Predicting Movie Box Office Revenues

# Part 1: Background

## Part 1a: Description/Why are we interested

## Part 1b: Literature Review

# Part 2: Data Gathering

## Part 2a: Data Sources

The primary data sources we used were three websites that contained ready-made data sets and two websites that hosted an API that allowed users to pull movie information. In addition to these five primary data sources, we also needed to supplement them with other data found online.

The first primary website was an account on Kaggle[[1]](#footnote-1), which provided data from the MovieLens database. It included 45,000 movies, along each movie’s revenue, budget, release data, language, production company and several accompanying data sets on movie keywords, credits, and ratings. Because this was the largest and most comprehensive data source we would find, we made this our base file and sought out more information to make it a more complete data set.

The second primary website was a second account on Kaggle[[2]](#footnote-2), which provided data on 5,000 movies using The Movie Database. This dataset included the movie’s cast, crew, budget, genre, keywords, production company, revenue, languages, runtime and tagline.

The third primary website was the-numbers.com[[3]](#footnote-3). This dataset include just the movie’s release data, revenue, and budget.

The first of the two movie APIs we used was The Open Movie Database (OMDB)[[4]](#footnote-4). Because it was likely that these APIs had more up-to-date data or potentially had more features, we wanted to pull the same movies we had in the large MovieLens dataset and eventually merge everything together. In order to pull data from this API, we used the one of the movie ID columns in the MovieLens data and fed ID’s for any movie released in 1995 and onwards. We took the output and stored that as another dataset. Because of the amount of data we needed to pull, we required the assistance of the API’s owner to give us both his direct server’s URL and a temporary increase in our daily pull limit.

The second of the two movie APIs we used was The Movie Database (TMDB)[[5]](#footnote-5). This is where the data from the second Kaggle account came from; however, because of the possibility that pulling data from an API would give us more up-to-date data, we went ahead and also pulled data from this API in order to eventually merge all the data sources together. Pulling data through TMDB was a bit more involved because we had to do a two-step procedure. The reason for this is because TMDB provides a full set of movie features only when you pull movies using TMDB’s own movie ID system. Only a portion of the movie’s features would be pulled if any other recognized movie ID was used. As a result, we first pulled information using the same movie ID we used to pull from OMDB. We then extracted TMDB’s own ID for those movies and performed a second pull of the movie using this new ID.

Lastly, because our base file of movies was the MovieLens database, it did not have good coverage of movies in 2017 or 2018. As a result, we generated a list of all movies that came out those two years by a simple Google search. We took the names and release dates of those movies and input them into our two movie API sources in order to pull as up-to-date information we could.

# Part 3: Data Wrangling

## Part 3a: Joining datasets together

With all the files we either collected from websites or generated ourselves, there were roughly a dozen files that we needed to join together to begin our process of creating one complete dataset. The code associated with joining these files is in the joinDataModule.py[[6]](#footnote-6). Excluding the dataset we created from the-numbers.com, all the datasets from all the sources could be joined relatively easily after it was determined which movie ID each dataset was using.

Joining the dataset from the-numbers.com was required a different method because that data did not come with an ID. It only had the movie name and release data as usable joining columns. The movie name in both this dataset and other more-consolidated set were made lower case. Then the two data sets were joined using movie name and the year of the release. We believed this was the most reasonable way to join these two datasets. However, there were instances when this joining method resulted in duplicates; for example, if a movie was released in limited release in the winter (and was captured as such in one dataset), but was released widely at the beginning of the next year (and was captured as such in the other dataset). Future work on this could be on the best way to join these datasets to avoid such errors.

## Part 3b: Parsing data

Once we joined all the datasets into a single joined dataset, we were able to see that several columns were JSON strings of lists of dictionaries. In order to make them into more usable columns, we wrote several functions to first convert those strings into actual lists of dictionaries. Then we either parsed the values into separate columns or we extracted the values and replaced the lists of dictionaries with simple lists of values. The code associated with parsing these files is in the parseColumnsModule.py[[7]](#footnote-7).

## Part 3c: Merging data

Because several data sources were used to create the consolidated dataset, there were roughly 100 columns of features, many of which were similar because our data sources all provided similar features. However, we simply could not just delete any similar sounding column. The whole benefit of using multiple sources is to try to ensure that some combination of these sources would yield a complete set of data for each movie. Meaning, one source may only have data on a particular movie’s revenue, budget, and cast, but another source may have that movie’s plot and crew. By having multiple sources, we hoped to have a more complete picture of each movie.

Properly merging the features was the longest part of the data wrangling phase of our project because we had to ensure we were doing it correctly and were using our best judgement. The code associated with merging these columns is in the mergeDataModule.py[[8]](#footnote-8).

For each similar set of features, we implemented a different algorithm to merge them into one feature:

1. Name: Pick the first title found among list of movie name columns.
2. Revenue: Pick the maximum revenue number found among list of movie revenue columns. We made the assumption that a largest number indicated that either, that number came from a more up-to-date data source, or because the largest number indicated that it was most likely a global revenue data point.
3. Release Date: Pick earliest release date among list of movie date columns. This was a choice that may have been decided differently if we had the benefit of hindsight. Initially the idea was the earliest release date may be more likely to indicate a U.S. release date, or be a more accurate date for any subsequent new feature created that relied on when a movie was released. However, from our analysis, it seems like there is an inconsistent reporting in release dates and more analysis should be done on this matter. When a movie is first released in a limited number of theaters, that date could potentially have been chosen rather than the wide-release date. Further, some movies may have actually opened in other countries first, which would also result in a less-than-accurate guess on an appropriate release date.
4. Movie Length: Pick the average movie length.
5. Budget: Pick the average movie budget. This was because most movie budgets are estimates and taking an average would have been better than simply picking the maximum or minimum number.
6. Genre: Create a list and get the union of all genres listed in all movie genre columns.
7. Production company: Create a list and get the union of all companies listed in all movie company columns.
8. Actors/Cast: Create a list and append any actors listed in all actor columns. Because actors are usually listed in order of importance in the movie, we wanted to preserve the ordering and could not use sets or unions.
9. Keywords:
10. Movie Collection:
11. Movie Overview:
12. Movie Tagline:
13. Director:
14. Writer:
15. Movie Rating:

## Part 3d: Cleaning Data

# Part 4: Data Preprocessing

## Part 4a: Generating usable features

# Part 5: Modeling

## Part 5a: Regressions

## Part 5b: Classifications

1. <https://www.kaggle.com/rounakbanik/the-movies-dataset> [↑](#footnote-ref-1)
2. <https://www.kaggle.com/tmdb/tmdb-movie-metadata/home> [↑](#footnote-ref-2)
3. <https://www.the-numbers.com/movie/budgets/> [↑](#footnote-ref-3)
4. <http://www.omdbapi.com/> [↑](#footnote-ref-4)
5. <https://www.themoviedb.org/> [↑](#footnote-ref-5)
6. <https://github.com/georgetown-analytics/Box-Office/blob/master/codes/Data_wrangling_code/joinDataModule.py> [↑](#footnote-ref-6)
7. <https://github.com/georgetown-analytics/Box-Office/blob/master/codes/Data_wrangling_code/parseColumnsModule.py> [↑](#footnote-ref-7)
8. <https://github.com/georgetown-analytics/Box-Office/blob/master/codes/Data_wrangling_code/mergeDataModule.py> [↑](#footnote-ref-8)