Supervised Learning: Movie Rating Predictions

Parker Hague
Oklahoma State University

Dax Jones
Oklahoma State University

Steven Mikels
Oklahoma State University

Abstract

It's notoriously difficult to predict the success of a movie. Movies with enormous budgets such as John Carter and The Lone Ranger were filled with well-known actors backed by respected publishing companies and flopped, while films with miniscule budgets such as The Blair Witch Project have gone on to make enormous impacts on cinema and culture for years past their release. Some actors like Dwayne 'The Rock' Johnson seem to guarantee a movie performance, while others such as Nicholas Cage often star in several terrible movies for each well-received picture they decide to take a role in. The question of what makes a good movie has been debated for decades... Until now!

We are building three different machine learning models to make predictions about the critical rating of a movie: a feed forward neural network, Naive Bayes, and a recurrent neural network model. These models are fed using data from idmb, after we have transformed it to be in a more readily legible form and pruned it of extraneous points of data. Using our three models, we will determine which criteria are most heavily linked with the critical reception of a movie, and which combinations of genres, actors, and other factors may lead to surprising amounts of success. Then, armed with this information, we will become the greatest filmmakers of the twenty-first century, and create several theoretical movies of our own and predict how they would do at the box office.

1. Introduction

The goal of our models is to take the genre and cast of a movie to predict the critical rating of the movie. We are drawing our data from an imdb dataset of over forty-five thousand movies from Kaggle. After extracting the actors and genres and converting them into a more useable form, we will feed it into three different models: a feed forward neural network, a recurrent neural network, a naïve bayes model. After using a model selection method to decide which of our models is the best, we will then test different combinations of actors and genres to see which theoretical films box office poison and which pairings are destined for the Oscars.

2. Background/Related Work

Our primary paper to establish a bit of a background will be 'Identification of Key Films and Personalities in the History of Cinema From a Western Perspective' by Livio Bogolia and Ruggero G. Pensa. This paper will give us a lot of context into the general trends of western cinema (most of our data comes from western films as opposed to somewhere like Bollywood), and of prolific actors. By examining some of the most important movies in our data set, by comparing them by hand we can see what, at least in the popular mind, what makes a "good" movie. Another paper we will consult is 'Exploring Movie Construction and Production' by John Reich. Instead of looking at the make-up and reception of particular films that captured the popular imagination, this paper instead looks at the various aspects that go into making and classifying a movie. Ideally, this will help us better understand what the various different genres we are testing actually mean. Defining exactly the difference between an action and an adventure movie for instance could prove useful in deciding how we configure the testing for our models.

3. Approach

The problem of our model is to take the cast of a movie and its genre and use that information to predict the critical reception of the movie on a scale of one to ten. Before we begin building any models, we need to tend to our dataset. Unfortunately, while our dataset has a lot of entries which will allow us to divide it into a training and a validation set, it is not without issues. The first issue is that a lot of potential data points have a lot of movies that lack any information on it in the dataset. Chiefly, the example here would be budget.

We initially tested trying to use budget as part of our dataset, but it cut down the number of movies we would train on from forty thousand all the way to a little less than seven thousand. Other categories, such as the production company, directors, and production location had a similar effect on the amount of data we could use. While having more types of features in our machine learning models might have led to an overall better picture of the problem in theory, we believe it would sacrifice too much accuracy in

the model to have such a smaller amount of data to train on. This led us to narrow our focus to two different factors: the genre of the movie, and the highest billed actors in the film.

The second problem with our dataset was how the .csv files were formatted. Instead of having an individual column for each cast member, the file instead had a single "cast" column that contained everyone who worked on the movie, from actors, directors, producers, and stage techs. The genres were formatted in a similar manner, because some movies could have multiple genres; How to Lose a Guy in 10 Days, for example, was categorized as a Romance and as a Comedy. Additionally, the amount of cast members and genres was inconsistent; some movies had a single actor, while others had a cast of dozens. Our solution was to only grab the first genre of a movie, and the first five actors of a movie. If a movie had said only two actors or lacked a genre at all, the spot where a genre or actor would normally be filled with a value of zero. Since the names of genres and actors are strings, they were converted into a number and stored in a dictionary.

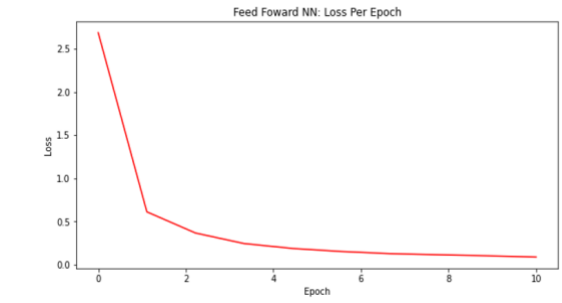
One key decision we had to make is if we wanted to yield categorical, or continuous results. A movie rating is normally a categorical result, but in our dataset, it is stored as a continuous result since it is a bunch of ratings averaged together; it is a number ranging from 0.1 to 10.0. However, there was some discussion to round each score to the nearest whole number and have eleven categories for movie scores to fit into to be more like a reviewer assigning a movie a score. The primary advantage of deciding to keep our results as continuous was ending up with more accurate results. If a film is a 7.5 for example, a guess of 7.0 or 8.0 would be the same distance apart, but in our model only an eight would be correct. This sort of inaccuracy in our results seemed to be undesirable.

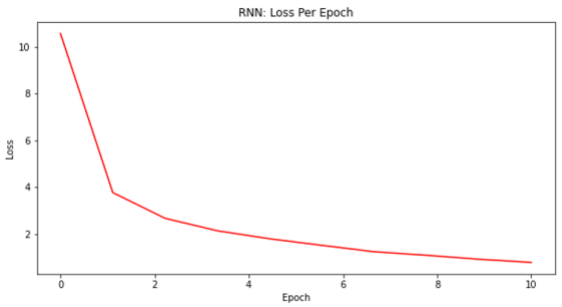
Yet, it wasn't as if picking categories had no advantages. It made the problem itself quite a bit easier to work with. In class we have dealt with more categorical problems than continuous ones, and there are far more resources online for testing with categories than for continuous results. Additionally, because our model would not be giving as a precise of a guess, it is likely it would end up more "accurate" after all our tuning is said and done, even if that is more due to the less precise nature of our guesses than any actual advantage of the model. Since we picked Naïve Bayes as one of our model types, we needed categorical data for at least one our models anyway. Since we already had this categorical model, we decided to keep our other models as using continuous data so we could see which type of approach would lend itself better to our stated problem.

The first model we decided to test is a feed-forward neural network model. While this type of simpler model may not be as accurate, that was fine with us. We think that having a simplistic "baseline" model is a good thing to include in our project. It gives us something good to compare our other results to, it is simple to construct and test, and it allows us to optimize and test tweaks in an iterative fashion quickly and reliably. Our second model is a recurrent neural network. RNNs "remember" earlier inputs due to their internal memory. This makes them excellent for working with sequential data and why an RNN seemed like a perfect fit for our problem. Lastly, we decided on a Naive Bayes model precisely because it is so different in comparison to the two other neural network models. In contrast with our other options, it is a generative, non-parametric model. By testing a model that is quite a bit different than our other two, we hope this makes it unlikely that all models will experience the same kind of variances so we can attempt to piece together a more holistic view of the problem.

Since we are using text, we also had to use embedding layer which allows us to translate our words and their relationships with each other into vectors. Since we have so many actors, using one hot encoding simply isn't practical; we'd have an enormous vector for each actor name that is not only way too big but very, very sparse. Additionally, embedding will help us capture the relationships between actors and genres better. While Tom Hanks might be in a lot of successful movies, maybe every single time he is paired up with Angelina Jolie it is a recipe for disaster. Similarly, maybe all of Brad Pitt's sci-fi movies are highly rated, but he is less successful in his children's movies. Because Keras does not allow for the use of more than one embedded layer at once, we had to concatenate our genres and actors into a single "sentence".

We ran into relatively few problems constructing the feed-forward and recurrent neural network models. After mastering the embedding layer, it was a simple matter of creating some layers and hitting go. However, the embedding layer posed a few issues; namely, that it changed the output dimension of our data. The recurrent neural network seemed to solve this problem on all its own and changed the matrix back to the appropriate size. For the feedforward neural network, we added a flatten layer right after the embedding. The loss of these two models over time is shown below.





Constructing the Naïve Bayes models presented a new issue, turning continuous data into categorical data. Luckily, our continuous data comes from categorical data. Ratings are always in a finite number of categories; you can only give them a whole number out of ten after all. It is only continuous because we are averaging a lot of reviews. As a result, simply rounding our numbers to the nearest whole number was enough to make categories.

Keras-Tuner Optimzations Tables

RNN			
Number of Models Tested	2770		
Elapsed Time	4:42		
Keras Tuner Parameters Tested	**Best**	**Second Best**	**Third Best**
Optimizer	RMSprop	Adam	SGD
Learning Rate	0.002	0.001	0.0001
Activation Function	relu	tanh	selu
Number of Embedding Neurons	450	350	400
Number of RNN Neurons	200	400	300
Number of Hidden Layers	3	2	4
Number of Neruons in Hidden Layers	20-80	20-100	20-100
Batch Size	512	256	1024

FFNN			
Number of Models Tested	2003		
Elapsed Time	6:21		
Keras Tuner Parameters Tested	**Best**	**Second Best**	**Third Best**
Optimizer	RMSprop	Adam	SGD
Learning Rate	0.0001	0.001	0.01
Activation Function	tanh	sigmoid	relu
Number of Embedding Neurons	450	500	600
Number of Hidden Layers	4	3	2
Number of Neruons in Hidden Layers	55-120	55-120	55-120
Batch Size	256	512	128

From here, most of our testing involved attempting to improve the accuracy of our models through tuning. To streamline the tuning process, we decided to use Keras's Keras Tuner. The Keras Tuner allowed us to give the model variable hyperparameters and the tuner tested all the combinations of those hyperparameters. For example, if we gave it three options for the optimizer and three options for the learning rate, then the tuner would test a total of nine different combinations of the model in order to see what the best hyperparameters are. We ran over 4,700 different combinations of hyperparameters across the feed forward and recurrent neural networks.

For every model, there were a lot of factors to consider such as from the number of layers and the neurons in them, the activation function, learning rate, optimizer, and batch size. Pictured before this paragraph is a table summary of the best hyperparameters for the two neural networks.

Here are the final results after using the test data on the trained model.

Model Peformance on Test Data

Model	Validation MSE or Accuracy
RNN	0.85646
FFNN	0.6698
Naïve Bayes (categorical)	86.87%

Also, included below is a summary of a few test inputs against the three different models and they're results for each input.

Input
['Drama DannyGlover RonPerlman LindaHamilton ZoeWeizenbaum DavidStrathairn']
['Drama PaulWalker PiperPerabo LambertWilson LindaCardellini ShawnHatosy']
['Comedy MollyShannon WillFerrell ElaineHendrix HarlandWilliams TomGreen']

Actual Rating	FFNN Predicted	RNN Predicted	Naïve Bayes Predicted
6.1	6.188	6.521	6
6.0	6.464	7.356	6
5.0	5.695	5.130	5

Out of our three models, we decided to select Naïve Bayes as our final model. For one, it had a categorical output. In our problem, we felt like it would be better to simulate being a reviewer and rating a movie rather than trying to predict the average rating of a movie. This primarily leads to easier to comprehend outputs; as people, we think of a movie rating as a six or a seven rather than a 6.2321 or a 7.3328. This makes our model feel more like a person, which we think will lend additional "character" to the results of our experiments.

Additionally, it was the most accurate model. Part of the

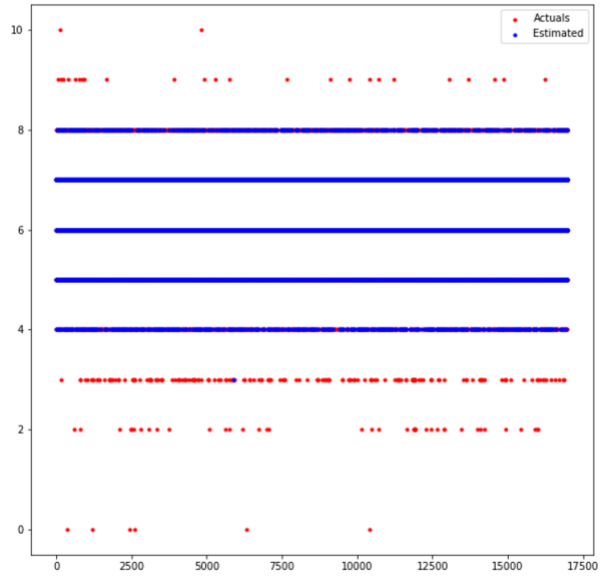
reason for this was simply that our data was less precise, which let us somewhat nullify the effect outliers had on our dataset to a degree. Guessing a movie was a 7 was really guessing that it was between 6.5 and 7.49. This means where our continuous models might have been off by around half a point, the Naïve Bayes model might have guessed it perfectly. The feed forward neural network did perform better than the recurrent network model, but surprisingly, not by a whole lot.

We think this is because recurrent neural network really only performs better when there is sequential data. Our data is nominally sequential, but in reality, that isn't really the case. The order the actors are billed in, at least in our dataset, seems almost entirely random. Each actor, regardless of their position, is equally important. This means that the primary benefit of the recurrent neural network was almost entirely nullified, which lead to having worse results than initially expected.

4. Experiment

The first movie we decided to test was "Conquest of the Planet of the Apes". This film was picked because it seemed like a fairly average movie by basically every possible description. It did alright at the box office. It had an average score close to the mean of our data. It had a cast of actors that, while not completely unknown, are/were not exactly blockbuster stars. The score of the movie was a six, and we were given back a six as a result! This was a promising sign. In our dataset, movies very heavily lean toward being typical; movie scores follow an extremely normal distribution. Almost all movies score some variation of a 5 or 6, with a healthy amount of movies also getting a 7. Beyond those parameters are the wild west; they are extremely isolated.

This sort of experiment was repeated with a few more movies; Iron Man, Fast and the Furious, several of the Transformers movies, among others. When the movie is scored a six or five, we almost always guess correctly. While still performing fairly well, our guesses aren't quite as efficient for guessing a seven; for example, our model gave American Psycho a six, while it should have been a seven. This was a common trend in both directions. When we missed a guess, it was almost always because we guessed too highly on bad movie, or too low on a great movie.



This is a graph of our data in the test set, overlaying what we estimated a movie to be versus what it actually was. Clearly, we've painted the map for our middling movies extremely well. If it's in the middle spectrum, we guess it right almost every time.

Clearly, our model is extremely reluctant to hand out a score outside of it's comfort zone. We only guessed a three for a single movie in the entire dataset, and literally never guessed a 10,9,2,1, or 0. Additionally, on these guesses it also wasn't usually particularly close. 'The Civil War' is rated a nine, and has prominent actors such as Morgan Freeman starring in it. Additionally, it's a documentary, a genre that typically recieves more favorable reviews. Based on a human point of view, it would have been easy to guess this would be a high scoring movie. Our model picked up none of that, and assigned it a score of six; not even a seven or eight!

Our model performed similarly bad at guessing which movies were complete stinkers. 'Britney Ever After' is a drama starring a bunch of no name actors that scored a three. Our Naïve Bayes model suggested that it thinks it would be scored a six! One time, our model guessed a three. It is the singular example of a three in our entire dataset, and is pretty much the only movie noteworthy enough to receive any sort of outlier score. This movie is the masterpiece known as "Super Shark".

Our model actually guessed it correctly; it was a three in the system, and a three in our model. With a tagline like "Bikinis, Bullets, and Big Bites", it clearly didn't start off on the right foot. For some reason, this movie was classified as Science Fiction. That helped our model a little bit, since there are a lot of movies that are particularly bad in this genre as it is weighed down by a

lot of pretty old movies that were mostly low budget scholck aimed at young boys. In particular though, the actors of this movie really weighed it down.

Two of it's billed actors have this movie as their only film credit. Our model doesn't seem to take this into account very much; movies filled with actors with only a couple movie credits seemed to be weighed very lightly by the model, but with only this horrendous movie as their credit, their presence certainly didn't help. One terrible offender here is an actor that is John Schnieder. He is best known for his role on the TV series "Dukes of Hazzard". It seems like he didn't get that many invitations to star in movies, but no matter how bad the movie looked, Mr. Schnieder wouldn't turn it down.

As a result, with fourteen credited parts, his average score is horrendous. Not the worst in the system (that is saved for another actor in this movie), but for having as many movie credits as he does, his average is awful. Most of his films score somewhere in the three to four range. Delving further into John Schnieder, our model seems to truly hate him. His best rated film is October Baby, which was scored a seven by critics. We rated it a five! While on occasion our model would rate a seven as a six, out of all the movies we tested, this was the only film to be underrated so bad while within the middle range of scores.

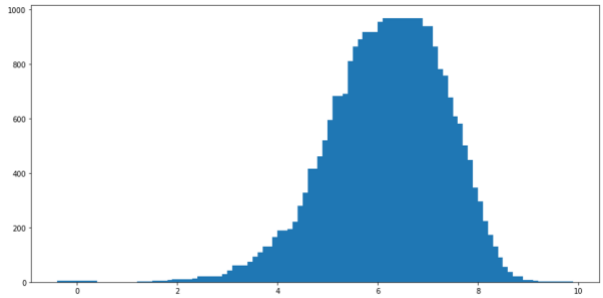
For the worst actor in the dataset with any reasonable amount of volume we managed to dig up, we have Sarah Lieving, John Schnieder's co-star in this movie. Out of all eight of her film credits, none of her films even managed to score above a three, with the vast majority being rated as a two. Most actors that star in such terrible movies either have most of their roles uncredited, or simply stop finding jobs. Not Sarah. Mrs. Lieving ignored the reviews, and managed to have a career in Hollywood that spanned over a decade; good for her! With the pairing of Sarah Lieving and John Schnieder, two actors that love to star in abysmal movies, the stars managed to align to assign "Super Shark" a three.

5. Conclusion

Our primary conclusion is that our model is a success; 86% accuracy seems like a great mark, considering the small amount of features we worked with. Additionally, the movies we failed to predict accurately also made sense. Almost all of them were outliers, or at the very least close to being outliers. Considering how few movies achieve a 10 or get rated a 1, it would've been extremely difficult to collect enough data on these points to differentiate them from more average movies while keeping our accuracy high.

The first conclusion that we came is that the genre and cast combination are actually fairly decent (but not outstanding) factors to predict how successful a movie will be. Without even taking into account things such as directors, budget, or writers, we attained a fairly high accuracy and reasonable precision while doing so. However, these features aren't perfect. Whatever factor that elevates an eight to a ten or downgrades a three to a one simply wasn't present in our features. If we had included a feature such as budget, it could have helped; according to our background research, most movies that score really, really low tend to be from first time directors and shoestring budgets. While there simply wasn't enough budget data in our dataset to justify adding it is a feature, if we had added it, we possibly could have better luck guessing on the extreme lower end of movies.

Whatever factor makes a movie a critical masterpiece, we have concluded it is something that is beyond the cast and the genre. Perhaps having acclaimed writers as a feature might have helped, but it is hard to say for sure. If it were as simple as using a machine learning algorithm to determine how to make a critical masterpiece, we're pretty sure it would have been done by now. We firmly believe that what makes a great movie an excellent movie simply isn't a factor that would be present in any dataset.



The above graph shows how heavily concentrated the scores regarding movies is towards the average; unfortunately, this makes somewhat difficult to draw a conclusion on how great our model is. When a movie is near the middle, we have an extremely high accuracy. When a movie deviates towards the ends of the spectrum, we basically have no chance of ever guessing it. We believe this shows the inherent limit in using genres and actors to predict the score of a movie, at least as the sole factors. Actors typically, no matter how good they may, still only usually average a score of a six or seven for a career. Actors that have horrific averages typically only have a few acting credits under their belt. As a result, the model weighs their presence less than perhaps it ideally should. Genres also weigh far to heavily towards the average, with the worst and best genres not even two

points apart on average. As stated earlier, while the cast and genre are decent predictors, our experiments shows that when used alone their guessing power is somewhat limited.

Another conclusion we drew from our testing is that often, there isn't a single actor that carries a movie. Even actors that are often picked to "carry" (actions stars like Dwayne Johnson come to mind here) a movie, often fail to do so on their own, at least as far as critical reception is concerned. Rather, great movies are often from a deep cast. It takes multiple good actors working together, rather than a single great actor and multiple mediocre ones. In several results, ensemble casts combined together to make a great movie, and switching one of them out might have had a detrimental effect on the final score, but not an enormous one.

6. Division of Labor

Dax Jones' primary contribution was constructing and testing the Naïve Bayes model, as well as configuring the format to input a guess into each model. Parker Hague's main task was to extract the data from our dataset, configure the embedding layer to work properly, and build/test the feed forward neural network. Steven Mikels' responsibility was to construct and test the recurrent neural network, as well as be the primary author of the written report. As a whole, we worked very collaboratively on the project. Everybody was able and willing to help other team members with the task they were currently working on, and the majority of our work was done within in-person meetings where we all added to the project at the same time.

7. References

Bioglio, Livio, and Ruggero G. Pensa. "Identification of Key Films and Personalities in the History of Cinema from a Western Perspective." *Applied Network Science*, Springer International Publishing, 30 Nov. 2018, https://appliednetsci.springeropen.com/articles/10.1007/s41109-018-0105-0.

Reich, John. "Introduction." *Exploring Movie Construction and Production*, Open SUNY Textbooks, 11 July 2017, https://milnepublishing.geneseo.edu/exploring-movie-construction-and-production/front-matter/introduction/.