# A Combined Ranking Based on IMDB and TMDB

## Introduction

Our team consists of two members: **Zhengshu Zhang** (USC ID: 4664738582, email: zhengshu@usc.edu, GitHub: forestzs) and **Zhuoran Deng** (USC ID: 5873397495, email: dengzhuo@usc.edu). Within the team, Zhengshu is mainly responsible for the project topic design, data source selection, data collection, and data preprocesses like the cleaning and selection, while Zhuoran mainly focuses on the statistical analysis and result visualization of the cleaned datasets. And we usually discuss the code design and the function's detailed realized methods together to guarantee the fine logic connection of this project.

This project is driven by the following research questions: To what extent are movie quality, popularity among users, and box office success consistent? How can we combine them into a single, standardized ranking model? In practice, streaming platforms and audiences rely on multiple signals such as ratings, online popularity, and financial performance when deciding what to watch, but these dimensions are not always consistent. In this project, we constructed a small-scale movie ranking system that integrates three signals from public data sources: long-term quality indicators from IMDB ratings, short-term popularity ratings from TMDB, and financial performance indicators based on TMDB revenue data. By normalizing these variables and combining them into a joint representation, our goal is to investigate where the three views on 'success' are consistent, where they diverge, and which types of movies perform consistently across all dimensions.

## Data Collection and Cleaning

In this final project, we collected three datasets related to movie quality ratings, describing different dimensions of whether a movie is "successful": long-term quality (rating related), popularity, and financial performance (box office related). These three datasets were all built using Python scripts and exported as CSV files, so that our analysis can be copied from the original source. In this section, we will explain the specific data collected by the project, how the data was collected, how many reference samples were roughly obtained, what changes have occurred compared to the original plan, and the main data cleaning steps.

For long-term quality, we use the official public dataset provided by IMDB that can be directly downloaded, rather than crawling the IMDB website's API. IMDB is available https://datasets.IMDBws.com/. We manually downloaded the title. basics. tsv and title. ratings. tsv from the portal. The first file contains metadata such as a unique identifier, title type, main title, year, etc; The second one includes the average rating and number of votes. We placed these files in the project folder and built a movie rating table with a clear structure suitable for direct analysis using a Python script (clean_data_IMDB_ratings.py). This script uses pandas to read TSV files, retaining only the feature film (title Type == "movie") and all age oriented movies, converting startYear to a number and removing

invalid years, converting averageRating and numVotes to numbers, and removing records with missing values. We also applied a minimum voting threshold (e.g. numVotes >= 1000) to focus on movies with meaningful ratings. Then merge these two tables on tconst, rename the columns to imd_id, title, year, average Rating, and numVotes, and save the results as imd_ratings.tsv. The original IMDB file contains hundreds of thousands of titles; After filtering by type, age marker, and number of votes, our rating table still includes tens of thousands of movies, although in future analysis, we will only use subsets that overlap with TMDB samples.

To indicate the popularity of the current movie, we use the API of the Movie Database to obtain data, which provides "popularity" scores and voting statistics. We first registered a free level developer account on TMDB's official website to obtain API keys. The scripts used call this endpoint sequentially for multiple pages; Return 20 popular movies per page. In the page paging configuration of the script, the code will attempt to request multiple pages, so we obtained approximately one to several hundred different movies. For each movie, the project extracts the TMDB ID, title and original title, release date, popularity rating, as well as the average TMDB rating and number of votes. At the same time, we also removed rows without tmd_id, removed duplicate items with the same attribute, and exported the results as tmd_populist.csv using UTF-8-SIG encoding, so that non ASCII titles can be displayed correctly in Excel. In the analysis, we mainly use tmd_id, title, population, and voting statistics; Original_title is reserved as optional metadata.

For financial performance, we choose to rely on the TMDB API to obtain data. This time, we use the detailed endpoint/movie/{id} to obtain the production budget and global box office. The first script (get_data_TMDB_revenue.py) starts from tmd_popularity raw csv , reads the unique list of tmd_id values, and then calls/movie/{id} for each movie in the list. For each response, the script will record the TMDB ID, title, original title, release date, budget, revenue, and runtime. We insert a brief delay between requests to avoid data retrieval interruptions caused by API rate limitations. A tricky is the server skip the id:1583907, so we added it by hand based on the prior data from website. For data cleaning, the final collected records are merged into one DataFrame, cleaned up by removing rows without valid TMDBoids and removing duplicates, and saved as TMDBorevenue.csv. The number of lines in this file matches the number of unique movies in popularity csv, so each popular movie in our sample has corresponding financial records. However, many games have zero revenue in the API. We consider these as missing or unreported values, rather than true zero ticket rooms, and keep them in the dataset, taking note of this limitation for future analysis.

Compared to our initial proposal, there have been significant changes in our data collection strategy. Initially, we planned to directly crawl three data sources through a Python web crawler: IMDB Top 250 as long-term quality, IMDB's MOVIEmeter "Most Popular Movies" as short-term popularity, and Box Office Mojo's weekly chart as box office. After communicating with the proposal and the instructor, we learned that IMDB and Box Office Mojo have powerful anti crawling protection functions. Previous projects encountered technical and policy issues when crawling these websites. The professor also pointed out in detail that the overlap between the top 250 "currently most popular" movies (mainly old classics) and the current weekend box office may be small, which will make it difficult to compare the quality, popularity, and revenue of the same film. In response, we completely abandoned HTML based data scraping and redesigned our data acquisition pipeline around official download links and official API style sources: IMDB's public TSV dump for ratings, TMDB's "popular" list plus movie details for

popularity and revenue. In the new design, TMDB popular movies first define the sample; then, we directly attach a popularity rating and revenue to each movie from TMDB, and ultimately match it with the IMDB rating by title and year. This method is technically more robust, more in line with acceptable data usage, and produces a dataset where each movie can represent all three dimensions.

**Statistical Analysis and Visualization**

For the analysis stage, we focus on combining the three cleaned datasets into a single, comparable framework and then quantifying how movie quality, popularity, and financial performance relate to each other. All analysis was implemented in Python using pandas, numpy, scikit-learn, and matplotlib, and operates only on the subset of movies that appear in all three sources (IMDB ratings, TMDB popularity, and TMDB revenue).

We first merged the three datasets into a single table using the TMDB ID as the key between the two TMDB tables, and the movie title (optionally combined with year) as the link between TMDB and IMDB. This produces a matched sample in which every movie has an IMDBrating, a TMDB popularity score, and a TMDB revenue value. Movies without valid revenue (missing or zero), missing ratings, or missing popularity scores are removed from the analytical sample.

Because raw box office values are extremely skewed, we take the base-10 logarithm of TMDB revenue to obtain a more symmetric log-revenue variable. This transformation reduces the influence of a few mega-blockbusters and makes linear relationships easier to interpret. To put all three indicators on a common scale, we then apply min–max normalization to each variable:

- IMDB average rating
- log-transformed revenue
- TMDB popularity

Each of these is rescaled to the [0,1][0, 1][0,1] range. The resulting normalized variables (rating_norm, revenue_norm, popularity_norm) are directly comparable in magnitude and serve as the inputs to the principal component analysis.

To build a single, data-driven ranking that reflects multiple aspects of success, we apply Principal Component Analysis (PCA) to the three normalized indicators. Conceptually, PCA searches for new axes (principal components) that are linear combinations of the original variables and that explain as much variation as possible in the data. In our case, PCA produces three principal components, denoted PC1, PC2, and PC3, which are uncorrelated with each other and ordered by the amount of variance they explain.

The first principal component (PC1) is interpreted as a combined performance dimension. Its loadings assign positive weights to all three inputs, with particularly strong contributions from

normalized rating and normalized log-revenue. This suggests that PC1 captures a traditional notion of movie "success" that blends long-term audience evaluation and financial performance, with online popularity playing a secondary but still positive role. The second component (PC2) places most of its weight on popularity and relatively little on rating or revenue, indicating that it describes a more "pure" popularity dimension that is only weakly aligned with long-term quality and box office. The third component explains only a small residual fraction of the total variance and is not used further.

We then define a PCA-based performance index by taking each movie's score on PC1. Sorting this index in descending order yields a unified ranking that balances the three dimensions without arbitrarily hand-picking weights. The ranked table, exported as movie_pca_ranking.csv, allows us to identify films that perform consistently well across rating, popularity, and revenue, as well as those that are strong in only one or two dimensions (for example, critically acclaimed but financially modest films, or high-grossing blockbusters with relatively low ratings).

To directly address the question of how closely the three signals agree, we compute pairwise Pearson correlation coefficients between IMDB rating, log-revenue, TMDB popularity, and the PC1 performance index. These correlations are summarized in a correlation matrix heatmap, where color intensity reflects the strength and direction of each relationship and the numerical values are overlaid in each cell.

The results show a moderate positive association between IMDB rating and log-revenue, meaning that higher-rated films tend, on average, to earn more at the box office, although the relationship is far from perfect. In contrast, TMDB popularity is only weakly correlated with both rating and log-revenue in this sample, consistent with the idea that TMDB's popularity score captures a more short-term, attention-driven signal that does not always align with long-term critical reception or lifetime gross. The PC1 performance index is positively correlated with all three input variables by construction, but the correlations are strongest with rating and log-revenue, again reinforcing the interpretation of PC1 as a combined quality-and-revenue axis.

To complement the correlation heatmap, we also generate a two-dimensional scatter plot of log-revenue versus TMDB popularity, with each point representing a single movie and point color encoding the IMDB rating. This visualization highlights several patterns: clusters of financially and popularly successful titles with mid-to-high ratings, "sleeper hits" with high ratings but moderate popularity, and highly popular movies whose ratings are relatively low. Together with the PCA-based ranking, these plots make it easier to visually identify films where the three notions of success—quality, popularity, and box office—are aligned and cases where they diverge.

**Limitations and Future Work**

During the analysis, we noticed an important issue in the merged dataset: some movie titles appear multiple times in the final table. Upon inspection, these are not simple duplicates but often correspond to different versions or adaptations of the same story under the same title. A clear example is A Christmas Carol, which has multiple film versions released in different years. Because our current matching strategy primarily relies on the movie title (with only limited consideration of year), these versions can be incorrectly blended together, leading to "mixed" records where ratings, popularity, and revenue from different adaptations are treated as if they belong to a single movie.

This has implications for our combined ranking and PCA-based performance index. In such cases, the PC1 score and overall rank may implicitly reflect the aggregate behavior of several adaptations rather than a single, well-defined film. While the number of affected titles is relatively small compared to the full sample, this issue can still introduce bias in the interpretation of specific movies, especially well-known properties that have been remade many times.

In future work, we plan to address this by introducing stricter year-based matching and filtering. Instead of matching only on the title, we will treat "title + release year" as the primary movie identifier, and require that TMDB and IMDB entries agree within a narrow year range (for example, a difference of at most one year) before they are merged. For multi-version titles like A Christmas Carol, we can either explicitly select a target version (only the 2009 film) or treat each "title + year" combination as a separate movie throughout the analysis. Additional metadata such as runtime, original title, or even director information could also be incorporated into the matching logic for higher precision. By tightening the matching criteria in this way, future iterations of the project can produce cleaner one-to-one mappings between IMDB and TMDB records and yield more accurate rankings and component scores for each individual film.