Saving Cinema: Designing Movie Theater Showing Schedules

Data Preparation Report

Subject Area: Film Industry

Jillayne Clarke, Jeewon Han, Jay Lawrence, Sam Oberly, and Lucas Rozendaal

EN.553.602: Research and Design in Applied Mathematics: Data Mining

November 2, 2023

Review of Business Question

Our data mining project is concerned with the problem of designing movie theater schedules amidst a turbulent industry landscape. Using a wealth of features from *The Movie Database* (TMDB), Box Office Mojo, and *The Numbers,* we intend to craft a predictive model to assist theaters in choosing which movies to showcase. Although this report in particular discusses our preparation for predicting domestic box office revenue, our project is ultimately poised to create revenue-maximizing scheduling solutions for theaters.

2.1 Collect Initial Data

The initial data collection phase evolved from using an API for a single source into combining the API with web scraping from two sources and manually-collected data from a third source. Initially, IMDb was thought to be the most complete and accessible source of movie data available. This assumption proved to be correct, but a prohibitively expensive paywall for using the API and an inability to scrape (both because IMDb prevents web scraping and because the relevant pages are hidden) forced us to reconsider the primary source of data. Further research led us to TMDB. TMDB is an open-source alternative to IMDb with free API keys and documentation robust enough for simple requests. As TMDB is open source, we found a convenient API wrapper Python module to pull data from the TMDB website. This wrapper made obtaining the details of a movie by ID or other characteristics quite simple. However, this module ran rather slowly and made pulling large numbers of movies unviable. As such, we opted to use TMDB's native API and run it iteratively on Python to acquire the unique IMDb IDs associated with each movie from 2000 through 2023.

As we began pulling movies in 2000, we noticed two key issues:

1. Data is pulled from the API in "pages" of a small number of movies.

2. Not all movies can be pulled in one run without API access being paused.

To combat these issues, the methodology for pulling IMDb IDs was altered to account for these restrictions. First, we noticed that the total number of "pages" of movies in a given year was present in each API call, as was the number of movies in the database in that year. A script was created to store these numbers. We realized that after 500 "pages," the API would not let us pull anymore, so we conducted some analysis and found that each year in our dataset had under 1000 pages. An API call requires specifying how the JSON file that is returned should be ordered. We hypothesized that ordering by ascending release date, pulling from all 500 pages, then ordering by descending release date and pulling from all 500 pages would give us all the IDs with some duplicates, which we would drop.

The final implementation of the TMDB scraper involved taking the number of pages, dividing it by 2, taking the ceiling of that result, and then pulling that many pages of both ascending and descending IDs by release date (to minimize overlap). We then compared the number of IDs obtained to the number expected and found that we got more than 99% of the movies in each year. This list of unique IDs was stored in a CSV to allow for further analysis into which subset of movies we would use.

Once we had a list of movie IDs, we called TMDB's API several more times to create a dataframe of relevant information that TMDB collects about each movie (e.g. the revenue, budget, genre, title, etc., discussed in greater detail in our preliminary report). While doing these calls, we implemented a few initial filtering steps, choosing to drop any (1) adult films, (2) non-English-language films, (3) films without theatrical releases, and (4) films without IMDB IDs. The justification for the first three filters was to both narrow our scope and eliminate as many obviously irrelevant movies as possible. The fourth filter was a necessary step because we later

supplemented our TMDB dataset with information from Box Office Mojo, which stores data using IMDb IDs.

Our preliminary analysis into the dataset led to concerns about the usability of revenue data from TMDB, as outlined in Sections 2.4 and 3.2. After a collaborative search effort, we identified a site called Box Office Mojo, an affiliate of IMDb, as an additional data source. Fortunately, Box Office Mojo did not restrict scraping directly from their site. We built a custom scraper in the Beautiful Soup Python package that searched for a movie by IMDb ID and then took all the available summary information present on the primary screen of Box Office Mojo for a given movie, such as revenue and budget data. We scraped all the details for each movie in the dataset of IMDb IDs pulled from TMDB. This scrape was largely successful!

A key feature that we desired was a metric for measuring the star power of the actors/actresses in a movie. A website called *The Numbers* maintains a list of actors and actresses with their relative popularity in a given year. We attempted to scrape from this site, but found that they, unlike Box Office Mojo, had restricted scraping. Due to the very finite nature of the dataset, we decided to manually copy and paste each list of actors and actresses and their associated scores into .txt files so we could store them and later collect them into dataframes.

The TMDB, Box Office Mojo, and *The Numbers* datasets made up our initial data collection and each presented unique challenges to obtaining the data that we collectively overcame through a variety of technical and brute force solutions.

2.4 Data Quality & 3.2 Clean Data

Before we made the decision to add Box Office Mojo data to our model, we contemplated using only the data from TMDB and *The Numbers.* However, because TMDB is crowd-sourced, we had questions about the accuracy of its data. We were particularly concerned

with the possibility of incorrect inputs in the financial data (i.e. the budgets and revenues). Potential errors in financial data seemed particularly likely due to of the possibility of typos and rounding errors. Additionally, because TMDB does not automatically update its financial data, there is a high risk that TMDB box office data is slightly outdated. There is a lag between when a movie is input into the database and the end of its theatrical run, and users likely do not update a movie's final box office revenue.

To assess the initial state of the TMDB data, we extracted a random sample of 50 movies from our dataset of TMDB movies from 2013-2023, the span for which we justify in Section 3.1, and checked their information against data on IMDb. For the data in the sample, we found that approximately 30 had the correct worldwide revenue, 7 had worldwide revenue information that was almost correct but slightly below the final total on IMDb, and 14 had extremely low revenues listed or no revenues listed. These results suggest that TMDB data is not necessarily unusably bad, but that the concerns about the accuracy of the financial data have some validity. (These concerns are addressed later in Section 3.4). As such, we chose to supplement our dataset with information from Box Office Mojo. Box Office Mojo gets its data from IMDb, and so is likely to be more reliable than data from TMDB.

After we had verified the usability of our financial data, we needed to verify that our dataset contained as many of the relevant movies within our timeframe as possible. In other words, we needed to make sure that we had not (1) failed to pull any relevant movies that we should have in our initial API calls or (2) dropped any relevant movies that we should not have in the subsequent filtering steps or merges. Assessing this question is somewhat difficult, because there isn't a complete list of movies with the same filtering queries that we used create our dataset that we could compare our results to (i.e. there is no list of all the movies released in

theaters between 2013-2023 that are not adult films, are in English, had theatrical releases, have IMDb ids, have TMDB ids, etc.—otherwise, we would have used it!). Instead, we obtained a list of the top 100 movies by domestic revenue from 2013-2023 on Box Office Mojo, and manually checked it against our dataset, verifying that the movies that were in that list but were not in our dataset had been filtered out intentionally (e.g. because they were in another language).

This data quality check allowed us to identify three flaws with our dataset. First, during one of our initial API calls on TMDB, our code timed out part-way through the call, causing slightly over 1000 IDs from 2022 to not get pulled properly, a problem we had anticipated when retrieving IDs but not when calling movie details. Correcting this problem simply required calling the missing IDs on the TMDB API, running them through all our processing steps, and adding them back to the dataset. After all of the processing steps were performed on the 1000 IDs, we ended up with about 30 movies that had revenue and met all our relevant criteria, which was still a meaningful addition to the dataset.

Secondly, we identified a few other films from the top 100 movies list that were missing from our dataset that should have been there. The following 12 films were identified to be missing from the dataset for reasons independent of the API call timeout issue: *Zoolander 2, Monsters University, Minions, Love and Monsters, Arrival, The Shallows, The Fifth Wave, Sully, Suicide Squad, Storks, Rogue One: A Star Wars Story*, and *Pinocchio*. Given that these films have little in common and came out on dates spanning the entire 2013-2023 range, it seems probable that there are individual issues with each one, likely to do with how they were tagged on TMDB. In any case, the 12 missing films represent an average of 1.2 missing films for each of the 100 films per year that we checked: somewhat problematic, but not a crippling flaw in the

data. It would be straightforward to add back these missing films individually, though we have not yet done so.

Finally, we identified a problematic subset of films in our dataset that were not in the list of top movies by U.S. domestic revenue by year from Box Office Mojo. These films in our dataset were foreign films that were not released domestically. Although we filtered by English language and theatrical release, we did not correctly filter by U.S. release in our initial TMDB API call. As such, the foreign films in this problematic subset were either made in English but released in another country or were falsely labeled as being made in English and were released in another country. These films represent the largest and most troubling subset of 'bad data' in our dataset. In order to drop them, we will have to recollect Theatrical Release dates, being sure to select the earliest U.S. release date. We will then have to drop any movies that do not have U.S. theatrical release dates. Finally, because the U.S. release dates will be slightly different than our old release dates, we will have to regenerate the star scores, production company scores, and director scores. Unfortunately, since the third problem was identified later than the other two, this corrective measure has not yet been implemented. As such, all subsequent discussion of our data in this report will still be referring to the dataset that includes this subset of foreign films that need to be removed.

3.1 Select Data

Timeframe for Movies

When we initially collected data from TMDB, we limited our timeframe to 2000 to 2023 for efficiency and predictive reasons. Given our limited computing power, we certainly could not pull data for every single movie in TMDB, which dates back as far as 1888 (Roundhay Garden Scene, the earliest surviving motion-picture film). We additionally recognized the diminishing

relevance of the data for older movies, which were produced and consumed under different economic and social conditions than today. Given these considerations, we opted to look at movies which premiered between 2000 and 2023, as we believed the subset would provide us with the largest feasible amount of data to use for relevant predictions for today.

Once the data was retrieved and we decided to collect Box Office Mojo data, we further restricted our timeframe to 2013 to 2023. When investigating data quality for Box Office Mojo, we found that much of the data for movies pre-2013 was incomplete, likely due to the fact that Box Office Mojo was acquired by IMDb and Amazon only in 2008.

Filtering Out Short Runtimes

While we made every effort to sort out non-feature-length films in our queries, we still needed to filter by runtime due to inaccuracies in user-inputted TMDB data. As such, we filtered out all movies with a runtime of less than 1 hour in this step.

3.4 Integrate Data

Original Methodology

Once we collected all the relevant data from TMDB and Box Office Mojo, we merged the datasets using IMDb ID. We still had conflicting values of budgets and revenues from the two datasets, so we wanted to create merged values for budget and revenue. We prioritized creating as large of a dataset as possible, so, initially, we wanted to include budget and revenue data from IMDb and TMDB in our final values for budget and revenue. To this end, we included all movies with:

1. either a TMDB or Box Office Mojo budget AND
2. either a TMDB revenue or Box Office Mojo worldwide revenue.

When there was conflicting data, we used the decision rule that Box Office Mojo data, which is reported by IMDb, superseded TMDB data, which is more prone to inaccuracies. The second step of our merge methodology specifies that we are merging TMDB revenue with the worldwide revenue from Box Office Mojo. Our initial concern with this step was that these comparisons would not be consistent across the board, since the reported TMDB revenue is either the worldwide revenue or domestic revenue, depending on which figure is most readily available, while the Box Office Mojo revenue is consistently one of three types: worldwide, domestic, or international. As outlined in our business question, we were interested in domestic rather than worldwide revenue, but felt that worldwide Box Office Mojo revenue would be a more comparable figure to the TMDB revenue than domestic Box Office Mojo revenue. This turned out to be a non-issue and is addressed in the section that follows.

Revised Methodology

During our merge, we found that there were only 118 movies with TMDB revenue data but *not* Box Office Mojo revenue data. When we looked into these titles, we found that most of these movies had been released outside of the traditional box office setting, like at a film festival or on DVD; i.e., they were non-theatrical releases. We suspect these miscategorizations are the result of the default release category on TMDB being 'theatrical.' Since there were a relatively small number of these movies, out of an abundance of caution, we decided to exclude these titles entirely. Having excluded movies with only TMDB revenues, we were now free to use Box Office Mojo revenue for all the movies in our dataset. One benefit of doing so was that we could now use domestic revenue rather than worldwide revenue, as we were originally interested in, while keeping our methodology and decision rule in place for budget. That is, under our new methodology, we included movies with

1. either a TMDB or Box Office Mojo budget AND
2. Box Office Mojo *domestic* revenue.

Domestic revenue is more relevant to our research question, so the new resulting dataset better fits our needs. Additionally, because most of the movies that had TMDB revenue but not Box Office Mojo revenue were non-theatrical releases and would not have fit into our model anyway, we are not losing much relevant data by doing so. However, we decided to keep TMDB budget data because there were only 790 movies with budget data from Box Office Mojo as compared to 4801 movies with budget data on TMDB. We intend to verify that this data works as an acceptable substitute for TMDB data but have not yet completed that check.

3.3 Construct Data

Generating Star Scores

One of the primary feature engineering problems we had to overcome was the need to change the categorical variables in our dataset into numerical variables. We wanted to develop a way to convert the cast list, production company list, and director list into numerical values representing the relative importance of those contributors into the overall profitability of the movies.

First, we wanted to transform the cast list into a score representing the star power that a given cast would bring to a movie. As mentioned in section 2.1, we collected data from *The Numbers* for this purpose. *The Numbers* maintains a list of actors and actresses with their relative popularity each year. This relative popularity is called a "star score." As described by *The Numbers:*

> [A] Star Score […] represent points assigned to each of the leading stars of top 100 (based on domestic box office) movies in the current year and two preceding years. For appearing in the number one movie in a year a star gets 100 points, the number two movie 99 points and so on. This rewards actors for appearing in a number of hit films

over the course of three years more than starring in just one monster hit over the same time period.

As an example, Margot Robbie currently has the highest Star Score for 2023 at 315. This is from *Barbie* (2023), *The Suicide Squad* (2021), *Asteroid City* (2023), *Babylon* (2022), and *Amsterdam* (2022). *The Suicide Squad* was ranked 23rd, contributing 78 points. *Babylon* was ranked 57th, contributing 44 points. *Amsterdam* was ranked 59th, contributing 42 points. *Asteroid City* was ranked 50th, contributing 51 points and *Barbie* was ranked 1st, contributing 100 points.

Using the information from *The Numbers* and the cast IDs column of our dataset, we can create an additional column to store a vector of actor star scores for every actor in a film. Note that we want to use star score as a *predictive* variable, so for every actor that appears in the film, we need to collect their star score from the previous year. If we instead used the star scores from the same year as the film, the relationship between actor star scores and the film revenue would be correlated in problematic ways since an actor's star score for a given year increases when they appear in a highly profitable movie that year. As such, for each film in the dataset, we obtained a vector containing the star scores of the actors in the film from the year prior.

There were some issues we ran into while constructing this feature, the biggest being that some of the actor's names were encoded or spelled differently between *The Numbers* and TMDB. We ran a script to add star scores from *The Numbers* into our current database of people from TMDB, but 125 actors could not be matched because of these differences between the two datasets. For those 125 actors, we had to match up the person across the two sources and manually add their star scores to our data.

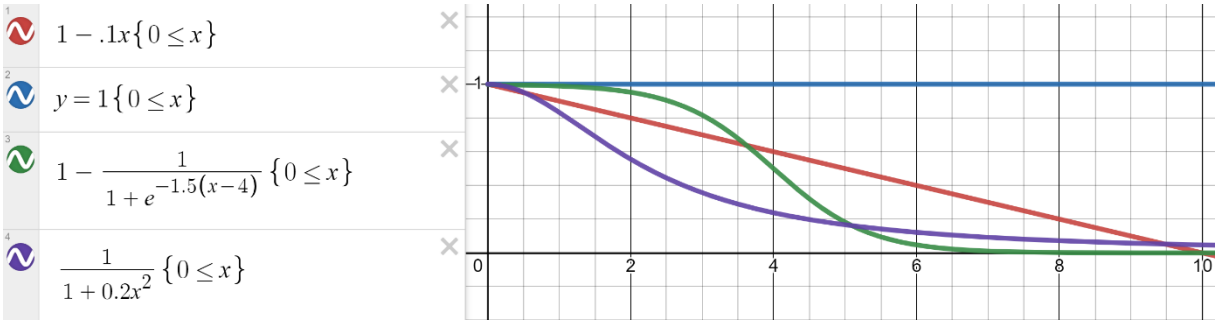
Now that we had our raw vector of star scores for each movie, we had to develop a way to turn that vector into a single number representing the overall star power that the whole cast

brings to the film. The simplest approach would simply be to sum the vector and use that sum as our overall star score. We implemented that approach, but we also wanted to try a few other strategies to derive total star power. These other strategies made use of billing order as weights, the logic being that a top actor having a small cameo in a film should not have the same effect on the movie's star score as a top actor appearing in a leading role. We decided that our weighting scheme needed to satisfy the following criteria:

1. In general, a higher quantity of actors with star powers appearing in a movie should mean the movie has a higher star power.
2. However, the contribution of the individual star score of an actor appearing 10^{th} or later in the billing order to the total star score should be close to zero.
3. After about five actors, there should be a large drop off in weight.
4. Movies with small casts should not necessarily be penalized.

To meet these criteria, we developed a few different ideas for weighting functions. Some of these weighting functions are displayed below:

Figure 1. Various Weighting Functions



Theoretically, these functions are also modular, so we could slightly tweak the parameters of each function and see how the star scores change to see if we can develop an optimal weighting scheme. Two problems emerged as we implemented these weighting schemes. First, the vector of raw star scores is very sparse for the majority of movies: most actors in a

given cast list have a star score of 0 for every year. Secondly, the billing order feature comes from TMDB, which is, of course, user editable. TMDB policy encourages users to input cast in order of importance, but there isn't any system checking that they do so correctly. As such, there is likely a relatively high degree of variation between the official billing order of a given movie, the order of importance of the actors in that movie, and the billing order on TMDB. Though we intend to implement various weighted sums and test them rigorously, there is some degree of doubt as to whether these weighting schemes will result in a better predictor than simply using an unweighted sum.

Generating Director and Production Company Scores

After we obtained our star power features, we wanted to engineer similar numeric variables to measure the impact of the directors and production companies that worked on a given film on its revenue. *The Numbers* had a year-by-year list of star scores for actors, making it easy to obtain those star scores for use in our analysis. Unfortunately, the website does not have a comparable "Director Score" or "Production Company Score" ranking for every year that we can pull. Instead, we needed to engineer that information ourselves.

Within our dataset, we found the 100 movies with the highest domestic revenues for each year. We assigned "contribution points" to those movies based on their place in that ranking (i.e. the movie with the highest domestic revenue each year is assigned 100 points, the movie with the 2nd highest domestic revenue that year is assigned 99 points, and so on). Then, we obtained a set of all the unique director IDs in the entire dataset. We calculated a "director score" for each director for each year using the same scoring scheme that *The Numbers* used. Each year, for every director, we summed all the "contribution points" of the movies they had worked on that year and in the two years before. We want to use director score as a predictive feature, so, for

every movie in the dataset, we created a vector containing the "director scores" of film's directors from the year prior to the film's release. We calculated a vector of "production company scores" for each movie in the same way. Then, to get our final "director score" and "production company score," we did a simple sum of all the elements of each of the two vectors. We may also try to implement an average value of the director scores or the production company scores. Unlike in the case with star-studded casts, we don't necessarily have the requirement that multiple directors or production companies with high scores should result in a higher overall score, so an average-based overall score may be worth implementing.

Although we attempted to derive our production company scores and director scores using the same methodology as *The Numbers,* our dataset differs slightly from the dataset in *The Numbers* because we have several filters that they do not. For example, our dataset excludes non-English films and films without theatrical releases. As such, the way that we allocate points to the movies in our dataset will be slightly different, and our final production company scores and director scores will be slightly different as well.

Date-Based Information

The next step in constructing data was using the release date to get release year, holiday and season. We think that a movie's revenue will be affected clearly by the year it is released in, but also wanted to include something about when in the year it was released. On holidays for example, we would expect more people to be off work and therefore more people at movie theaters, so if a movie opened on a holiday weekend it might have higher revenue. Another feature that we can pull from release date is the season that a movie is released in as a length 4 one-hot-encoded vector to represent winter, spring, summer and fall. Summer is usually when big blockbusters are released, partially due to school not being in session, which can mean more

traffic for theaters. January to May is generally viewed as a dead zone for movie releases. Knowing what season a movie is released in could give the model important information about how the movie will perform.

Budget Floor

As mentioned in Section 3.4, we merged TMDB and Box Office Mojo budgets into a single merged budget feature, preferring Box Office Mojo to TMDB when available. However, some of the budgets in the new dataset were near zero or otherwise appeared unusually low. For these movies, we implemented an extremely conservative estimate of a budget floor of $100,000 as the lower bound of all budget figures.

Similarity Scores

We wanted to make note that while it was not explicitly an engineered feature in our dataset, we generated Jaccard similarity scores for cast similarity and genre similarity as part of data construction. Our hope was that we would be able to use these scores for spectral embedding in our modeling step. The analysis for cast similarity proved unfruitful due to its limited application to test data, but the similarity scores for genre will be used to perform spectral embedding and clustering analysis to group genres together during our modeling process. This process will be documented in more detail in our final report.

2.2 Describe Data

In total, there are 37 features and 1 indexing column. Of the features, 18 are numerical, and the other 19 are categorical. All of the features are shown in Table 1. Some features are duplicates of a sort, such as “Genre” and “Genre ID” or “Release Date” and “Release Year.” Since some features like “Genre” and “Genre ID” are redundant and categorical variables, they will be dropped in the format data step. “Genre ID” comes from TMDB, while “Genre” is

gathered from Box Office Mojo. However, in this process we are omitting some minor differences between the descriptions of genres. For example, Box Office Mojo differentiates between a "Sport Documentary" and a "History Short." However, TMDB does not, and assigns them to the same Genre ID of "99". We do not expect such a distinction to significantly affect our model's predictive power, but with multiple data sources, we have the capacity to reincorporate some of this lost data. With the exception of four financial features describing budget and revenue, the engineered features for star power, and two features describing run time, all variables are categorical.

We used feature engineering to make use of some of the categorical variables such as the "cast_ids" and the "director_ids" by associating them with rankings or existing popularity metrics to devise a star power for each personnel associated with the production and filming, as described in the previous section. This required the creation or implementation of additional dictionaries of cast, director, and production company rankings by year. In total, there are 9,806 movies with non-zero revenues from at least one data source that we are able to use to build our model.

Table 1. Variable Descriptions

Variable Name	**Type**	**Notes**
Unnamed: 0	Categorical	Index
Running Time	Numerical	Formatted as [] hrs [] mins
Genres	Categorical	
IMDb Title	Categorical	
MPAA	Categorical	Maturity Rating
Domestic Distributor	Categorical	
Domestic Opening	Categorical	
Earliest Release Date	Numerical	
IMDb Budget	Numerical	
TMDB Budget	Numerical	
Genre ID	Categorical	

Genre Name	Categorical	
IMDb ID	Categorical	
Production Company ID	Categorical	
Production Company Name	Categorical	
Release Date	Numerical	
TMDB Revenue	Numerical	
Runtime	Numerical	Integer
TMDB Title	Categorical	
TMDB ID	Categorical	
cast_ids	Categorical	Row Array
order	Categorical	Cast Order
director_ids	Categorical	
IMDb Domestic Revenue	Numerical	
international_revenue	Numerical	
worldwide_revenue	Numerical	
Release Year	Numerical	
Merged Budget	Categorical	See Merge Methodology
Merged Revenue	Categorical	See Merge Methodology
Raw Star Scores	Numerical	Array
Unweighted Star Score	Numerical	Uniform Sum
Simple Weight Star Score	Numerical	Linear Weighted Sum
Log Weight Star Score	Numerical	Log Weighted Sum
Exponential Weight Star Score	Numerical	Exponential Weighted Sum
director_scores	Numerical	Uniform Sum
production_company_scores	Numerical	Uniform Sum

3.5 Format Data

Formatting Our Final Dataset

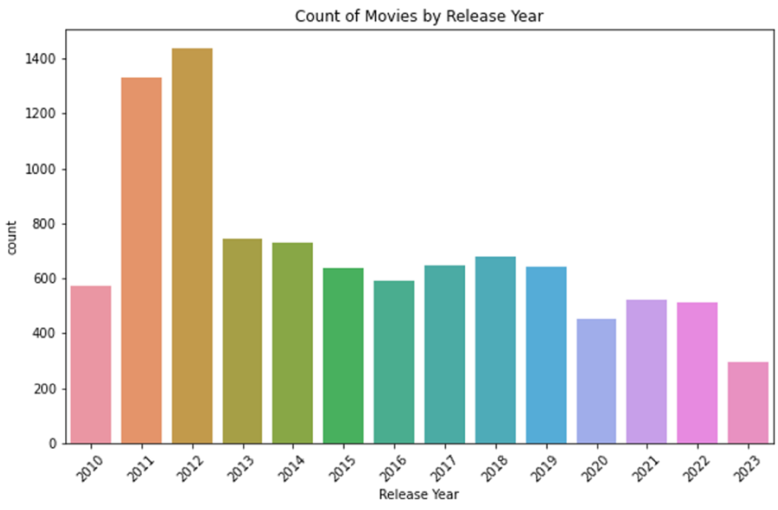
Once we had finished the construction and integration of our dataset, we implemented a few minor formatting steps to make it suitable for use in modeling. We dropped any index columns and redundant columns that had been added to the dataset while combining various other datasets. We also renamed all the columns to add greater clarity and legibility to our dataset.

While we have not yet done so, the next formatting step that must be implemented is to drop all the columns that won't be used in our model. We won't be using international_revenue, worldwide_revenue, TMDB_revenue, IMDB_revenue, IMDB_budget, or TMDB_budget, since we will be using our engineered budget and revenue features instead. Similarly, categorical variables like IMDB Title, TMDB Title, Genres, Genre Name, Production Company Name, Director IDs, Production Company IDs, cast_ids, order, Genre ID will be dropped, since we have captured the relevant information from those categories in our engineered features (our engineered measures of genre similarity, production company score, director score, and star score). Finally, we obtained the Domestic Distributor, Domestic Opening, and Earliest Release Date features from Box Office Mojo, but we are not currently planning to use them in our model, so those categories will also be dropped from the final dataset.

Other syntactic changes may be required as we begin applying our modeling tools and will be implemented as needed. For example, we may need to reorder our dataset or alter the data types of a column to satisfy the requirements of a modeling tool.

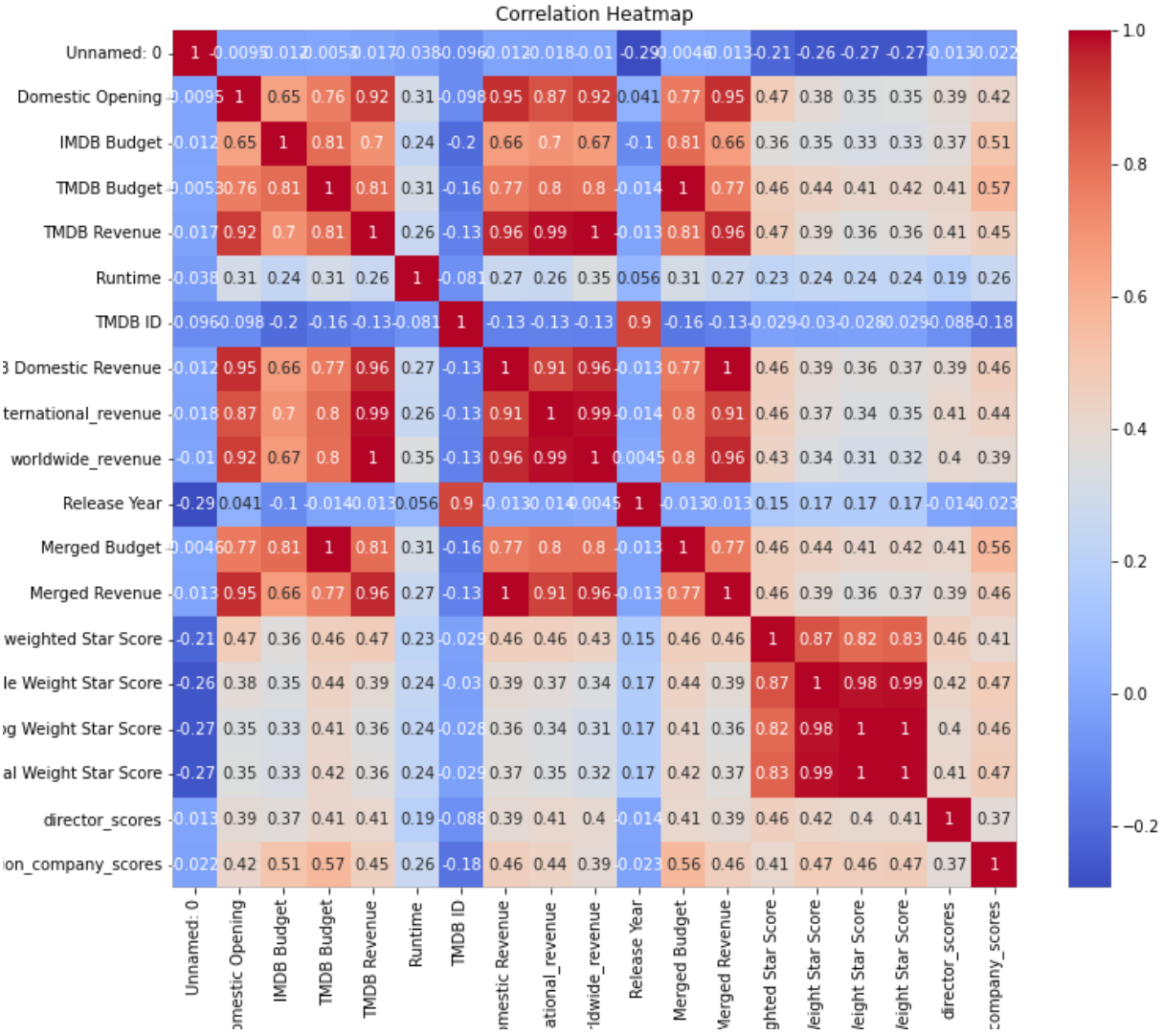
2.3 Explore Data

Figure 2. Histogram of Movies by Release Year



The number of movies released per year seems generally consist across most years, with the exception of 2011 and 2012. As expected, we see a slight decrease during and post COVID-19. We will need to investigate further why the number of releases in 2011 and 2012 seems so high—however, since we are only using data from those years for production company score and director score computation, any abnormalities will have a limited impact on our predictive model.

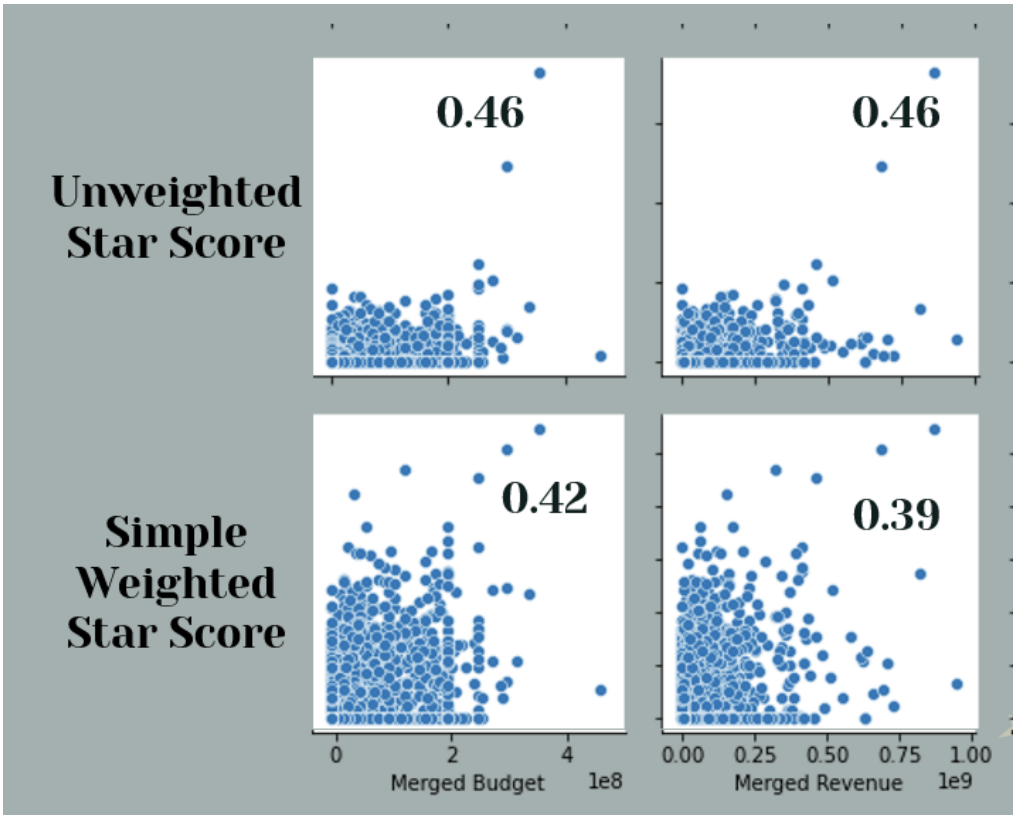
Figure 3. Heatmap of Numeric Variables



Using the above heatmap of the correlation coefficients, we observed expected high correlations between both scraped and engineered financial features (i.e. budget and revenue).

There was also a mild correlation of some of the engineered score variables with budget, which is encouraging.

Figure 4. Merged Budget and Revenue vs. Unweighted and Simple Weight Star Score



Above are a few examples of scatterplots for selected pairs of relevant variables. We see that the unweighted star score is slightly more correlated with both the merged budget and the merged revenue. We speculate there to be a few reasons why: it could be that the billing order data from TMDB is not a reliable proxy for actual billing order, our weights are incorrect, or our assumptions about how individual star scores should be aggregated are incorrect.

These are just the first steps of our exploratory data analysis, and they will need to be redone after we correct some of the errors in our dataset. We are excited to see how the relationships of our data change after some of the corrections.

Conclusion

To summarize, our data preparation within the CRISP-DM framework has been marked by a series of obstacles. Throughout the project, we have encountered and addressed challenges related to data collection, cleaning, and formatting. However, we have successfully engineered and compiled a dataset with valuable information which can now be used to create our predictive models.