Stat 706 Final Project: Investigating Predictors of Movie Profitability

Nickhil Sethi

Introduction

Profitability and financing has become a contentious subject in the Film industry in recent years. The highest grossing films are consistently remakes of one kind or another – e.g. Marvel, Star Wars, etc. On the other hand, we've seen a surge of relatively small-budget independent films that receive high critical praise – Moonlight, Birdman, etc. Acclaimed director G. Inarritu once described the issue as analogous to global inequality; "1% [of the films have] 99% of the budget (https://youtu.be/SQ7qKKQrSBY).

Films must, at least on average, make a profit for the studio producing them. The goal of this paper is to investigate predictors of profitability, defined as `revenue - budget` (note, we assume films use all of their budget – no more, no less). In particular, we consider the following variables.

- Release date. This allows us
- Budget (captures the constraint that movies must be generally profitable)
- Genre. Potential
- average rating (captures the "quality" of the film)

The dataset used here is Kaggle's "The Movies Dataset", located here https://www.kaggle.com/rounakbanik/the-movies-dataset. The dataset is an aggregate of data from GroupLens and TMDB.

Methods

The first step was significant some data transformation. The two relevant tables and columns from the Movies dataset are `movies_metadata.csv` and `ratings.csv`, which have the following schemas, respectively:

```
MOVIES_METADATA {
  categories: { int: str },
  revenue: float,
  budged: float,
  movieId: int
}

RATINGS {
  movieId: int,
  userId: int,
  rating: float,
}
```

Three transformations were performed in order to turn the above data into a useful format. First, the genre column of movies_metadata.csv is JSON, encoded as a list of pairs e.g. `{[genreId: genreName] ...}`. I obtained the set of unique genres named in the dataset, and added one boolean column for every genre found. Second, the `ratings` table contains movie ratings at the level of `(movieId, userId)` pairs; a groupby operation was performed to extract the average rating for each movie. Finally, a join was performed between the two tables using the join table `links.csv`.

Given the computationally expensive nature of the above transformations and the need to be able to "start fresh" i.e. recover from potential data corruption, I conducted the data transformation and resampling on AWS servers. See terraform code in this repository which sets up a postgresql database and a single server on AWS to store and manipulate this data.