# Springboard Data Science Career Track

# Capstone Project Report:

# Predicting IMDB Rating of Upcoming Movies

**Submitted By:**

**Koshika Agrawal**

**April 15, 2019**

## Table of Contents

# 1. Introduction

## 1.1. Problem Statement

Today's movie watcher is spoilt for choice with increase in movie streaming platforms, DVD rental services and easy access to movie theatres. Knowing a movie's rating before watching the movie helps reduce the decision fatigue arising out of increased options. IMDb (Internet Movie database) is often a go-to source for knowing a movie's ratings, user reviews, plot, casting and other details before watching that movie. The ratings in IMDb are a function of the ratings given by the users.

But unfortunately, IMDb allows users to provide ratings and reviews only after movie release. So, if you want to select a movie for a first day show, you don't have the ratings to rely on.

## 1.2. Objective

Develop a model to predict IMDB rating of an upcoming movie

## 1.3. Client

- Movie watchers
    - Will have a rating even before a movie's release to decide whether to watch the movie or not
- Movie Theatres
    - The buyers representing the theatres can use the model to decide which movie to lease
    - They can further use the model to decide for how many weeks they should show the movie to have an optimum box-office collection
- Movie makers
    - The model will be using factors like cast, director, genre to predict revenue earned. So the movie makers can use this model to find a perfect recipe for a movie

# 2. Data Acquisition and Wrangling

Following 3 broad steps are involved in wrangling the data:

1. Data Acquisition: Collecting data from source
2. Filtering the dataframes
3. Merging dataframes

## 2.1. Data Acquisition

The dataset to be used in the project has been acquired from IMDB website from the link: https://www.imdb.com/interfaces/.
IMDB has 6 relevant datasets contained in separate gzipped, tab-separated-values (TSV) formatted files. These datasets have details of all the movies, tv episodes, documentaries etc for all countries of the world.

**Challenges with dataset**

Initial loading and inspection of datasets exposed following challenges in it for our study

1. Datasets have redundant columns that are not useful for the problem
   - ➢ These columns were excluded while loading data
2. The missing or null values are present as '\N'
   - ➢ '\N' entries replaced by NaN while loading data
3. Some columns do not have uniform datatype for all entries
   - ➢ dtype was set for non-uniform datatype columns

The aim was to resolve maximum issues during data load.
The shape of all datasets after loading is:

```
title_akas (3486933, 4)
title_basics (5642968, 7)
title_crew (5642968, 3)
title_principals (32301717, 3)
title_ratings (919215, 3)
name_basics (9139818, 5)
```

**Fig 2.1**

## 2.2. Filtering the dataframe

For this problem, we are just considering full-length feature films (movies) for the region United States of America released after 1980..

The reason for considering this subset of the dataset is mentioned below:

a. Full-length feature films:
   The goal is to predict the rating of upcoming movies, so taking tv episodes and other videos into consideration is redundant

b. Region: US:
   It would be irrelevant to consider movies from region A to predict ratings of movies from region B. So the model to be developed should use movies from a particular country to predict the ratings of upcoming movies from same country.

c. Movies released after 1980:
   To start with, we are considering only the movies which have been released after 1980. The reason for this is that the ratings of a movie are greatly expected to be influenced by the cast and crew involved. Most of the crew involved in movie making today, probably would not have any movies to their credit before 1980.

As can be seen above in fig. 2.1, number of entries in almost all the datasets are in millions. So before merging, the datasets are required to be filtered out on above parameters so as to keep the merge operation light on system resources.

## 2.3. Merging Dataframes

All the datasets to be merged have the column ‘tconst’ which is used as the merge key. The dataset name_basics is merged on ‘nconst’, which is present in other datasets too. The final dataset has 432774 rows and 18 columns.

The wrangled dataset is saved to a csv file.

The code for data acquisition and wrangling can be accessed in this ipython notebook.

Apart from the initial wrangling as mentioned above, there will be more data wrangling involved throughout the process as required..

# 3. Data Exploration

To begin our data exploration, we look at the trend of ratings:

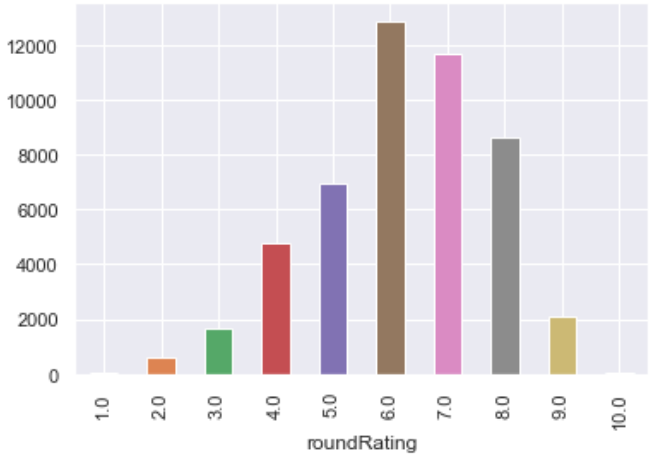


**Fig 3.1**

From Fig 3.1, we can make following observations:

1. The average rating of the movies seems to be 6.
2. The movie ratings are almost normally distributed

Let us take a look at some of the yearly trends:

a. Number of movies released each year
b. Average runtime of movies
c. Average rating of movies
d. Number of votes per movie

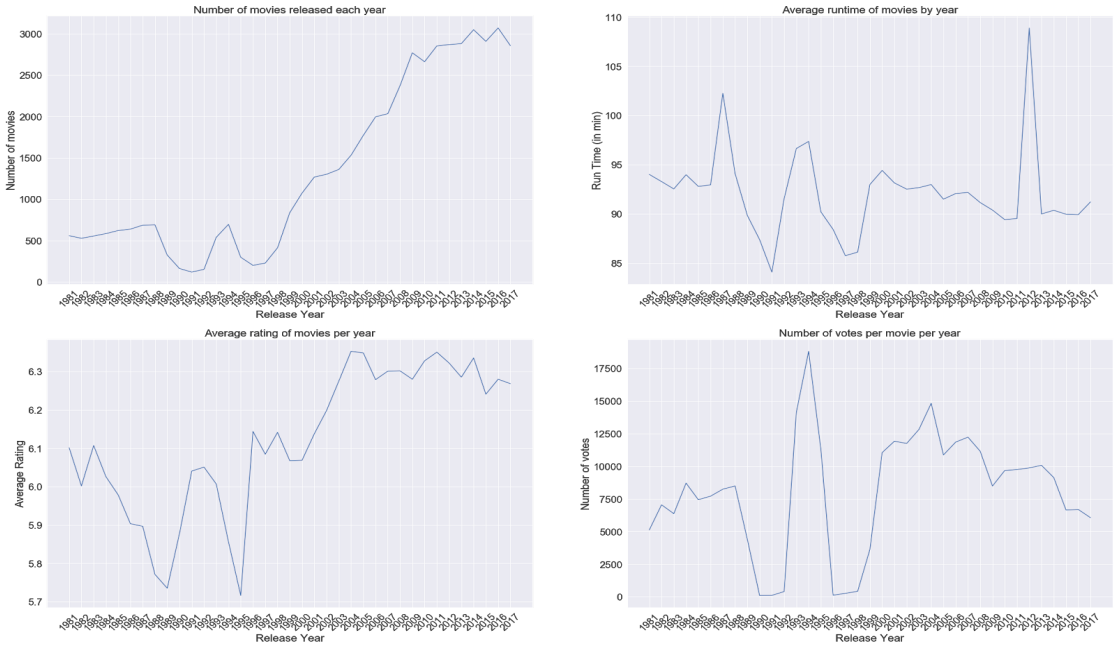


**Fig 3.2**

The above plots reveal inconsistent trends before 1999 on multiple features. So, in order to improve our data quality, we are considering the movies released after 1999 for the purpose of this project.

The code for above visualizations can be found at this ipython notebook.

We will further look at some of the features of interest:

## 3.1. Runtime (Length) of the movie

The trend of movie runtime (Fig 3.2) reveals the maximum runtime of one of the movies as 51420 minutes or 857 hours. Most feature films are between 70 and 210 minutes long. (Source: https://en.wikipedia.org/wiki/Feature_film). So, we will consider only movies with runtime of 240 minutes or lesser than that.

Length of the movies is positively correlated with the ratings. The average runtime of movies is at 102 minutes. Movies with runtime of more than 175 minutes have higher ratings.

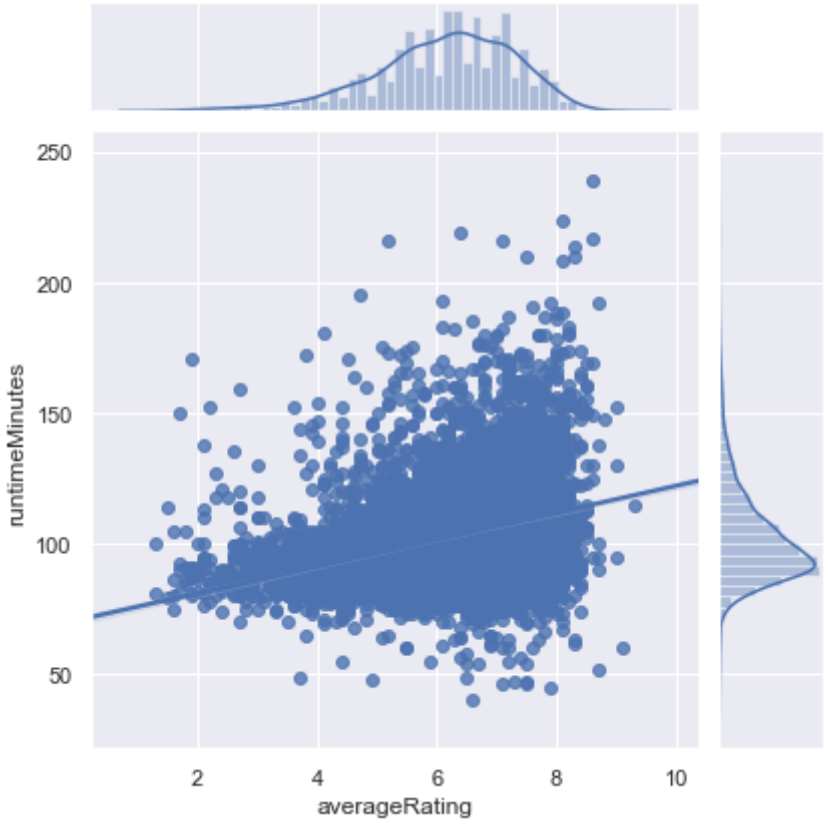


**Fig 3.3**

## 3.2. Number of votes

The dataset contains many titles with very few number of votes. As these observations lead to lower quality data, we will consider only the titles with number of votes more than 1000.

Number of votes could not be used to predict the ratings, as this will not be available to us during prediction. We have plotted it to get insights.

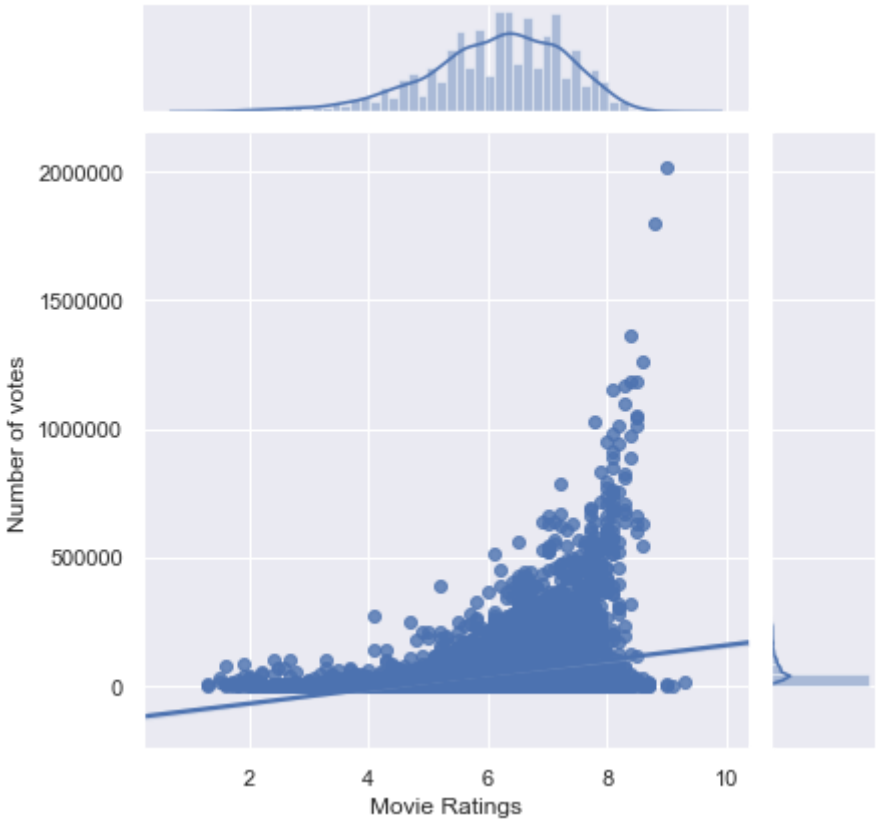


**Fig 3.4**

1. Number of votes for a movie are positively correlated to the rating.
2. Most of the cases of high vote count are for highly rated movies. From this we can conclude that people are more likely to vote for movies if they like it.

## 3.3 Genre of the movie

Genres with higher than average rating are - history, biography, war, documentary. So it appears that non-fictional movies get a higher rating.

Horror movies are rated much lower than average.

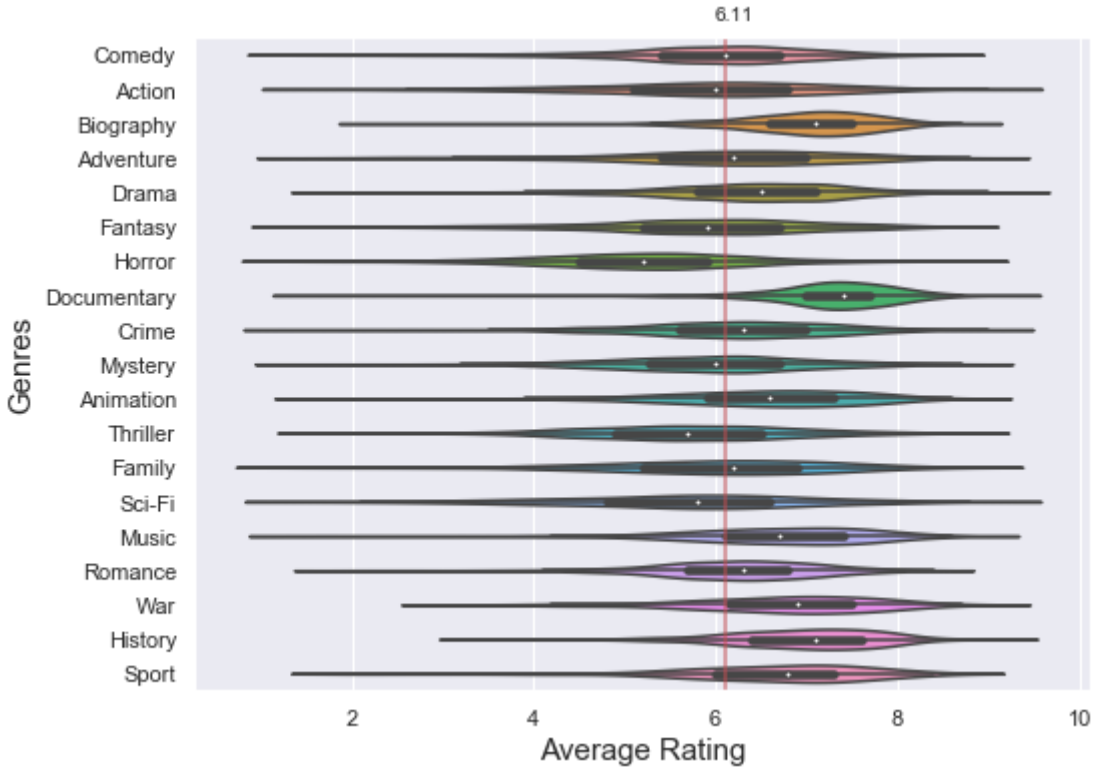


**Fig 3.5**

# 4. Feature Engineering

We have some categorical features that have to be converted to numerical variables.

## 4.1 Genres

This is a categorical variable with 22 different categories. We convert all these categories into features by adding a column for each of them with a boolean value for each row. We only consider the columns for genre entries which are applicable to 100 titles or more. So now we have 19 genre columns.

## 4.2 Cast and Crew

We refer to cast as those who have acted in the movie, which means it includes actor, actress and self. Crew includes those who were involved in making the movie - director, producer, editor etc.

We analysed the effect of cast/crew by parameters derived from cast/crew associated with the movies. These parameters are:

a) Number of each category of cast/crew in a movie. Category refers to actor, actress, director and so on.
b) Total count of cast, which means total number of actors, actress and self.

c) Total count of crew
d) Ratio of number of actors to number of actress in a movie

The relation between these parameter can be seen in the heatmap in Fig 4.1.

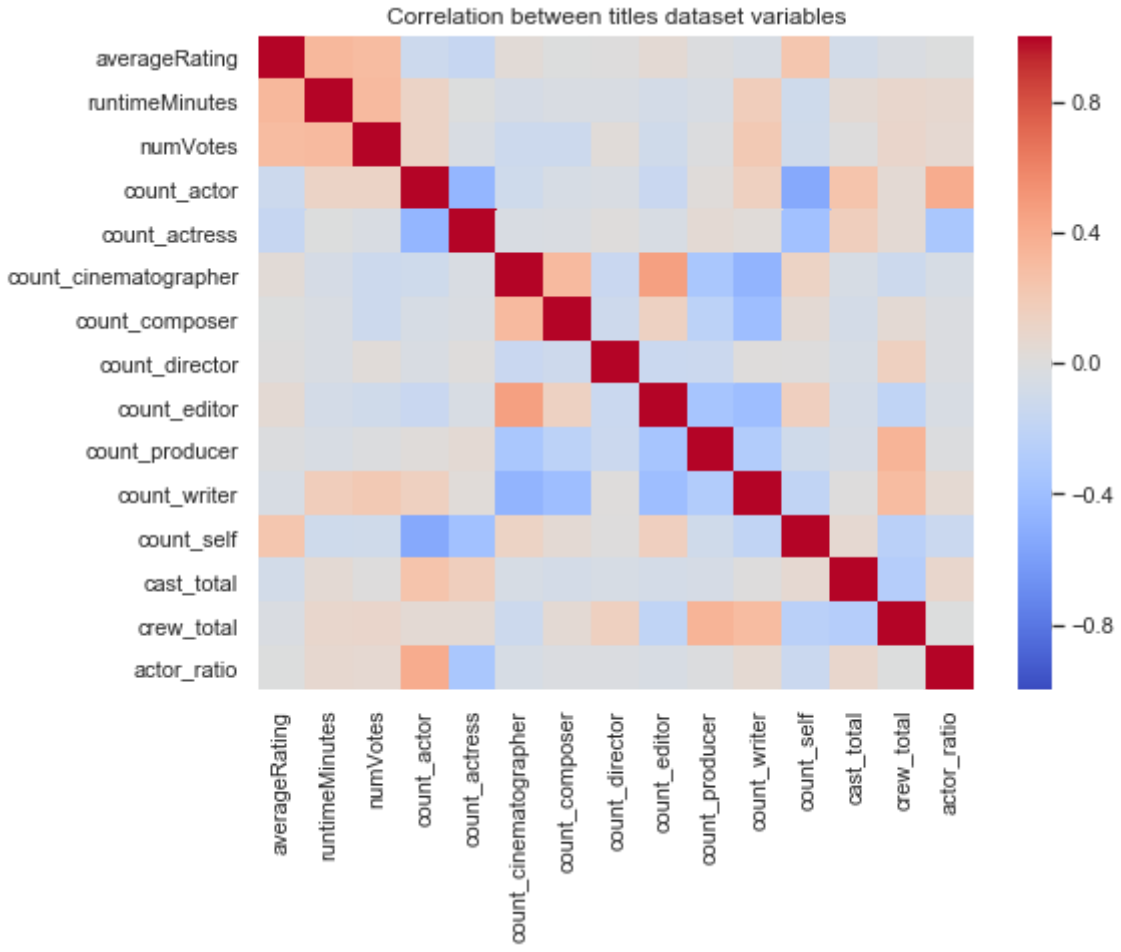


**Fig 4.1**

From fig 4.1, we can conclude that:

1. The heatmap shows positive correlation between ratings and
    1. runtimeMinutes (length of the movie)
    2. Count of self
2. The heatmap shows negative correlation between ratings and
    1. Count of actors
    2. Count of actress
    3. Count of total cast

## 4.3 Cast/Crew Ratings

We work out the ratings of cast/crew on the basis of the ratings of movies they have worked on. We are only considering cast/crew that has worked in more than one movie. Any cast/crew associated with a single movie will be assigned a NaN rating to avoid data leakage.

For each movie's category, the ratings of cast/crew in that category were aggregated to get the rating for that category. The aggregation function used is mean.

Heatmap of cast/crew mean ratings:

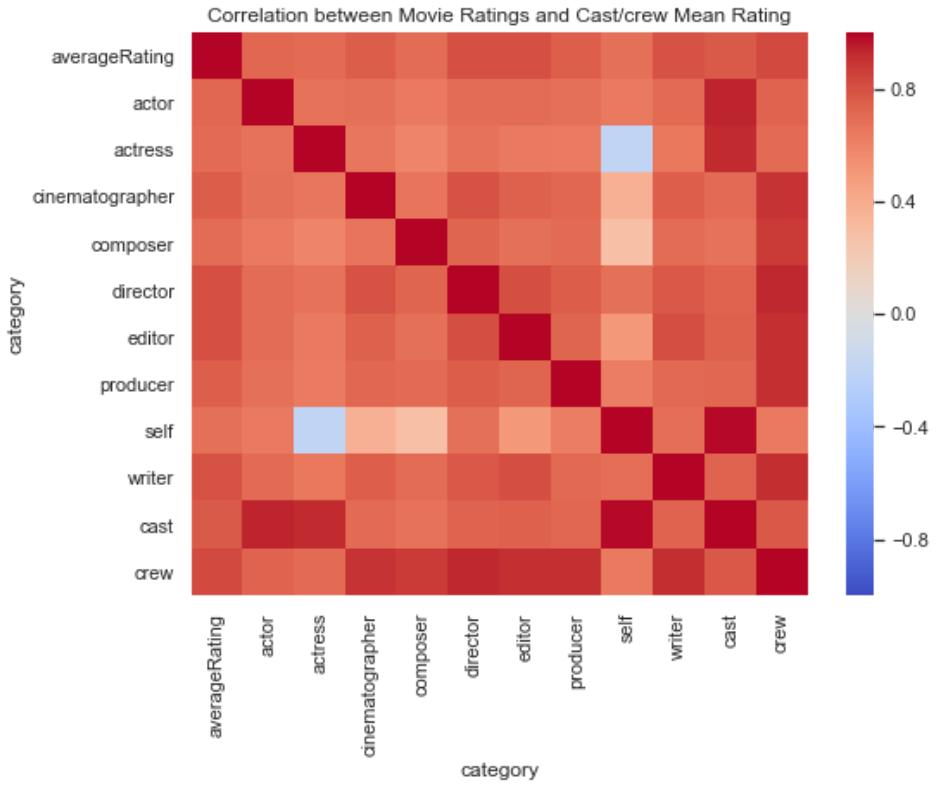


**Fig 4.2**

All the variables show a positive correlation with the ratings, probably due to the fact that these variables have been calculated from the ratings itself.

## 4.4 Most frequent cast/crew

There might be some cast/crew that affects the ratings of the movies. It is not possible to incorporate all the crew members as features as their are around 83000 unique cast/crew.

In order to incorporate the cast/crew, we are considering the 10 most frequent cast/crew each in the top-rated (rating 8 and above) and bottom-rated (rating 4 and below) movies. These 20 (most frequent-10 from top rated movies, 10 from worst rated movies) cast/crew are added to the dataset as columns with boolean values.

The code for the above plots can be viewed on this ipython notebook.

# 5. Modeling

The code for the modeling procedure can be found at this ipython notebook.

## 5.1 Target and Feature Variables

The target variable ranges from 1 (lowest rating) to 10 (highest rating). As it is a continuous variable, we will be using regression algorithms.

We have 63 feature variables that we will be using to train our model.

## 5.2 Train Test Split

Using the entire dataset to train our model might lead to data leakage and thus affect the performance of the trained model on unseen data. To address this issue, we split our dataset into train and test datasets wherein we train our model on the train dataset, and test its performance on the test dataset. For our project, we split our dataset 70% in train dataset and 30% in test dataset.

## 5.3 Process and Methodology

To find the best model for our purpose, we train our data on different algorithms, compare them and then select the one that gives the best performance on our evaluation criteria.

We will also use stacking learning technique to train our model. Stacking is a multi level learning technique in which the base level models are trained on the training set, then the next level model is trained on the outputs of the base level model as features.

We will use the following algorithms:

1. Linear Regression
2. Regularization Model - Ridge
3. Regularization Model - Lasso
4. Ensemble Model - Random Forest
5. Ensemble Model - Gradient Boost
6. Ensemble Model - Ada Boost

7. Stack - Linear Regression
8. Stack - Random Forest
9. Stack - Gradient Boost

## 5.4 Performance Evaluation

To compare the performance of different algorithms, we are using RMSE (Root Mean Square Error) of the prediction and time taken to fit/predict the model, and select the best. The model with lowest RMSE and time taken is desirable.

## 5.5 Model Comparison

We can compare the RMSE for train and test datasets for different algorithms in Table 5.1.

| Modelling Algo | Train RMSE | Test RMSE | Hyperparameters | Training+Test Time(sec) |
|---|---|---|---|---|
| Linear Regression | 0.889720 | 0.903349 | | 0.214 |
| Ridge Regression | 0.890131 | 0.902272 | {'alpha': 2.75} | 5.414 |
| Lasso Regression | 0.913179 | 0.919122 | {'alpha': 0.01} | 59.968 |
| Random Forest | 0.300850 | 0.630694 | {'n_estimators': 5000, 'min_samples_split': 10... | 662.8 |
| Gradient Boost | 0.324200 | 0.624689 | {'n_estimators': 500, 'min_samples_split': 2, ... | 89.796 |
| Ada Boost | 0.691293 | 0.694207 | {'n_estimators': 300, 'learning_rate': 0.05} | 53.94 |
| Stacking LR | 0.192679 | 0.650138 | | 872.194 |
| Stacking RF | 0.111337 | 0.643452 | {'n_estimators': 5000, 'min_samples_split': 5,... | 1012.3 |
| Stacking GB | 0.089379 | 0.647480 | {'n_estimators': 250, 'min_samples_split': 10,... | 2082.43 |

**Table 5.1**

Stacking Random Forest has lowest RMSE for train dataset. But RMSE for test dataset is very high, which indicates overfitting.

# 6. Summary and Conclusion

Below is the graphical representation of training and test RMSE for all algorithm used. The best performing algorithms are also overfitted, but that does not concern us much, as despite overfitting, we are getting low RMSE for test dataset.

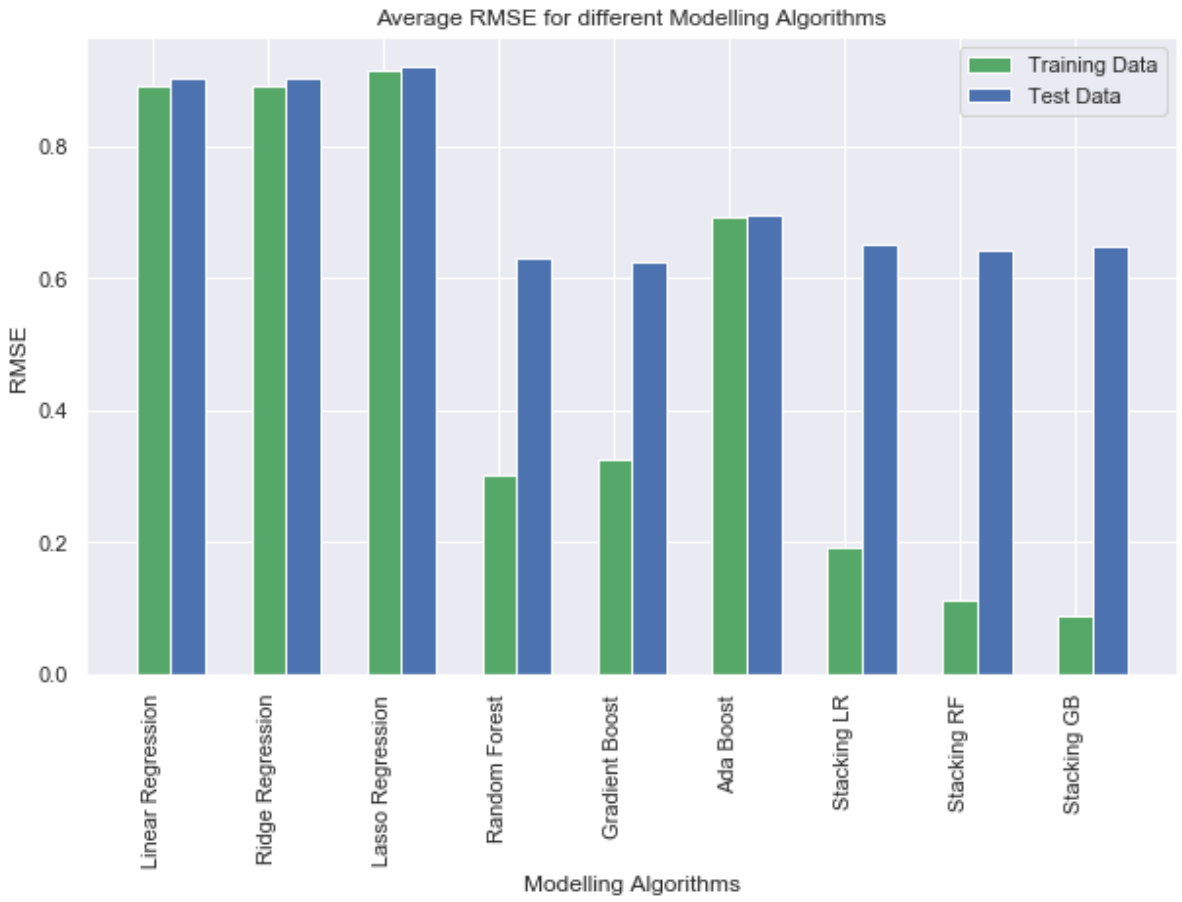


**Fig 6.1**

Random Forest and Gradient Boost show the best performance on the test dataset. In order to decide one of these, we look at the time taken in fitting the model. In all the algorithms, the prediction time was very small in comparison to the fit time.

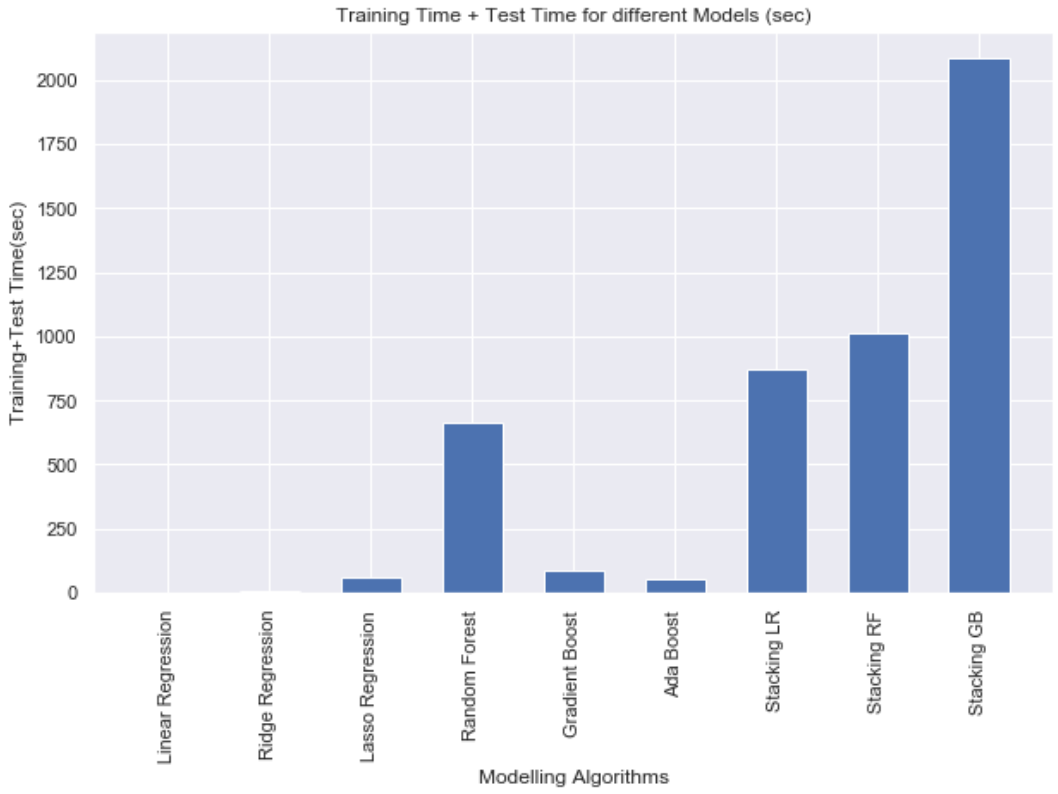


**Fig 6.2**

As seen above, Gradient Boost is taking much lesser time as compared to Random Forest. So we choose Gradient Boost as our final algorithm. Gradient Boost gives an RMSE of 0.625 on our test data.

Let us take a look at the plot of actual target variable and error.

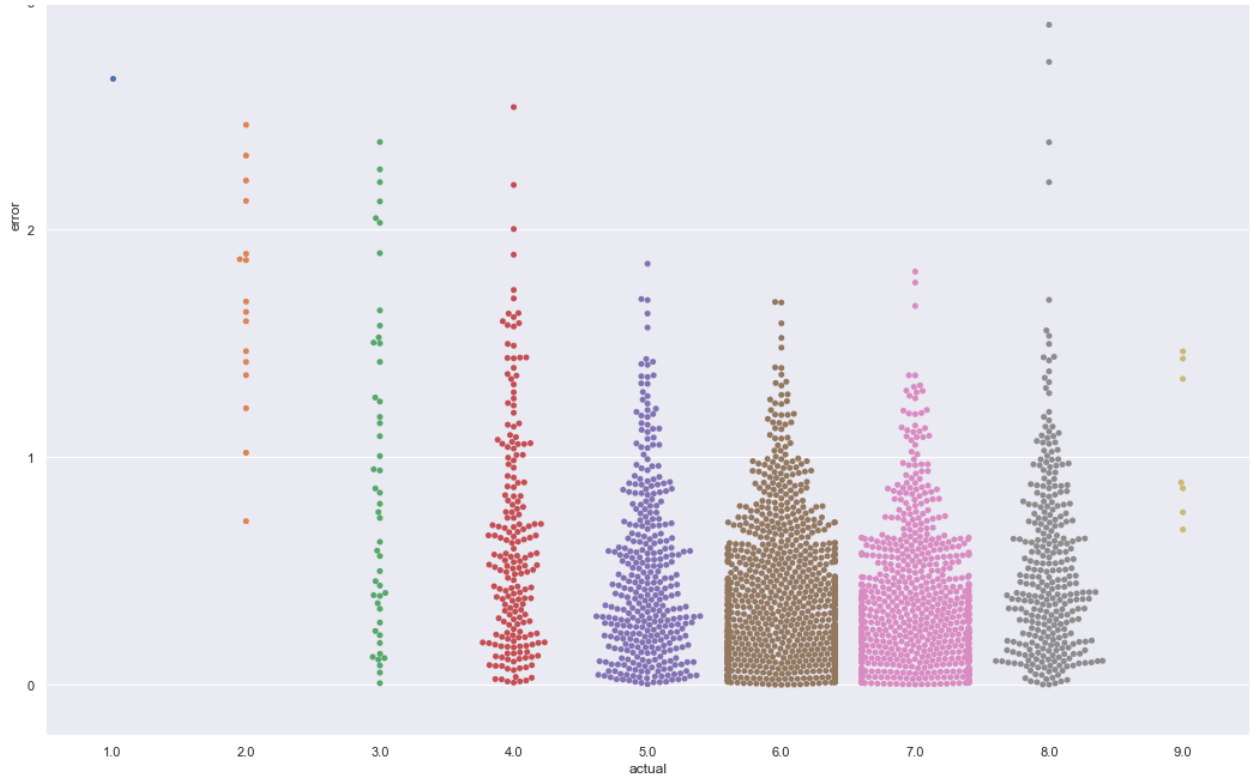


**Fig 6.3**

We notice that error is more for ratings for which we don't have enough observations. For rest of the cases, we get predictions close to the actual values.

# 7. Scope of further study

1. We have not considered the box office data in our project. Our model can be easily extended to predict box office collections. Also, we can use the box office collections as a feature to predict ratings.
2. While analysing the yearly trends, we saw a drastic change in all the trends after the year 1999. Though this might be a result of bad data but the data before year 1999 can be further explored in a separate study to explore interesting insights.