**Making Money in Movies:**

**Determining the Profitability of Films**

Final Project Using Neural Networks for Regression (and Classification)

By Rick Stanton | CS5300 | Dr. Badri Adhikari

# Introduction

Movies are a massive industry generating billions of dollars of revenue worldwide per year, but they do not come without risk. Most films do not make a profit, in fact approximately 50% lose money [[stephenfollows.com](https://stephenfollows.com/hollywood-movies-make-a-profit/)]. According to Investopedia, in 2007 it cost about $100 million dollars to produce, distribute, and market a major studio movie. That is an expensive bet. What if the producers and investors in movies had a way to determine if their investment was worth the risk? In this paper, I attempt to answer this question by using a neural network to predict profitability based on a set of existing movie data.

# The Data

## Initial Phase – Finding a Data Set

There are many sets of data concerning movies that exist and are available, but not all are suited for this task. I had certain criteria that had to be met.

1. **Record Size**: There had to be enough records to perform the data analysis. Ideally it would contain more than 1000 records.
2. **Quantitative over Qualitative**: Many sets of data had the title and description of the movie, but I looked for data that was more number based and comparable.
3. **Representative**: I wanted a set of data that would try to represent a wide cross-section of movies and genres, of hits and flops. Many sets focused on blockbusters or critically acclaimed movies.
4. **Pre-release Information**: In order for this data to serve in a predictive fashion, I only focused on data-points that would be available before release. Reviews and review scores are information available after the movie has been released and would be excluded in analysis.
5. **Budget vs. Revenue Data**: Budget is intuitively the most important piece of data as it determines the size of the initial risk of investment. Revenue is key to be used for training.

After reviewing many sets of data, I decided to choose a set on kaggle.com [<https://www.kaggle.com/karrrimba/movie-metadatacsv/kernels>]. This was a set of 5043 movies, with information gleaned from ‘IMDB.com’. Encompassed was 28 separate features with a mixture of qualitative and quantitative data, including budget and revenue data. It also looked representative with 46.3% of the movies showing a loss in revenue.

## Second Phase – Cleaning the Data

After seeing the initial in-class demo of using neural networks, I decided to reduce the number of features to begin training of the model. The first task was to eliminate unnecessary columns. Here are a list of columns that I excluded:

* **Color vs. Black/White**: The majority of the movies were modern so this was almost always color
* **Number of Critic Reviews**: Post-release data
* **All Facebook Related Data**: Director, three actors, and movie facebook likes
* **IMDB User Data**: Number of votes, ratings, etc.
* **Links**: link to imdb page

### Removing Bad Records

Now there were 15 columns of data. Less, but still more than desired. Now the second task was cleaning the data. I noticed that there were duplicate records, so they had to be identified and removed or they would overly influence model training. There were also records with data missing, so I choose to remove those as well. This was a time consuming process.

### Normalizing the Data

The next thing I noticed was that the gross revenue numbers were wildly different and this was due to the inclusion of international movies using different currencies in the same column. This made the data far less reliable, so I elected to focus on one country with the most entries, the USA.

Also, some of the very old movies did not adhere to the modern content rating system (e.g. PG-13, R, etc.), so I elected to remove very old movies that didn’t use the modern system for better comparison.

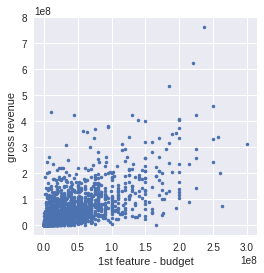
After removing the extraneous records, there were 2982 records remaining. This was a loss of over 2000 records, but there were still enough for thorough training.

## Second Phase – The Initial Data Set

The training set would be 2000 records, with 980 reserved for validation. Being unfamiliar with neural networks, I decided to start with a simplified set of data using only 3 features.

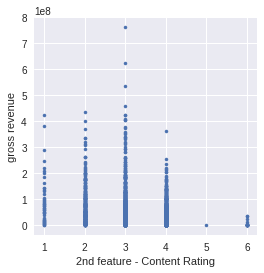
Budget

I hoped to see a high correlation with revenue from this feature, and while there was some correlation, with higher budgets relating to higher revenues, it wasn’t very clear. See below for an example:



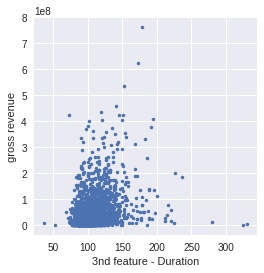
### Content Rating

For example: G, PG-13, R, etc. This feature affects the potential size of the audience so therefore could have an impact on revenue. This feature is discrete so the graph only has points on the integer values that represent the respective content rating. I arranged them from least restrictive (G) to most (NC-17). Note that there appears to be a somewhat normal curve with PG-13 (value – 3) at the peak with higher associated gross revenues.



### Duration

This feature indicates the length of the movie in minutes. I didn’t expect a strong correlation here, but there is the idea of a movie being too long that it affects its reception. This graph indicates that there is a somewhat of a peak at the 1hr 40min mark, along with many noise points, especially at higher durations. Longer movies appear to be more random, and thus more risky.

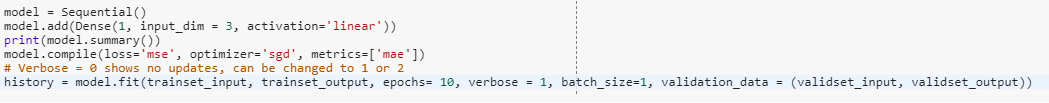


# Building the Model

Ideally, the model would be able to predict, through regression, the amount of money the movie would make (gross revenue).

## Initial Model

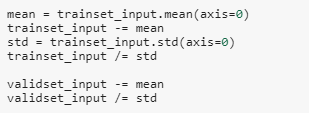
At first, I tried to implement a simple model using linear activation with one neuron. The model would be compiled with the Mean Squared Error used for the loss function.



At first all of my results yielded ‘Nan’ (Not a number) values. I could only begin to get meaningful predictions when I began using normalization.

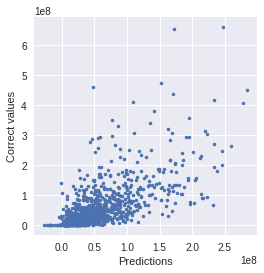
### Normalization

Before the construction of the model, I normalized the training set (see below). I tried omitting this step before every attempt at a new model and almost always saw the ‘Nan’ behavior unless I used a sigmoid activation, but then the results suffered. So this step is almost always included.



### Results of the Predictions

Now I began to see actual predictions. My first comparison of the predictions to the validation set yielded a Mean Absolute Error (MAE) of *34,325,374.96*. This was mostly consistent with an average of each prediction being off by over $34 million. That is quite high!



### Verifying the Predictions

I verified the predictions produced by the keras model by getting the weights and bias and multiplying that with some of the values. From my calculations, the keras model predicted correctly.

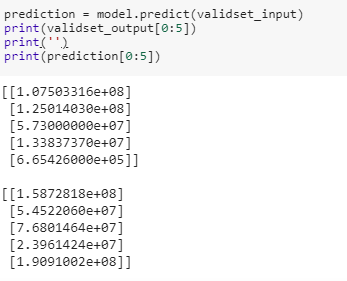


Figure 1: Sample of results produced by Keras Model

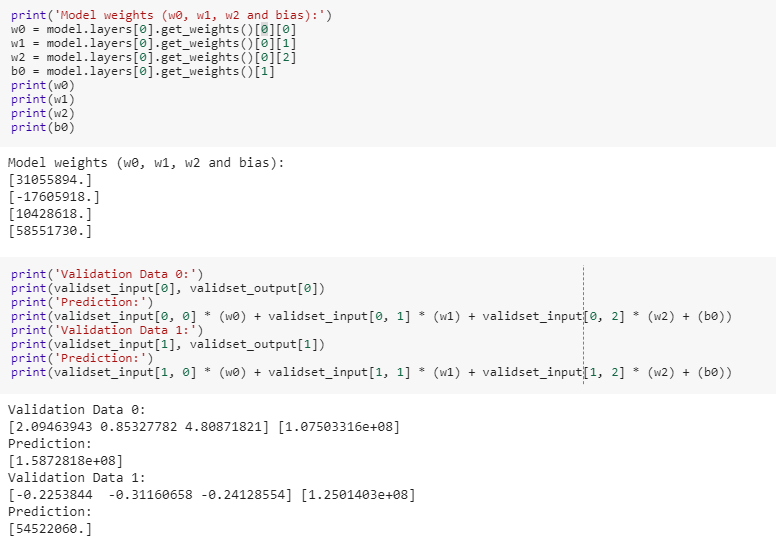


Figure 2: Small sample of results produced manually

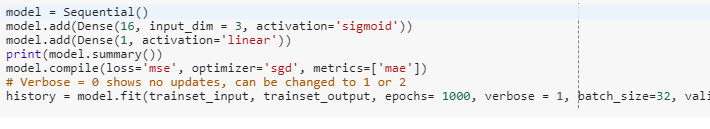
The two results that I manually checked compared successfully to the keras model.

## The Model Evolves

The results produced were not satisfactory, but at least I had a starting point. First, I explored changing the parameters in the initial model, thinking there may be a small change that could produce better results. I tried changing the loss function, optimizer, and metrics parameters, but they had either no improvement or drastically worst performance. I then tried adding epochs and changing the batch size of the training portion. This too had very little effect. The model would reach a stable loss in just a dozen or so epochs typically. The batch size also had very little improvement, with lower numbers performing slightly better while increasing run time.

### Building a More Complex Model

The next attempt to improve was to add more complexity to the model in hopes of getting better predictions. I tried adding a sigmoid activation step before the final linear activation step, but this produced even worse results than the original.



I also tried changing the number of neurons and epochs, etc. with little success. Every time the result would be worse, typically greater than $40 million dollars; although with a lot of trial and error, I did see some MAEs around $35 million. Still worse than the original model. Perhaps the model wasn’t the problem, maybe the data was insufficient.

### Adjusting the Data Set

One attempt to improve the data was “splitting” one of the features. The second feature, content rating was initially given a numerical value and perhaps that was throwing things off. I decided to separate each content rating into a binary value. This did nothing to improve the results. This situation was handled well by the neural network, so I reverted this data into one column again.

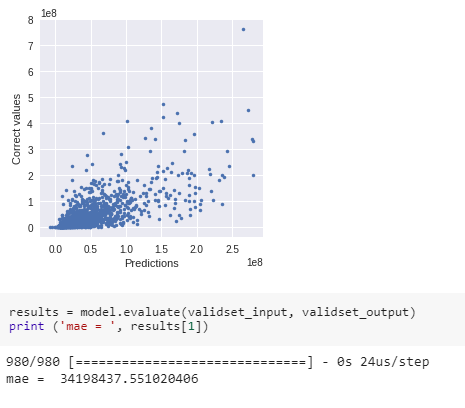


Figure 3: Results of 'split' content rating feature

Perhaps there just wasn’t enough data to make this prediction. I began to add additional columns in the hopes of improving predictions. I focused on adding genre data. As each movie could contain multiple genres, I decided to make each genre (19 total) into its own column with an attribute value of ‘1’ if the movie had elements of that genre associated with it, and a ‘0’ if not. Some genres seem to have good comparisons to gross revenue.

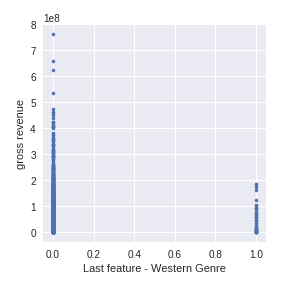
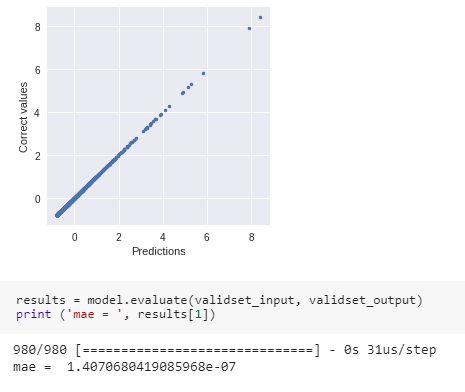


Figure 4: Westerns are not blockbusters

In spite of adding all these columns, there was only a marginal improvement in predictions with a new MAE of approximately $32 million. Perhaps there were too many noise points. I removed all movies before 1999. Movie watching patterns have changed and perhaps the older data was polluting the data set. But even after removing the older movies, the predictions held at $32 million. My assumption is that even though the budget and revenue were smaller, perhaps there was still a similar correlation, so removing those movies didn’t change the predictions.

### Getting Desperate: Verifying Successful Training

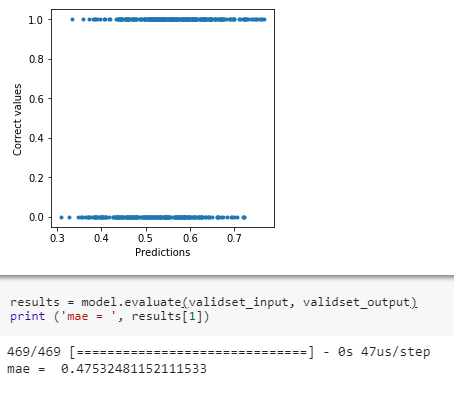
At this point, I wanted to prove that the model training was working, so I decided to include the training classifications as part of the training features (i.e. using outputs as inputs). This should make the predictions very close, which is what occurred with the resulting MAE very close to zero.



So, this means the training does work. So now what? My last attempt to try and get more accurate predictions was to move to a classification model.

### Classifying Profitability

The first part of this adjustment was to alter the data set to include a binary classification for profitability. If the gross revenue was greater than the budget, then the classification was marked with a ‘1’, or else ‘0’. My approach was similar to the regression model, but I used the sigmoid activation. My results were very poor with numerous variations of models. A typical result would produce an MAE of approximately .48, which is close to guessing randomly.



# Conclusion

I was not successful with creating a good model for predicting movie profitability. One reason could be that given the data, good predictions are not possible. Intuitively this feels incorrect as some of the graphs produced indicating feature correlation with revenue showed at least some correlation, albeit loose. Perhaps there was not enough information. I could have coded keywords or actors, but that would have been a very large amount of work and given the size of the data set, it may have not had enough reoccurrence to indicate a pattern.

The other reason, and most likely, for failure is that I just don’t yet understand how to use neural networks effectively. I did learn a lot using the keras models and see a lot of value in it, but it may be that this project was too ambitious for my level of understanding. It’s too bad, I could have made a lot of money in the movie industry.