Stat 410 Aidan's Movies

Aidan Gerber

April 29th, 2022

Project statement

The purpose of my analysis is to determine what movies I will enjoy the most. I will do this by creating a model with the response variable of how much I liked a movie out of 100 and predicting it with data on various movie reviews, genres, cast and crew, and more. I am doing this so that I can figure out which movies I will want to watch because I can be confident that I will enjoy them.

Data description

The data was collected over time by me. A partial dataset in json form is available on my website and a full version is available on the project website. Data from TMDB was collected using their API. Data from IMDB was collected by web scraping their pages for awards and rating distributions. My rating and metacritic ratings are collected by me. This data is relevant since my ratings would be related to ratings from critics as well as attributes of movies like genres.

Exploratory data analysis

None of the variables are too highly related to my rating. The highest correlations are to imdb_rating and tmdb_rating. The other variables above 0.6 correlation are variables that are highly correlated with imdb_rating like imdb_arithmetic_mean, imdb_us_rating, imdb_top_1000_rating, and imdb_not_us_rating. Moreover, imdb_rating and tmdb_rating are also correlated (r=0.9). Metacritic_rating and imdb_median have weaker correlations, r=0.75 and r=0.76, respectively. A similar story holds true for the count variables imdb_count, imdb_us_count, and imdb_not_us_count. The correlation between log(imdb_count) and log(tmdb_count) is even r=0.95. Most variables in the dataset work

linearly like my_rating and imdb_rating. Since my_rating is on a 0 to 100 scale, other un-transformed variables within a specific range work best such as imdb_rating between 1 to 10, tmdb_rating between 1 to 10, or metacritic_rating between 0 and 100. To create the main dataset for analysis, I interpolated null values rather than removing them. The only variable with null values was metacritic_rating since some movies are not on Metacritic. To interpolate metacritic_rating, I created a basic MLR to predict them from movie characteristics that are not related to me or my ratings. The next step was to make dummy variables for MPAA ratings. The most common MPAA ratings for movies that I have seen are:

PG-13	PG	R	G	NR
209	179	134	27	21

After that, I wrote a function to check if a person of a specific id appeared in a movie and return 1 or 0. This combined the separate credits and movies tables. Credits are 1-to-1 related to both movies and people so I checked those columns. Next, I created dummy variables for both genres and crew. I chose every genre that I have seen more than 100 movies from.

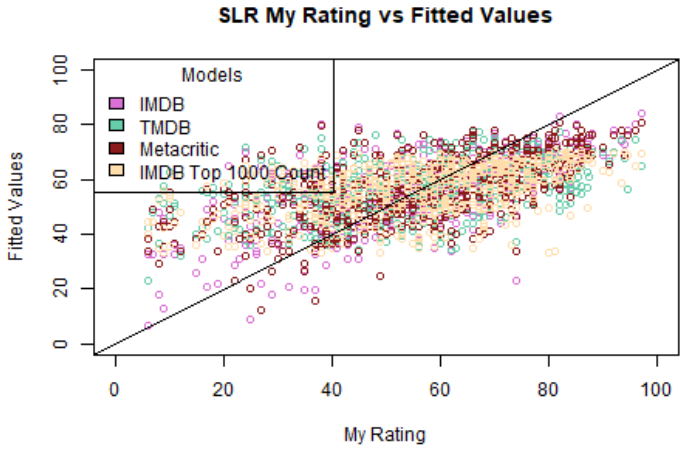
Comedy	Adventure	Action	Drama	Family	Science.Fiction	Animation	Thriller
266	239	213	167	166	132	126	102

Choosing cast and crew members was more difficult. I selected people that I thought had the ability to make movies more than the sum of their parts. I decided to choose many more people than I expected to be significant and have them whittled down through variable selection methods. An extra step was creating a caching method since creating continuous_movies takes a few minutes each time (for just cast and crew, it performs 1227168 calculations). Looking at the correlation matrix of my_rating to all the variables in the final dataset, I noticed that imdb_rating is most significant. Barely behind are tmdb_rating,

imdb_arithmetic_mean, imdb_top_1000_rating, imdb_us_rating, imdb_not_us_rating, imdb_rating_percentile, and tmdb_rating_percentile. None of these are surprising and each of them are correlated with each other. There were also reasonably strong correlations between my_rating and imdb_count with less strong ones to award_count, runtime, and revenue. In terms of genres, the highest positive correlation was to drama with large negative correlations to comedy and family. For MPAA ratings, R has a positive correlation and PG-13 has a negative correlation. For people, the highest positive correlations were to John Williams, Wally Pfister, Christopher Nolan, Harrison Ford, George Lucas, and Matt Damon.

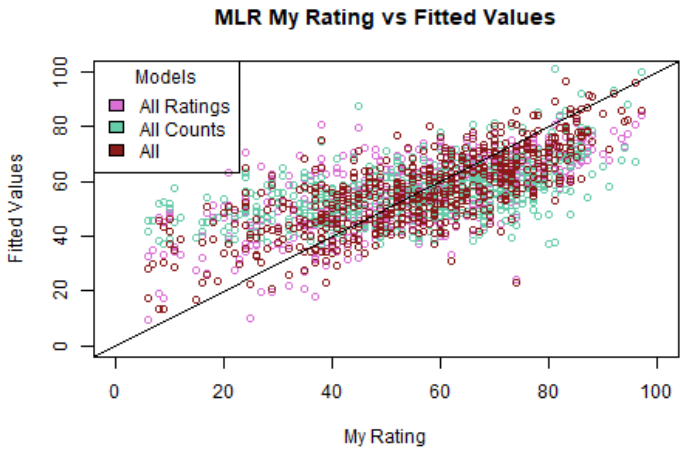
Data analysis

SLR



Each of these SLRs have substantial flaws. IMDB ratings are most predictive with an adjusted r squared of 0.43, higher than the other models. Each of the models have relatively low betas. This is because RSS highly penalizes particularly far values so it places values toward the center of the range. This is especially apparent for Metacritic which has a beta hat of 0.61. This means that the total range of predictions is 23.16 to 77.07 despite my rating range going from 6 to 97.

MLR



The full MLR is an improvement but all ratings and all counts models are not particularly effective. Looking at the ratings MLR, the only significant term is the IMDB rating and the adjusted r squared only increases 0 between the 2 models. Although many of the counts are significant, the overall model r squared is quite low at 0.29. This occurs because of the uncapped counts that I mentioned in data analysis.

Ridge and Lasso

Adjusted R Squared at 1 standard error for ridge and lasso:

Lasso	Ridge	MLR	Full.Lasso	Full.MLR
0.47	0.47	0.46	0.56	0.5

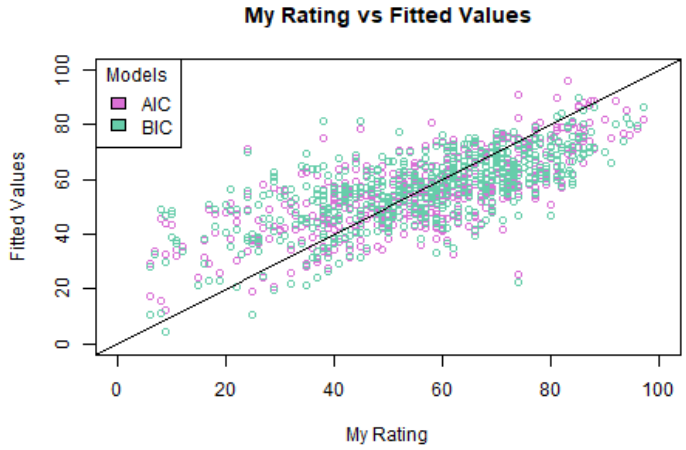
Lasso and ridge result in a slight improvement over the MLR since they take advantage of the bias variance tradeoff. Looking at the beta values, the different models change the coefficients a lot despite getting similar results. MLR values imdb_count and imdb_rating much more than ridge or lasso. Lasso actually removes sqrt(award_count) to not much of a detriment.

When comparing the full models, fit_lasso is highly predictive. Out of the 65 initial variables,

it selects 12 variables. Each variables lasso selects has positive coefficients, so the interpretation of the model makes substantially more sense than the full model. However, the lasso model does include multiple versions of similar terms, indicating potential overfitting. For example, it includes imdb_rating, imdb_top_1000_rating, imdb_us_rating, imdb_not_us_rating, and imdb_rating_percentile — all of which are very similar. It repeats this flaw by including imdb_count, imdb_top_1000_count, and log(imdb_count). The only dummy variable lasso includes is is_john_williams.

Steps with AIC and BIC

Because the p-value for the full model is significant in comparison to the null model, it makes sense to continue with AIC and BIC to find a significant model. The use of AIC and BIC is to penalize adding more parameters and select only useful ones.

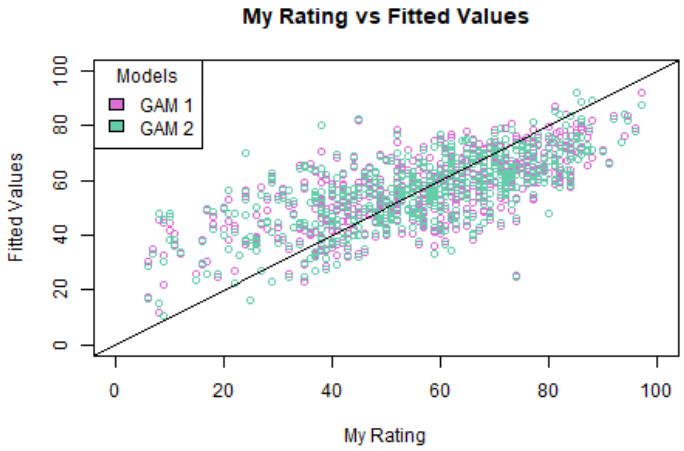


AIC and BIC had very different results. The AIC resulted in my highest adjusted R^2 of any model (0.51). The final AIC selected 20 coefficients. This is very different than the BIC which selected 4. Only 3 variables in the AIC have negative betas, is_jack_nicholson, is_adventure, and imdb_rating. This is unique to this model and the full model and is possible because lots of people have highly positive coefficients and log(imdb_count) is also highly positive. Moreover, imdb_rating does not have a significant p-value in this model. The

BIC chooses a much sparser with just imdb_rating, log(imdb_count), and is_john_williams. Based on these selections, BIC avoids the flaw of lasso including multiple highly correlated variables. Both full lasso and BIC include is_john_williams as the only dummy variable. John Williams is not only an excellent composer but links many of my favorite movies such as Indiana Jones and Star Wars while only appearing in 2 movies I rated below a 60: Close Encounters of the Third Kind and The Lost World: Jurassic Park. Even with such different models, both the AIC and BIC had similar adjusted r squared values and their adjusted r squared values were generally in line with other models.

General Additive Models

Comparison



GAM fit plots withheld for spacing reasons. Based on the GAM fit plots, imdb_rating and metacritic_rating are linear. Tmdb_rating is linear until the rating reaches about 6 before increase in slope between 6.5 and 7.5 before flattening. This means that the impact of an increase in tmdb_rating from 7.5 to 7.6 is expected to have a larger impact on my rating than an increase from 6.4 to 6.5. Imdb_count has a very steep slope until it reaches around 500,000 before flattening out. This impact is fixed by using a log transformation Finally, runtime actually flips. When runtime is under an hour, an increase in runtime leads to an

expected increase in my rating. However, as runtime passes 150 minutes, a higher runtime means an expected decrease in my rating. Imdb_arithmetic_mean is a new addition to this model and actually has a negative beta model. This exemplifies the slight difference in the way the advertised imdb_rating is calculated vs the arithmetic mean. Imdb_rating is calculated using a secret formula to prevent review bombing and is a weighted average. This reveals that using the weighted mean is more powerful than the arithmetic mean in this case and that the weighted average used by IMDB is useful.

For budget, there is a quick slope upward at the beginning revealing that a percentage increase in budget is more important than a dollar amount increase. However, this should be taken with a grain of salt as there are flaws to using budget as a prediction metric. Firstly, movies have been made at all different times and budgets are not normalized for inflation. This harms the predictive power of using budget. Moreover, some movies do not have public budget information which is related to their popularity. The mean IMDB count for a movie with a budget that is not 0 is 431549.13 while for movies with a budget that is 0 the mean count is 38796.13, 11.12 times smaller.

Summary and discussion

Ultimately, the model that I ended up liking the most was the MLR chosen by BIC with imdb_rating, log(imdb_count), and is_john_williams (adj r squared=0.47). This is very similar to the GAM version of the model which includes a smoothed version version of imdb_rating and imdb_count, a standard dummy variable of is_john_williams and an interaction term between is_john_williams and log(imdb_count). This resulted in an adjusted r squared of 0.47. Since the simple MLR found by BIC is much simpler and has a nearly identical r squared, it is the better choice.

My largest conclusion in this project is that predicting my ratings is difficult to do accurately and that getting an adj r squared of 0.47 is good, especially since the model is interpretable and does not overfit. I reached this conclusion since I fit many different models and ended

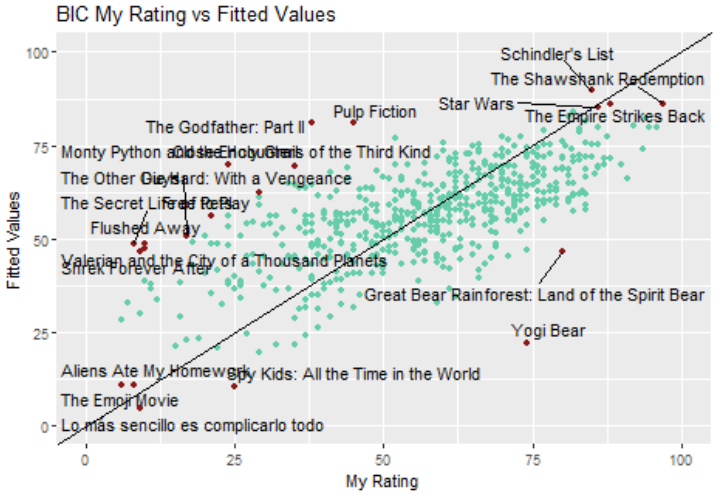
up with similar adj r squareds despite different betas. Even with these varied betas, models consistently overpredicted low values and underpredicted high values. This makes sense considering each of the things that I am predicting based on are aggregate metrics. When trying to turn these aggregate metrics back to an individual prediction, the model hedges and predicts values generally closer to the center. This allows the model to miss by less when it is not particularly close.

Interpretation

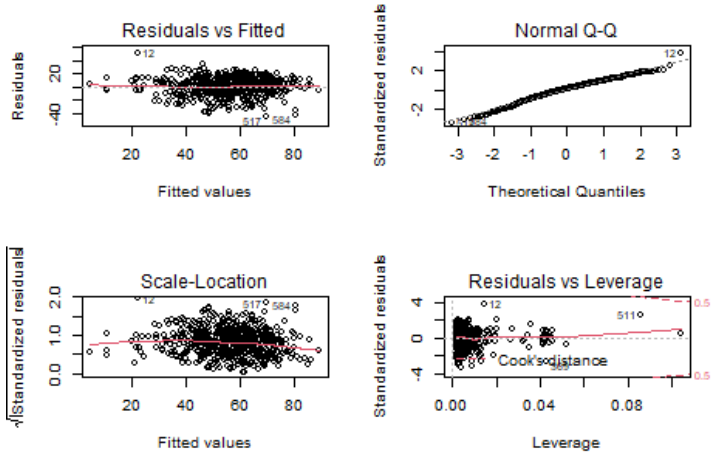
A movie with an IMDB rating of 0, that has been seen by 1 person, and does not have John Williams would be expected to have a rating of -53.75. For each 1 unit increase in imdb_rating, holding all else equal, my expected rating increases by -53.75. For each 1 unit increase in log(imdb_count), holding all else equal, my expected rating increases by 10.96. If John Williams is credited in the movie, holding all else equal, my expected rating increases by 2.57.

Fit BIC Largest Residuals

First	Second	Third	Fourth	Fifth
Yogi Bear: 51.66	Great Bear Rainforest: Land of the Spirit Bear: 33.46	The Smurfs: 27.28	Robots: 27.09	Misha and the Wolves: 26.46
Monty Python and the Holy Grail: -45.99	The Godfather: Part II: -42.91	The Secret Life of Pets: -40.82	Flushed Away: -38.7	Shrek Forever After: -37.66



Diagnostics



Out of the fit diagnostics, residuals vs fitted and residuals vs leverage both look good. The normal QQ plot tails off slightly below for higher theoretical quantiles but stays on the line for most of the points, This tailing off is because my ratings are bounded between 0 and 100 while a normal distribution is not. This result is not worrying and errors can be treated as Gaussian. The scale location plot contains a slightly downward slope at higher fitted values. This is okay because the data is much sparser at those values so the final model assumptions for MLR still hold.

Predictions

Movie	imdb_rating	imdb_count	is_john_williams	fit	lwr	upr
The Lord of the Rings 2001	8.8	1794349	0	79.7	52.9	107
Inglorious Basterds	8.3	1389883	0	73.6	46.8	100
The Silence of the Lambs	8.6	1381370	0	76.8	50.0	104
Saving Private Ryan	8.6	1342619	1	85.3	58.1	113
Titanic	7.9	1130852	0	68.7	41.9	95
Braveheart	8.4	1014965	0	73.8	47.1	101
Boyhood	7.9	350258	1	74.2	47.0	101
The Hottie & the Nottie	1.9	38184	0	-5.8	-33.3	22
The VelociPastor	5.0	5695	0	23.3	-3.6	50

These prediction intervals are computable since my final model was a Gaussian MLR. They are mostly reasonable but fall within very large ranges. On the high end, the predictions get above 100 and on the low end they get below 0. I feel confident that I would rate nearly all of these movies within the given range; however, the ranges encompass over half of the rating space. These ranges means the model must hedge because it cannot continuously predict values of 90 for movies with ranges between 63 and 117 because the range on the upper end is impossible.

Overall, predicting my movie ratings is very difficult to make highly accurate, but it is certainly possible to do much better than random guessing, and it is possible to get a relatively accurate model with high interpretability. I can now use the model that I found to have a better idea of which movies to watch and which to avoid.

Code and data

Datasets available on Github, the hosted website, movie.csv, credit.csv, continuous_movies.csv, and pdf form.