getting familiar with out mathematical tools:

- **R-squared**

Error metrics enable us to evaluate the performance of a machine learning model on a particular dataset.

There are various error metric models depending upon the class of algorithm.

We have the Confusion Matrix to deal with and evaluate Classification algorithms.

While R square is an important error metric to evaluate the predictions made by a regression algorithm.

R-squared is a regression error metric that justifies the performance of the model.

It represents the value of how much the independent variables are able to describe the value for the response/target variable.

Thus, an R-squared model describes how well the target variable is explained by the combination of the independent variables as a single unit.

The R squared value ranges between 0 to 1 and is represented by the below formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{m} (X_i - Y_i)^2}{\sum_{i=1}^{m} (\bar{Y} - Y_i)^2}$$

- The numerator is the sum of squares of the residual errors.
- The denominator It represents the total sum of the errors.

It is evident that the higher the R square value, the better the predicted model.

- **Ridge regression:**

There are two things we would want to remember. One is that we don't like overfitting. In other words, we always prefer a model that catches general patterns**.** The other is that our goal is predicting it from new data, not specific data. Therefore, model evaluation should be based on new data (testing set), not given data (training set)**.** Also, I will use the following terms interchangeably.

- Independent Variable = Feature = Attribute = Predictor = X
- Coefficient = Beta = β
- Residual Sum of Squares = RSS

Why And Why Not OLS:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon \quad \longleftarrow \quad \textit{True Model}$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p \quad \longleftarrow \quad \textit{Estimated Model}$$

$$\hat{\beta}^{OLS} = (X^T X)^{-1} X^T Y \quad \longleftarrow \quad \textit{Matrix Equation}$$

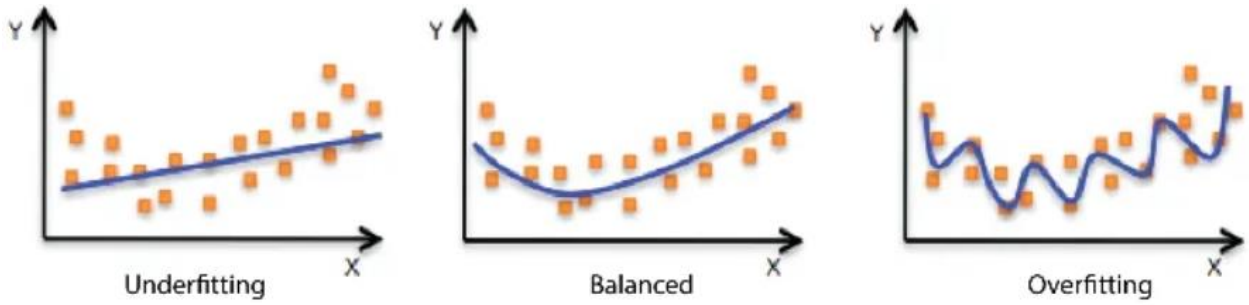
Least Square Method finds the *Best* and *Unbiased* Coefficients

You may know that least square method finds the coefficients that best fit the data. One more condition to be added is that it also finds the unbiased coefficients. Here unbiased means that OLS doesn't consider which independent variable is more important than others**.** It simply finds the coefficients for a given data set. In short, there is only one set of betas to be found, resulting in the lowest 'Residual Sum of Squares (RSS)'. The question then becomes *"Is a model with the lowest RSS truly the best model?"*.

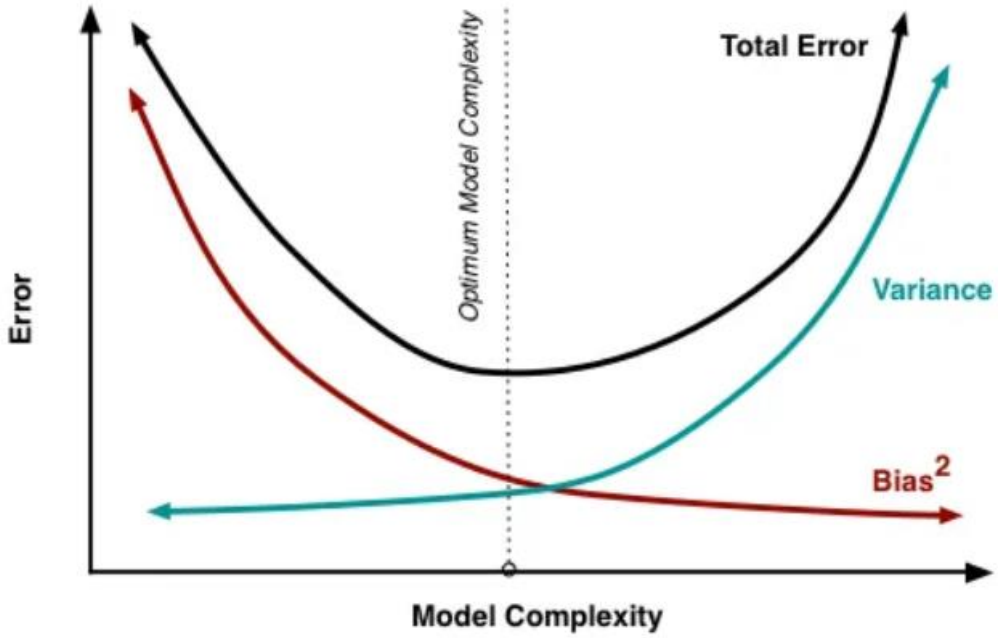
Bias vs. Variance

The answer for the question above is *"Not really"*. As hinted in the word 'Unbiased', we need to consider 'Bias' too. Bias means how equally a model cares about its predictors. Let's say there are two models to predict an apple price with two predictor 'sweetness' and 'shine'; one model is unbiased and the other is biased.

First, the unbiased model tries to find the relationship between the two features and the prices, just as the OLS method does. This model will fit the observations as perfectly as possible to minimize the RSS. However, this could easily lead to overfitting issues. In other words, the model will not perform as well with new data because it is built for the given data so specifically that it may not fit new data.



The biased model accepts its variables unequally to treat each predictor differently. Going back to the example, we would want to only care about ‘sweetness’ to build a model and this should perform better with new data. The reason will be explained after understanding Bias vs. Variance.



It can be said that bias is related with a model failing to fit the training set and variance is related with a model failing to fit the testing set. Bias and variance are in a trade-off relationship over model complexity, which means that a simple model would have high-bias and low-variance, and vice versa. In our apple example, a model only considering ‘sweetness’ would not fit the training data as much as the

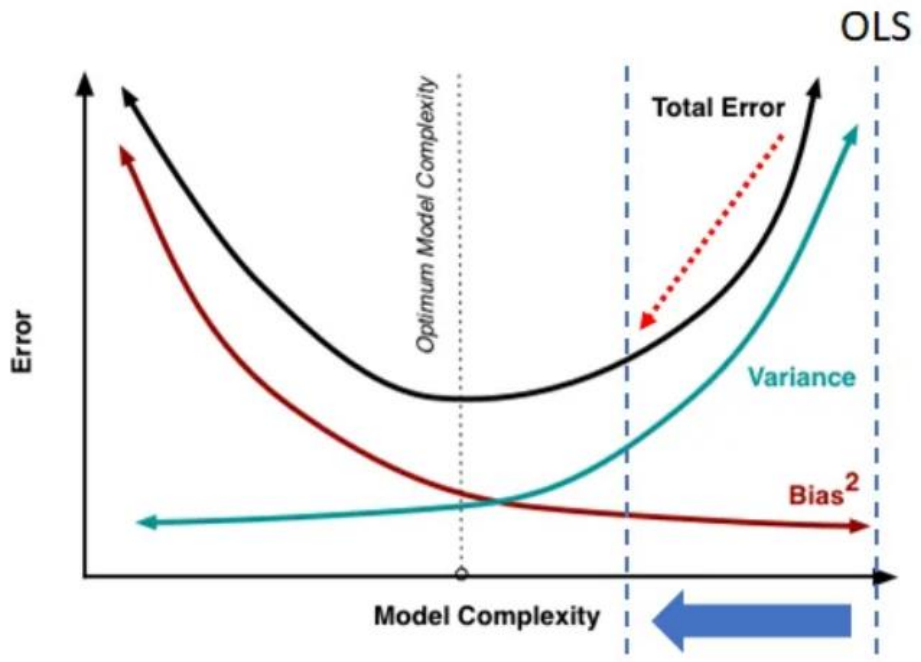
other model considering both ‘sweetness’ and ‘shine’, but the simpler model will be better at predicting new data.

This is because ‘sweetness’ is a determinant of a price while ‘shine’ should not by common sense. We all know this as a human but mathematical models do not think like us and just calculate what’s given until it finds some relationship between all the predictors and the independent variable to fit training data.

**Note*: We assume that ‘sweetness’ and ‘shine’ are not correlated

Where Ridge Regression Comes Into Play

Looking at *Bias vs. Variance* figure, the Y-axis is ‘Error’ which is the ‘Sum of Bias and Variance’. Since both of them are basically related with failing, we would like to minimize those. Now taking a second look at the figure closely, you will find that the spot the total error is lowest is somewhere in the middle. This is often times called ‘Sweet Spot’.



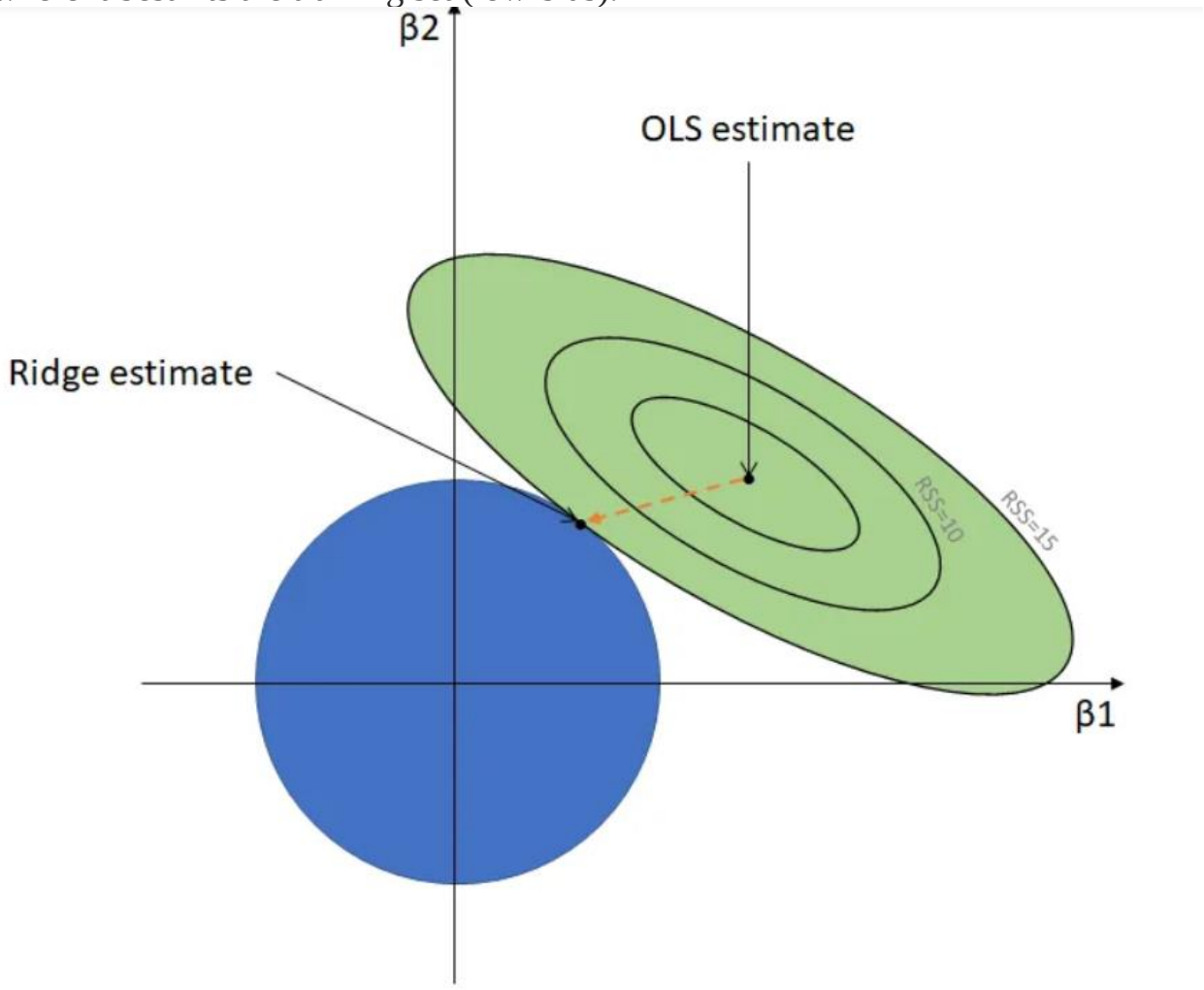
Let’s recall that OLS treats all the variables equally (unbiased). Therefore, an OLS model becomes more complex as new variables are added. It can be said that an OLS model is always on the rightest of the figure, having the lowest bias and the highest variance. It is fixed there, never moves, but we want to move it to the sweet spot. This is when ridge regression would shine, also referred to as *Regularization*. **I**n ridge regression, you can tune the lambda parameter so that model coefficients change. This can be best understood with a programming demo that will be introduced at the end.

Geometric Understanding of Ridge Regression

Many times, a graphic helps to get the feeling of how a model works, and ridge regression is not an exception. The following figure is the geometric interpretation to compare OLS and ridge regression.

Contours and OLS Estimate

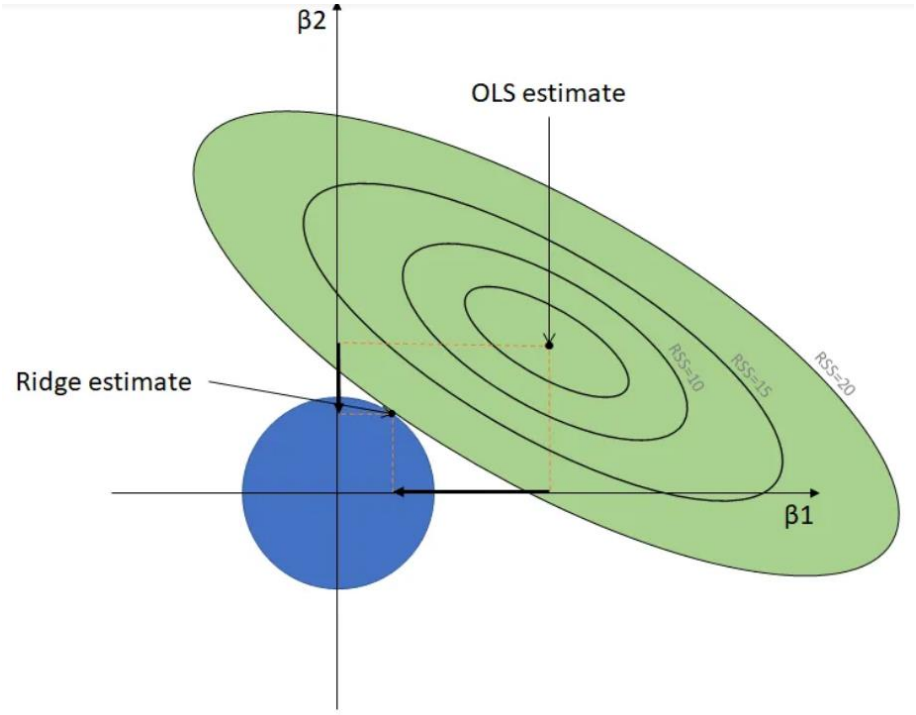
Each contour is a connection of spots where the RSS is the same, centered with the OLS estimate where the RSS is the lowest. Also, the OLS estimate is the point where it best fits the training set (low-bias).



Circle and Ridge Estimate

Unlike the OLS estimate, the ridge estimate changes as the size of the blue circle changes. It is simply where the circle meets the most outer contour. How ridge regression works is how we tune the size of the circle. The key point is that ***β*'s** change at a different level**.**

Let's say $\beta 1$ is 'shine' and $\beta 2$ is 'sweetness'. As you can see, ridge $\beta 1$ relatively drops more quickly to zero than ridge $\beta 2$ does as the circle size changes (compare the two figures). The reason why this happens is because the β's change differently by the RSS. More intuitively, the contours are not circles but ellipses positioned tilted.



Ridge β's can never be zero but only *converge* to it, and this will be explained in the next with the mathematical formula. Although a geometric expression like this explains a main idea pretty well, there is a limitation too that we can't express it over 3-dimension. So, it all comes down to mathematical expressions.

Mathematical Formula

We've seen the equation of multiple linear regression both in general terms and matrix version. It can be written in another version as follows.

$$\underset{\beta \in \mathbb{R}}{argmin} \sum [y_i - \hat{y}_i] = \underset{\beta \in \mathbb{R}}{argmin} \sum [y_i - (\beta_0 + \beta_1 x_1 + \beta x_2 + \cdots + \beta x_p)]^2$$

Here *argmin* means 'Argument of Minimum' that makes the function attain the minimum. In the context, it finds the β's that minimize the RSS. And we know how to get the β's from the matrix formula. Now, the question becomes "What does this have to do with ridge regression?".

$$\beta_0^2 + \beta_1^2 + \cdots + \beta_p^2 \leq C^2$$

Again, ridge regression is a variant of linear regression. The term above is the ridge constraint to the OLS equation. We are looking for the β's but they now must meet the above constraint too. Going back to the geometric figure, the C is equivalent to the radius of the circle, thus, the β's should fall in the circle area, probably somewhere on the edge.

Vector Norm

We still want to understand the very first equation. To do so, we need to brush up on vector norm, which is nothing but the following definition.

$$\|B\|_2 = \sqrt{\beta_0^2 + \beta_1^2 + \cdots + \beta_p^2}$$

The subscription 2 is as in 'L2 norm'. We only care about L2 norm at this moment, so we can construct the equation we've already seen. The following is the simplest but still telling the same as what we have been discussing. Notice the first term in the following equation is basically OLS, and then the second term with lambda is what makes ridge regression.

$$\hat{\beta}^{ridge} = \underset{\beta \in \mathbb{R}}{argmin} \|y - XB\|_2^2 + \lambda \|B\|_2^2$$

What We Really Want to Find

The term with lambda is often called 'Penalty' since it increases RSS. We iterate certain values onto the lambda and evaluate the model with a measurement such as 'Mean Square Error (MSE)'. So, the lambda value that minimizes MSE should be selected as the final model. This ridge regression model is generally better than the OLS model in prediction**.** As seen in the formula below, ridge β's change with lambda and becomes the same as OLS β's if lambda is equal to zero (no penalty).

$$\hat{\beta}^{ridge} = (X^T X + \lambda I)^{-1} X^T Y$$

Why It Converges to Zero But Not Becomes Zero

Deploying the matrix formula we saw previously, the lambda ends up in denominator. It means that if we increase the lambda value, ridge β's should decrease. But ridge β's can't be zeros no matter how big the lambda value is set. That is, ridge regression gives different importance weights to the features but does not drop unimportant features.

$$(\cdots + \tfrac{1}{\lambda} + \cdots) X^T Y$$

- **Now we take a look at the project:**

First, I decided to take the easy route by conducting a linear regression model with runtime as my sole feature and IMDb rating as the target. This resulted in an R^2

value of 0.2687. Honestly, I was fairly excited to get any number above zero, so I was ready to dive in to the rest of the data.

For MPAA rating and genre, I created dummy variables to add to the DataFrame and got an R^2 of 0.3997. As for directors, writers, stars, and production company, I created a list of the most frequently occurring players in each of those categories and created dummy variables for the top contenders. If a director only appeared once in my data, then that director's weight (or coefficient) would be a direct result of that specific film's rating, so having players with multiple rows of data would give the model more information to create a better informed coefficient.

To get a little more creative, I took the release date and made a 'release month' feature. In the same vein, I took the release date and created another feature that determined the years since the movie was released. It may not have been the most relevant feature, but I was excited to experiment with datetime information.

Having loaded the features into a model, a resulting R^2 of 0.4751 seemed promising, but the next step was to rigorously test the model with cross validation.

Step 3: Testing and training / the results

Although linear regression was getting the job done, I knew I wanted to compare the coefficients of the model, and using a ridge regression was a great way to force myself to scale the inputs and try a different approach to creating a model.

The final model resulted in an R^2 of 0.432 and a mean absolute error of 0.64. This is a fairly low R^2, but some sources mentioned an R^2 below 0.5 for predicting human

behavior is expected. Additionally, the plot to the left of predicted ratings vs. actual ratings provided more confidence in the model, as there is some sort of linear relationship between the two. Also, the movies with highest residuals had either a low number of ratings, or were movies like *Cats* and *The Emoji Movie*. These particular movies have good stats behind them, but the public just did not receive them well, which is a hard metric to incorporate into this model.

It's also important to look at the coefficients associated with each feature. As seen in the plot on the left, runtime, years since release, and budget were all big players in the model, with some genres and writers being up there as well. That's the beauty of the ridge regression: being able to use the coefficients to determine the weight of a specific feature.

In the end, I had a model that predicted IMDb rating with an R^2 of 0.432, significantly better than just predicting with the mean, and an MSE of 0.64, which means the prediction was liable to be wrong by 0.64 points in either direction.

More additional explanation is available in the jupyter notebook of the project.

Summary:

The things I did in this project:

- Created my own dataset through scraping the web for information
- Explored the dataset and cleaned up anything that was off

- Developed an MVP to have a working model at any given moment
- Iteratively improved that model to get a better product with each feature
- Visualized the validity of my model and what contributed to the rating of a movie

References:

S. Gaenssle, O. Budzinski, and D. Astakhova, “Conquering the box office: Factors influencing success of international movies in russia,” 2018.

M. Saraee, S. White, J. Eccleston et al., “A data mining approach to analysis and prediction of movie ratings,” Transactions of the Wessex Institute, pp. 343–352, 2004.

D. Chicco1, M. J. Warrens, G. Jurman, “The coefficient of determination R-squaredis more informative than SMAPE, MAE, MAPE, MSE and RMSE in regressionanalysis evaluation,” International Journal of Computer Scienceand Network Security (IJCSNS), vol. 16, no. 8, p. 127, 2016.

S. Pramod, A. Joshi, and A. Mary, “Prediction of movie success for real world movie dataset,” Int. J. of Advance Res., Ideas and Innovations in Technol, vol. 3, no. 3, 2017.

C. Zhang, “Research on IMDB Film Score Prediction Based on Improved Whale Algorithm”Film College of Changchun Guanghua University, China. Elsevier, 2022.