## Abstract

A new NBC affiliate wants to make movies to show on their station. We used Netflix ratings, genres, and old movie scripts to make a recommendation machine for new movie scripts. The prediction model tested a gradient booster, a random forest, and linear regression. The model also tested an optimal number of features. Using root mean squared error as a metric, the linear regression model did the best at predicting ratings from scripts.


## Introduction

XYZ is a new NBC-affiliated station. It is new to its market. It has the stable of NBC shows behind it, but it is looking for other ways to improve its ratings in its market. Management has decided that broadcasting movies may help. They feel that well-rated movies will improve ratings.

As an NBC affiliate, XYZ has access to some money. From this limited pot of money, they can create their own movies. Management thinks that producing movies will not only improve their ratings, but improve their standing with NBC.

XYZ has access to scripts for the movies, but the movies do not have ratings from customers. It wants to know which movies to produce, because the money from NBC is limited. That is, XYZ wants to **predict** how well a movie will rate based on the script so they can choose good movies for production and broadcasting.

## Data

In order to create a recommendation machine that will use scripts, we need scripts and recommendations. Scripts come from https://www.simplyscripts.com/movie-scripts.html. For movie recommendations, I used information from IMDB, stored at https://www.kaggle.com/datasets/ashirwadsangwan/imdb-dataset.

## Method

The IMDB information has five .tsv files on Kaggle, but I was only interested in three files for the script recommender. The first file contains title, type of media (e.g.: television, documentary, short) and year in addition to other information; the second contains the title and language of the film in addition to other information; and the third file contains the ratings of the entries. There were over 10 million titles, and 1.4 million ratings of films and television shows. The films and television shows are linked across files by a unique code.

The scripts were on various websites, and linked to from the simplyscripts.com website; they were scraped from the source sites. They came in three formats: HTML, pdf, and text (.txt). There are about 1,000 scripts linked on the site.

## Data Wrangling

The IMDB data was very clean; however, there was a lot of extraneous information. In order to find just the movie scripts that are linked on simplyscripts.com, I needed to delete other media (e.g., short, documentary), movies from other countries, and movies that aren't yet made. I linked the year and language files and deleted all the extraneous media, countries, and unmade

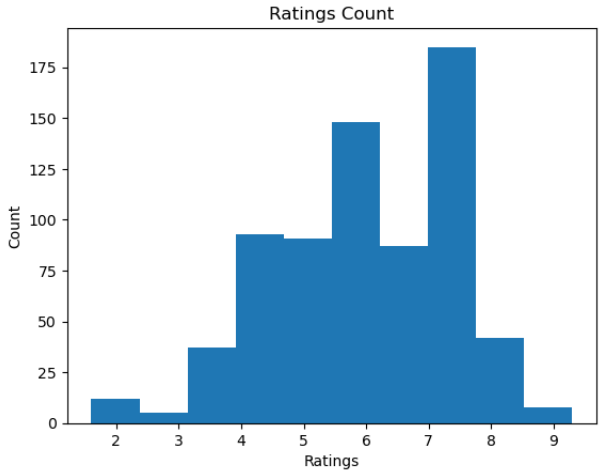
movies. I then linked the remaining movies to their IMDB ratings. This resulted in a dataframe that had the name of the movie, the genres, and the rating. I converted the genres into a bag-of-words, as most movies had multiple genres. I was left with 1100 movie ratings.

The scripts were not as clean. Some of the links were bad, and had to be avoided during scraping. In addition, the links that did exist required some manual work, as several of the downloaded files linked were merely notifications that the scripts were no longer posted. I manually deleted all the files that were not scripts. In addition, I deleted the file names of scripts that were unpublished. I read the scripts and movie names into a dataframe. Finally, I converted the scripts into a bag-of-words.

I merged the script dataframe and the IMDB dataframe into a final dataframe that contained all the words from the scripts and genres, and the ratings. All other information was dropped. This left just over 700 scripts and ratings to create a prediction machine.

## EDA

The ratings of the movies ran from 2.0 to 9.0 and clustered around 6.5:




I looked at the correlation between the words and the ratings. The correlation for individual words was not very high:

| | words | correlations |
|---|---|---|
| **6701** | genre_horror | -0.27 |
| **6703** | genre_musical | -0.21 |
| **6710** | genre_thriller | -0.16 |
| **6476** | watched | -0.14 |
| **6037** | thrill | -0.14 |

| | words | correlations |
|---|---|---|
| **878** | buildings | 0.13 |
| **6689** | genre_adventure | 0.14 |
| **6702** | genre_music | 0.15 |
| **6712** | genre_western | 0.17 |
| **6695** | genre_drama | 0.22 |

The bag-of-words had over 6000 words in it, which is too many features for the number of ratings we have, so I used SVD to convert them to 27 features.

Algorithms and ML

I tested three models for the prediction machine: random forest, gradient booster, and linear regression. I compared them to a simple model that used the average rating as the predicted result for all scripts. In order to use a random forest and gradient booster, I converted the ratings into integers by multiplying them by 10, so they ranged from 20 to 90. I used root mean squared error (RMSE) as the accuracy metric.

I tuned hyperparameters in the random forest learner and the gradient booster learner. I found that the best hyperparameters for the random forest are:

| | |
|---|---|
| number of trees | 120 |
| max-depth | None |
| criterion | 'Gini' |

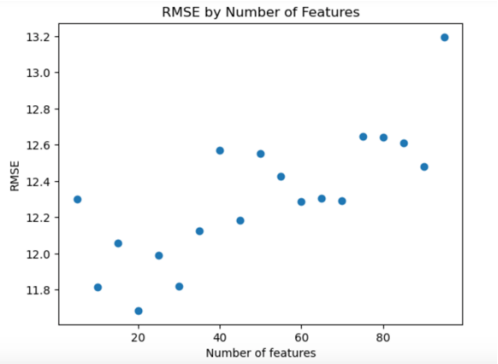
The best hyperparameters for the gradient booster are:

| | |
|---|---|
| learning rate | 0.05 |
| number of estimators | 250 |
| max. features | 10 |
| max. depth | 2 |

The RMSE results for the four models (including the simple model) are:

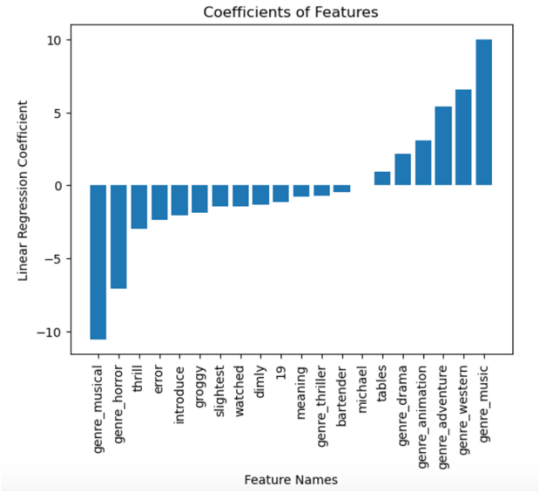
| | |
|---|---|
| simple model (average rating) | 13.923 |
| gradient booster | 15.761 |
| random forest | 15.156 |
| linear regression | 12.040 |

With these results, I reconsidered the SVD as a feature reducer. I then tried the Scikit Learn method of SelectKBest to find the ideal number of features for the linear regressor, using RMSE as the metric:




As can be seen, 20 features gave the best RMSE.

I also extracted the features that were chosen by SelectKBest, and their coefficients:




## Recommendations

A script can be fed through the machine to predict the likely rating, but we can make some predictions even before doing that. If we look at the coefficients, we can see that several genres have a strong effect on the ratings. Unfortunately, the strongest coefficients contradict each other: “genre_musical” has the strongest negative effect, and “genre_music” has the strongest positive effect, but both refer to musicals. However, it’s clear the station should not make a horror movie. The station may have some success with a western or an adventure movie. Finally, although animation has a weak positive correlation with ratings, the cost of making an animated movie makes that genre a poor option for the station.

## Future work

In the future, more scripts will make the script recommendation machine stronger; for example, the current list only goes through 2009. We could also drill down more into the “music” vs. “musical” genre to see what additional information is needed to separate out the two genres. Finally, we could see if including other ratings organizations such as Rotten Tomatoes or Amazon produced a more robust recommendation machine.