California State Polytechnic University, Pomona

Movie Success Prediction Using IMDb Ratings and Metadata

Optimizing entertainment investments through Big Data

By

An Vu

Kiet Nguyen

GBA 6430 - Big Data Technology in Business

Summer, 2025

Dr. He Zhang

8/15/2025

Table of Contents

Introduction

Introduction to the Business Problem and Role of Big Data

In today's streaming-driven entertainment economy, understanding what drives higher movie ratings is essential for platforms like Netflix, Disney+, and Amazon Prime. This intelligence can help optimize content acquisition, recommend personalized viewing options, and enhance user engagement. The movie industry generates vast volumes of data across decades, involving variables such as genre, runtime, cast, crew, and viewer ratings. Traditional analytics approaches cannot handle such variety and volume efficiently. This is where big data analytics becomes vital—enabling scalable querying, advanced filtering, and trend analysis over millions of records in real time.

Dataset Overview and Its Business Relevance

To tackle this challenge, we built a custom dataset named imdb_prepared by integrating five key tables from the IMDb public dataset on Google BigQuery:

- title_basics (movie metadata: genre, type, year, runtime)
- title_ratings (average ratings and vote count)
- title_crew (directors and writers)
- title_principals (main cast and crew roles)
- name_basics (names, birth years, professions)

By joining these datasets via primary keys (e.g., tconst, nconst), we created a wide, analyzable table with over 21 million records. This enabled us to explore key business questions such as:

- Which genres or title types receive the highest ratings?
- Is there a correlation between runtime and rating?
- Do top-rated directors or famous actors lead to better-rated movies?
- How have movie ratings and volumes evolved by decade?

Key Insights

The unified dataset allowed us to derive actionable insights, such as identifying that top-rated directors and famous actors significantly improve movie ratings. Moreover, ratings have trended upward over the decades, and genres like History, Documentary, and Animation consistently rank high. These insights empower stakeholders to make data-driven production, marketing, and casting decisions in a highly competitive content market.

Exploratory Data Analysis (EDA)

Schema Exploration and ERD Design

We began by exploring the schema of the IMDb dataset using BigQuery using 4 steps following:

1. Schema Discovery: Queried all table and column names in the bigquery-public-data.imdb dataset.
2. Shared Columns: Identified which columns appeared across multiple tables to determine potential join keys.
3. Pivot-Like View: Created a matrix view showing which columns appeared in which tables. (Table 1)
4. Primary Key Identification: Used Excel to filter and identify primary keys.(Table 2)

From this exploration, we identified the key columns needed to link the tables. We examined the dataset in detail to pinpoint primary keys that uniquely identify records within each table. Steps 3 and 4 served the same purpose both focused on primary key identification—but differed in presentation: one displayed results in the BigQuery console, while the other presented them in an Excel table.

column_na	title_basic	title_rating	title_akas	title_episo	title_crew	title_princi	name_bas	reviews	table_cour
tconst	1	1	0	1	1	1	0	0	5
title	0	0	1	0	0	0	0	1	2
nconst	0	0	0	0	0	1	1	0	2
ordering	0	0	1	0	0	1	0	0	2
review	0	0	0	0	0	0	0	1	1
split	0	0	0	0	0	0	0	1	1
label	0	0	0	0	0	0	0	1	1
movie_id	0	0	0	0	0	0	0	1	1
reviewer_r	0	0	0	0	0	0	0	1	1
movie_url	0	0	0	0	0	0	0	1	1
parent_tco	0	0	0	1	0	0	0	0	1
season_nu	0	0	0	1	0	0	0	0	1
episode_n	0	0	0	1	0	0	0	0	1
primary_n	0	0	0	0	0	0	1	0	1
birth_year	0	0	0	0	0	0	1	0	1
death_yea	0	0	0	0	0	0	1	0	1
primary_pı	0	0	0	0	0	0	1	0	1
known_for	0	0	0	0	0	0	1	0	1
average_ra	0	1	0	0	0	0	0	0	1
num_votes	0	1	0	0	0	0	0	0	1
title_id	0	0	1	0	0	0	0	0	1
region	0	0	1	0	0	0	0	0	1
language	0	0	1	0	0	0	0	0	1
types	0	0	1	0	0	0	0	0	1
attributes	0	0	1	0	0	0	0	0	1
is_original	0	0	1	0	0	0	0	0	1
directors	0	0	0	0	1	0	0	0	1
writers	0	0	0	0	1	0	0	0	1
title_type	1	0	0	0	0	0	0	0	1
primary_tit	1	0	0	0	0	0	0	0	1
original_tit	1	0	0	0	0	0	0	0	1
is_adult	1	0	0	0	0	0	0	0	1
start_year	1	0	0	0	0	0	0	0	1
end_year	1	0	0	0	0	0	0	0	1
runtime_m	1	0	0	0	0	0	0	0	1
genres	1	0	0	0	0	0	0	0	1
category	0	0	0	0	0	1	0	0	1
job	0	0	0	0	0	1	0	0	1
characters	0	0	0	0	0	1	0	0	1

Table 1. IMDb Table-Column Mapping Matrix

Row	column_name	title_basics	title_ratings	title_akas	title_episode
1	tconst	1	1	0	1
2	title	0	0	1	0
3	nconst	0	0	0	0
4	ordering	0	0	1	0
5	review	0	0	0	0

Table 2. IMDb Schema Presence Matrix

With these keys identified, we could now define our analytical goals. This step ensured we were clear about which tables and relationships were relevant for answering our business questions earlier

Analytical Goals

With the schema and keys mapped, we defined our analytical objectives to guide the EDA:

Rating-Based

1. Which genres have the highest average ratings?
2. Which movie types (title_type) are rated 5 the most?
3. Is there a correlation between runtime and rating?

Crew & Cast Impact

4. Do movies directed by top-rated directors tend to score higher?
5. Does having famous actors relate to better ratings?

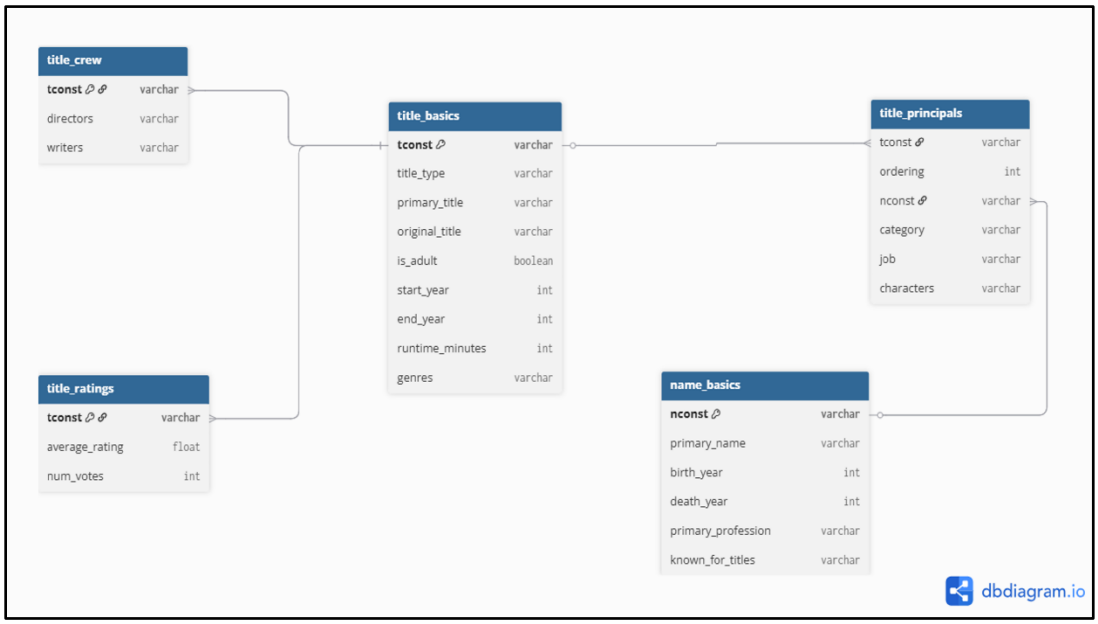
Temporal Trends

6. How have ratings changed over decades?

From this, we created an Entity Relationship Diagram (ERD) for five key tables:

- *title_basics*: title_type, genres, runtime_minutes, start_year
- *title_ratings*: average_rating, num_votes
- *title_crew*: directors, writers
- *title_principals*: actor/actress identifiers and roles
- *name_basics*: actor/director names, professions, birth/death years

Based on these goals, we mapped the relationships between our chosen tables, creating an Entity Relationship Diagram (ERD) to visualize how the data connects:

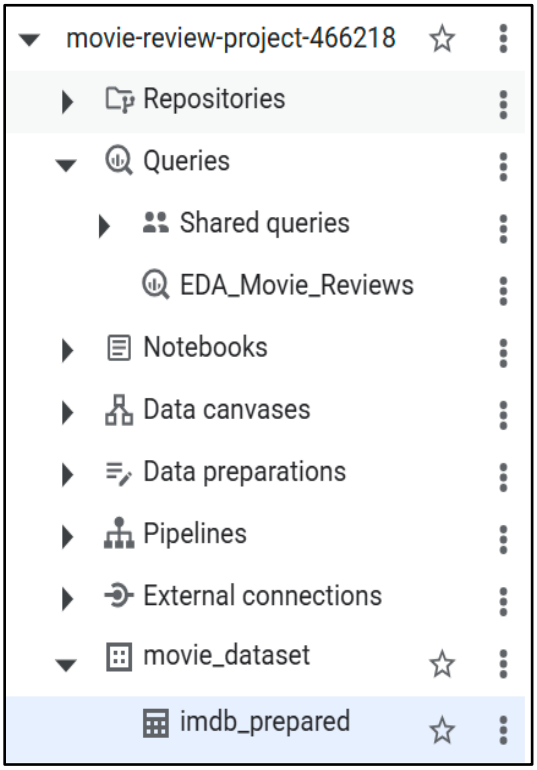


Entity Relationship Diagram (ERD)

Summary of Table Relationships in ERD

Table	Key Column	Connected To	Purpose
title_basics	tconst	Central Table	Movie details
title_ratings	tconst	→ title_basics	Rating data
title_crew	tconst	→ title_basics	Director/writer info
title_principals	tconst, nconst	→ title_basics, → name_basics	Cast/crew members
name_basics	nconst	← title_principals	Actor/crew details

This relationship mapping guided the process of joining our tables into a single dataset. By merging them using their key columns, we created a comprehensive dataset imdb_prepared ready for analysis



Comprehensive dataset called imdb_prepared

The imdb_prepared dataset consolidates all relevant movie details, ratings, crew and cast information, and individual person data into a single view. This unified dataset serves as the foundation for all subsequent exploratory analysis and modeling

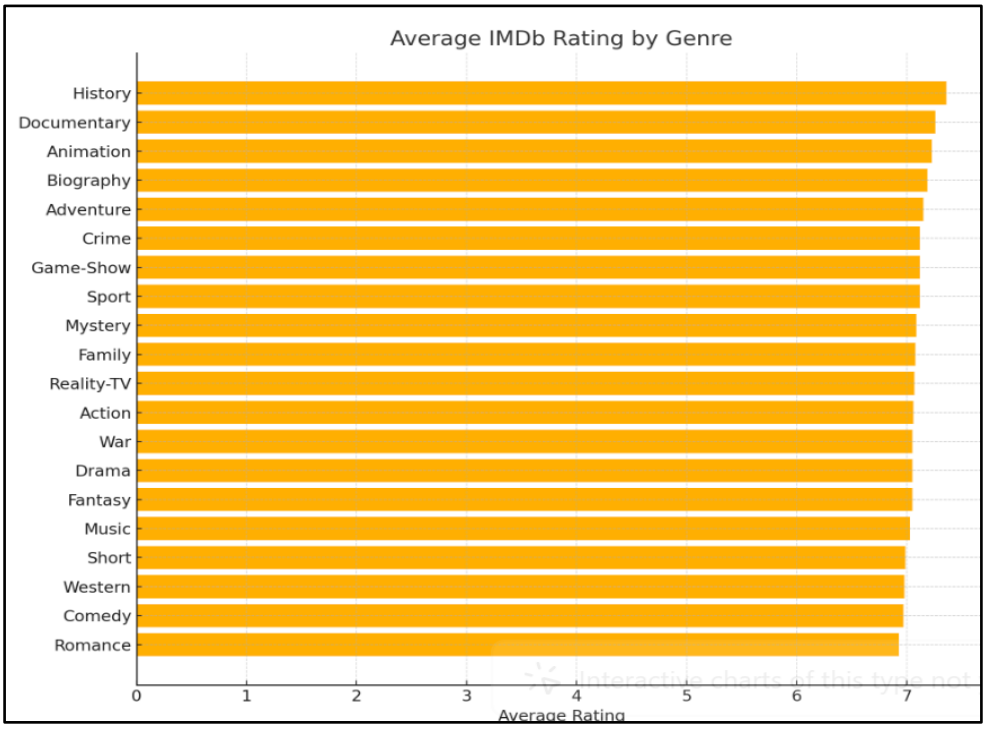
Visualization Section

Rating-Based Analysis

With the unified imdb_prepared dataset in place, we moved into the analytical phase. By leveraging the consolidated ratings, genres, crew, and cast information, we began our Rating-Based Analysis to answer the first research question: Which genres have the highest average ratings?

To find out, we split the genres column into individual categories and calculated the aggregated average rating for each genre. The top genres ranked by average rating were:

genre	num_titles	avg_rating
History	649252	7.36
Documentary	1965174	7.26
Animation	2784410	7.23
Biography	442488	7.19
Adventure	2884676	7.15
Crime	2839274	7.12
Game-Show	504118	7.12
Sport	390135	7.12
Mystery	1275730	7.09
Family	1651806	7.08
Reality-TV	958995	7.07
Action	3270566	7.06
War	249737	7.05
Drama	8812917	7.05
Fantasy	986644	7.05
Music	669998	7.03
Short	1772409	6.99
Western	271757	6.98
Comedy	7270316	6.97
Romance	1975395	6.93



As a result from above visulization, we can see that historical, documentary, and animation films dominate the top-rated genres, often due to higher production quality, strong storytelling, and targeted audience appeal. These genres tend to attract dedicated viewers, which can lead to more favorable ratings. In contrast, high-volume genres like comedy or romance, while popular, show slightly lower averages, possibly due to broader audience expectations and varying quality levels within the category.

Which movie types are most frequently rated exactly?

We identified the types of content most frequently rated exactly 5. The results are shown in the table below, with "movie" being the most common title type receiving a rating of 5.This suggest that these type of gernes may represent the largest share of total content so they receive more viewer ratings, increasing the chance of averaging to 5; or they may need quality improvement.

title_type	**count_rated_5**
movie	89442
tvEpisode	45555
short	15746
tvMovie	12473
video	10047
tvSeries	9245
tvMiniSeries	1595
tvSpecial	1430
videoGame	971
tvShort	215

To confirm our assumption, we filter the dataset to include only rows with valid average ratings and vote counts. It then categorizes the data into two comparison groups: titles where the genre includes *"History"* are labeled as *"History Genre"* and titles with the type "movie" are labeled as *"Movie Type"* For each group, the query calculates the total number of titles, the sum of all rating votes, and the average rating score. This allows us to compare the performance and popularity of historical content versus general movie content.

The result in *Table 3* supports the assumption that quality of "movie "is lower than "history" while "movie" is the most frequent type rated exactly 5, not all movie content performs equally in terms of quality. In contrast, historical content though less common—tends to be rated more favorably, potentially due to higher production value, educational value, or niche audience interest.

Row	comparison_group	num_titles	total_ratings	avg_rating
1	History Genre	649252	898080117	7.36
2	Movie Type	5094351	25677566174	6.06

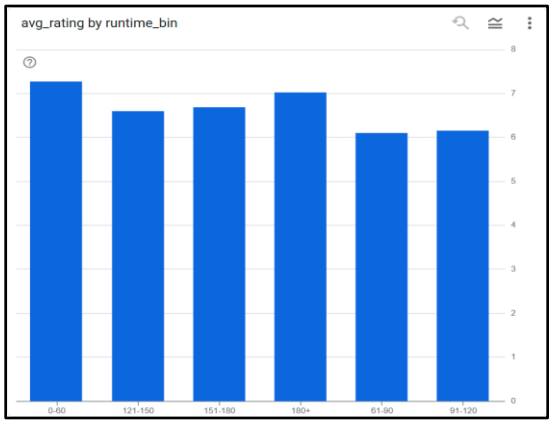
Table 3. IMDb Comparison Group Ratings Summary

Is there a correlation between runtime and rating?

We explored whether longer runtimes correspond to higher ratings by:

- Grouping runtime into bins (e.g., 0–60 mins, 61–90 mins, etc.)
- Calculating average rating per bin

Insight: Moderate-length films (90–120 mins) tend to receive better ratings. Very short or very long movies show more variability.



Crew & Cast Impact: Director Ratings

We defined top-rated directors as those who have directed at least five movies with an average rating of 7.5 or higher. Our analysis reveals that movies directed by these individuals consistently receive significantly higher ratings compared to those by other directors, highlighting the strong influence of directorial quality on audience reception.

Job information | Results | Chart | JSON | Execution details

Row	director_quality	num_movies	avg_rating
1	Other	17749308	6.72
2	Top-rated	4243966	7.93

Do famous actors relate to better ratings?

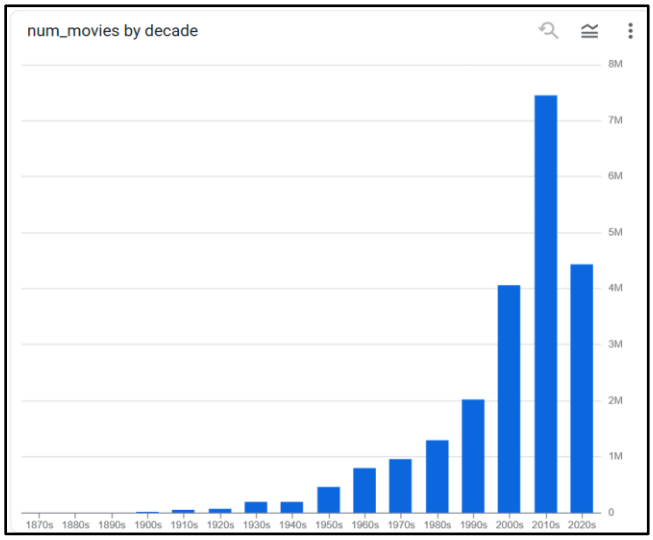
We defined a famous actor as one who has appeared in at least ten movies with an average rating of 7.5 or higher. The results indicate a strong positive correlation between casting such actors and achieving higher average movie ratings, suggesting that star power can play a significant role in enhancing a film's overall reception.

Query results

Job information | Results | Chart | JSON | Execution details

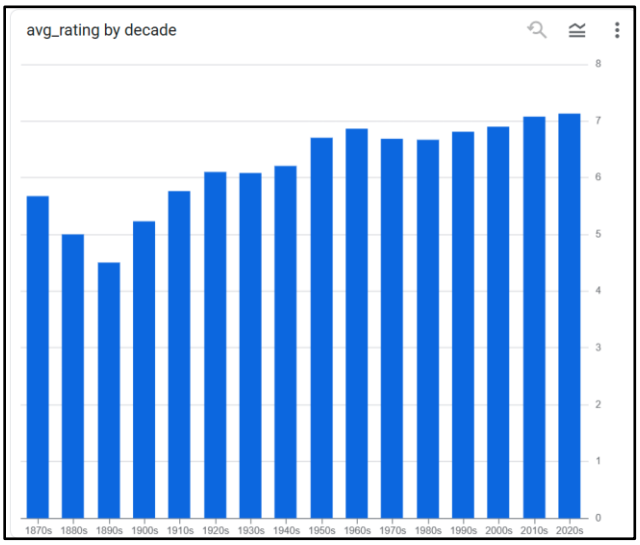
Row	actor_quality	num_movies	avg_rating
1	Famous actor	39502284	7.82
2	Other	334044570	6.86

Temporal Trends – "How have ratings changed over decades?



- There has been explosive growth in movie production since the 2000s.
- The 2010s saw the highest output.
- The 2020s dropped slightly (possibly due to incomplete data or COVID-19 impact).
- Ratings were relatively low and inconsistent in early decades (1870s–1910s).

- From the 1920s onward, there's been a steady upward trend.
- Highest ratings appear in the 2010s and 2020s, with average ratings above 7.0



Data Extraction and Cleaning

Starting with the imdb_prepared dataset, we initiated the cleaning process by filtering for complete and realistic records. First, we removed rows with missing critical values (tconst, title_type, genres, start_year, average_rating, runtime_minutes) and applied validity checks to keep only records with release years ≥ 2000 and runtimes between 1 and 400 minutes. We then assessed data quality by counting duplicate rows, identifying them through a detailed grouping of all key columns, and removing duplicates using SELECT DISTINCT.

Next, we explored the genres field by splitting comma-separated values, counting the number of unique genres, and listing them in a cleaned, standardized format. Finally, we verified the size of the resulting dataset by checking the total row count in the imdb_movie_only table.

This process ensured the dataset was consistent, free of invalid entries, de-duplicated, and well-understood in terms of genre diversity before moving on to analysis. In addition, we engineered a new column, **success**, based on the formula:

$$\textbf{success} = 1 \text{ if } (\text{average\_rating} \times \log_{10}(\text{num\_votes})) \geq 25$$

We will introduce this metric in more detail later in the report. The SQL query below was used to extract, filter, and engineer this success label from the IMDb dataset hosted on Google BigQuery:

```sql
SELECT
  b.primary_title,
  b.start_year,
  b.runtime_minutes,
  b.genres,
  r.average_rating,
  r.num_votes,
  CASE
  WHEN (r.average_rating * LOG10(r.num_votes)) >= 25 THEN 1
  ELSE 0
END AS success
FROM
  `bigquery-public-data.imdb.title_basics` AS b
JOIN
  `bigquery-public-data.imdb.title_ratings` AS r
ON
  b.tconst = r.tconst
WHERE
  b.title_type = 'movie'
  AND b.start_year >= 2000
  AND b.runtime_minutes IS NOT NULL
  AND b.genres IS NOT NULL;
```

Purpose

This query prepares the dataset for *supervised machine learning* by selecting relevant features and engineering a binary target variable.

Filtering Criteria

The query includes the following constraints:

- Only records with title_type = *'movie'*
- Movies released from the year *2000* onward
- Non-null values for *runtime_minutes* and *genres*

These conditions ensure data quality and relevance to recent cinematic content.

Output Fields

After cleaning, the dataset remains 2,826,960 rows. The final dataset includes:

- *primary_title*: The movie's name
- *start_year:* Release year
- *runtime_minutes*: Duration in minutes
- *genres*: Associated genre tags
- *average_rating*: IMDb rating

- *num_votes*: Number of user votes
- *success*: Binary label indicating high-impact titles

Query results

Save results | Open in

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	primary_title	start_year	runtime_minutes	genres	average_rating	num_votes	success
1	For the Cause	2000	100	Action,Adventure,Drama	3.4	857	0
2	Karobaar	2000	180	Drama,Romance	3.7	318	0
3	The Highwayman	2000	94	Comedy,Drama,Thriller	4.5	382	0
4	The Elf Who Didn't Believe	2000	90	Comedy,Family	3.8	157	0
5	The Adventures of Rocky & Bull...	2000	92	Adventure,Comedy,Family	4.3	21656	0
6	Fanny Hill	2000	86	Comedy,Drama,Romance	4.0	201	0
7	Knockout	2000	99	Action,Drama	4.1	229	0
8	Exposé	2000	92	Drama,Mystery	4.1	224	0
9	Nutty Professor II: The Klumps	2000	106	Comedy,Romance,Sci-Fi	4.5	54750	0
10	Sinbad: Beyond the Veil of Mists	2000	85	Action,Adventure,Animation	4.4	742	0
11	Shafted!	2000	90	Action,Comedy	4.2	136	0

This structured dataset is used for model training and evaluation in downstream stages of the project.

Predicting IMDb Movie Ratings and Real-World Success Using Machine Learning

The goal of this milestone is to analyze and predict movie performance using machine learning models. We implemented two distinct approaches:

- Regression model to predict IMDb ratings.
- Classification-style regression to predict movie success using a weighted scores

Method 1: *Predicting IMDb Rating (Regression – Random Forest)*

This method consists of two parts:

- Part 1 trains the model using the average IMDb rating as the target variable.
- Part 2 trains the model using a custom-defined weighted success score (weighted-rate).

Although the target variable differs, the machine learning workflow remains the same for both parts:

- Genre Transformation: Convert the genres column into lists and apply MultiLabelBinarizer to one-hot encode each genre.
- Feature Engineering:
    - log_votes: Log-transformed number of votes to reduce skewness.

    - release_decade: Extracted from start_year to capture temporal trends.
    - runtime_scaled: Standardized runtime using StandardScaler.
- Train–Test Split: Applied train_test_split() with test_size=0.2 and random_state=42.
- Model Type: Implemented RandomForestRegressor for both parts.
- Feature Importance Analysis: Used the model's feature_importances_ attribute to identify and plot the top 10–20 most influential features.

We used the Random Forest Regressor to predict both the average IMDb rating and the weighted success score based on:

- Scaled runtime
- Log-transformed number of votes
- Release decade
- One-hot encoded genres

Part 1: Random Forest Regressor using average rating

Although the dataset had completed initial cleaning, additional preprocessing was necessary during model training to ensure alignment between the selected features and the target variable. As a result, the dataset was reduced from over 2 million records to approximately 1.6 million clean samples. This reduction of about 358,830 records—roughly 17.9%—was primarily due to missing values in critical modeling columns. Despite this drop, the remaining data is sufficiently large and of high quality, making it suitable for training a robust and reliable model

To predict the average rating score, we applied a Random Forest Regression model using IMDb's average_rating as the target variable (y). The feature set (X) included runtime_scaled, log_votes, release_decade, and one-hot encoded genre columns generated from mlb.classes_ (e.g., Action, Comedy, etc.).

Results:

- MAE: 0.5727
- RMSE: 0.9816
- R^2: 0.5111

Interpretation:

- On average, our model is within ~0.57 stars of the actual rating, which is reasonable for a 1–10 scale.
- R^2 indicates the model explains ~51% of the variance in ratings, suggesting room for improvement but capturing meaningful patterns.

However, during the modeling process, we recognized that producers should not rely solely on IMDb ratings to assess a movie's success, as high ratings do not always equate to real-world impact. True success also depends on audience reach like how many people engage with the film.

To address this, we introduced a weighted rating that combines both average rating and vote volume. This allows decision-makers to better distinguish between:

- Commercial blockbusters – films with moderate ratings but massive viewership
- Critically acclaimed niche films – films with high ratings but limited reach

Building on this insight, we developed a second modeling approach focused on a more comprehensive success metric that accounts for both critical reception and audience reach.

Part 2: Random Forest Regressor using weighted rated

To predict movie success more holistically, we applied a Random Forest Regression model using a custom-defined weighted success score as the target variable. This score, calculated as average_rating × log10(num_votes), accounts for both critical reception and audience reach, allowing us to move beyond

relying solely on IMDb rating. The feature set (*X*) included runtime_scaled, log_votes, release_decade, and one-hot encoded genre columns from mlb.classes_.

Results:

- MAE: 0.9693
- MSE: 2.7919
- RMSE: 1.6709
- R^2: 0.9428

Interpretation:

The model achieves an R^2 of 0.9428, suggesting it explains over 94% of the variance in the weighted score. However, with an MAE of 0.97 on a 1–10 scale, the predictions still deviate by roughly 10% of the possible range on average, indicating that while overall fit is strong, individual predictions may be off by about one point.

Comparison of Part 1 and Part 2

Metric	Part 1 Average Rating	Part 2 Weighted Success Score
Target (y)	average_rating	average_rating × log10(num_votes)
MAE	0.5727	0.9693
RMSE	0.9816	1.6709
R^2	0.5111	0.9428

Part 1, which predicts the IMDb average rating, achieves a lower MAE (0.57) than Part 2 (0.97), indicating more precise point predictions on a 1–10 scale. However, its R^2 of 0.51 means the model captures only about half of the patterns or factors that influence IMDb ratings. In contrast, Part 2's weighted success score model achieves a much higher R^2 of 0.94, meaning it captures nearly all the variability in the target, but with a larger average prediction error.

This difference exists because the weighted score varies more widely than the average rating — it reflects not only how well a film is rated but also how many people have seen it. In practice, Part 1 is best for judging a film's overall quality, while Part 2 is better for evaluating its market success. Looking at both together helps decision-makers balance artistic value with commercial potential.

Method 2: Predicting Success (Classification –XGBoost)

Defines "Movie Success"

To classify movies as successful or not, our team used a Weighted Score:

success = 1 if (average_rating × log10(num_votes)) ≥ 25

A movie is labeled as successful if its weighted score is greater than or equal to 25 (success = 1), and unsuccessful otherwise (success = 0). This approach offers a more nuanced and data-driven alternative to fixed cutoffs such as "rating ≥ 7.0 and votes ≥ 100,000.

The Weighted Score balances two essential components of movie performance:

- Quality – captured by the IMDb average rating, reflecting both critical and audience sentiment.
- Reach – represented by the number of votes, indicating audience size and engagement.

By combining these factors, the metric avoids bias toward:

- Obscure cult favorites with high ratings but low visibility.
- Mediocre blockbusters with high exposure but weak reception.

Log-scaling the number of votes normalizes vote count differences between wide-release blockbusters and smaller films. Without log-scaling, large-scale releases could dominate purely due to raw popularity, overshadowing films with smaller but highly engaged audiences.

Another advantage of this approach is tunability — the success threshold (e.g., 20, 25, 30) can be adjusted to meet different business goals, such as identifying high-potential films early in development or tightening criteria for award consideration.

Example:

Frozen II was classified as successful despite a 6.8 rating due to high visibility (209,000+ votes), with a weighted score of 36.18. (6.8×log10(209,000) ≈36.18).

Since this exceeds the threshold of 25, the model classifies it as successful. This illustrates the strength of the weighted score: it recognizes films that achieve substantial market impact despite not having exceptionally high ratings.

XGBoost Model:

The XGBoost model was trained using a combination of core features and genre-based indicators. The core features included start_year, runtime_minutes, is_adult, average_rating, log_votes, genre_count, sequel_flag, franchise_flag, release_decade, release_season, release_window, and avg_director_success_rate. In addition, one-hot encoded genre variables were incorporated to capture the influence of specific film categories on success, including is_action, is_comedy, is_drama, is_romance, is_thriller, is_scifi, is_animation, and is_horror.

Results:

- Accuracy: 93%
- ROC-AUC: 0.9859
- F1-Score for Class 1 (Success): 0.91

Interpretation:

The model achieved an accuracy of 93%, meaning it correctly classified the majority of movies as either successful or unsuccessful. The high ROC-AUC score of 0.9859 indicates excellent ability to distinguish between the two classes across different classification thresholds, showing the model is highly discriminative. The F1-score of 0.91 for the “success” class reflects a strong balance between precision (avoiding false positives) and recall (capturing true successes), making the model reliable for identifying films likely to achieve significant market impact.

Key Insights

The classification model is ideal for clear yes/no decisions (e.g., greenlight or not), helping stakeholders quickly filter high-potential projects and avoid low-success titles. It works best when a definitive action is needed, such as approving production, allocating budgets, or selecting seasonal releases.

The regression model excels at delivering continuous performance estimates, enabling ranking, prioritization, and portfolio optimization. Studios can use predicted scores to compare scripts, balance genres, and maximize return within budget or release constraints.

A combined approach works best: use classification to screen out low-potential projects, then regression to prioritize and allocate resources among the shortlisted titles.

Two modeling workflows were tested:

- *Method 1 (Google Colab):* Manual feature engineering, BigQuery → Google Drive → Colab pipeline, with optional manual hyperparameter tuning.
- *Method 2 (Vertex AI):* Direct BigQuery integration, AutoML feature selection, and automated hyperparameter tuning for faster optimization.
-

Model	X (Features)	y (Target)
XGBoost	- runtime_minutes	success → Binary (0 = fail, 1 = success)
	- genre_count	
	- is_adult	
	- sequel_flag	
	- log_votes	
	- franchise_flag	
	- avg_director_success_rate	
	- One-hot encoded **genres** (genres_df.columns)	
	- One-hot encoded **release season** (release_season_df.columns)	

	- One-hot encoded **release decade** (release_decade_df.columns)	
	- One-hot encoded **release window** (release_window_df.columns)	
Random Forest – Average Rating	- runtime_scaled (standardized runtime)	average_rating → Continuous rating value
	- log_votes	
	- release_decade	
	- One-hot encoded **genres** (list(mlb.classes_))	
Random Forest – Weighted Score	- runtime_scaled	weighted_score → Continuous score = rating × log10(votes)
	- log_votes	
	- release_decade	
	- One-hot encoded **genres** (list(mlb.classes_))	

We intentionally started with a smaller set of X features to ensure each one had a clear business rationale and measurable impact on the prediction. Our goal was to add features incrementally based on their relevance to the problem rather than including every possible variable from the start.

This approach helps avoid overfitting, keeps the model interpretable for stakeholders, and ensures that each feature aligns with our research questions. Once the base model is validated, we plan to introduce additional features — for example, director track record, seasonal release, or franchise status — and measure whether they improve performance in a meaningful way.

Conclusion

From millions of IMDb rows to making decisions, we built an end-to-end pipeline that explores, predicts, and prioritizes. We unified five IMDb tables, ran EDA that surfaced what matters—moderate runtimes tend to rate higher, History/Documentary/Animation perform well, and strong directors plus well-known actors correlate with better outcomes—then trained two complementary models: a Random Forest to predict quality (MAE ≈ 0.57, $R^2 \approx 0.51$) and a weighted-score regressor to capture market impact where score = rating × log10(votes) ($R^2 \approx 0.94$, better for ranking despite larger errors). For go/no-go, an

XGBoost classifier delivers production-ready performance (accuracy ≈ 93%, ROC-AUC ≈ 0.986, F1 for "success" ≈ 0.91), so the workflow is simple: filter with classification, then rank the shortlist with the weighted score for funding and release strategy. Limits: we rely on IMDb ratings/votes and don't yet include budget, marketing, or distribution. Next steps: tune thresholds to business costs, calibrate probabilities, expand features (budget, campaign intensity, franchise maturity), and add explainability (e.g., SHAP) so creative and finance teams can trust and act on the signals. Net result: a repeatable system that cuts guesswork and improves ROI on content bets.

References

IMDb.com. (n.d.). *IMDb datasets*. IMDb. Retrieved August 14, 2025, from https://www.imdb.com/interfaces/

Google Cloud. (n.d.). *BigQuery public datasets*. Google Cloud. Retrieved August 14, 2025, from https://cloud.google.com/bigquery/public-data

Google Cloud. (n.d.). *Vertex AI*. Google Cloud. Retrieved August 14, 2025, from https://cloud.google.com/vertex-ai

Kaggle. (n.d.). *IMDb datasets and movie metadata*. Kaggle. Retrieved August 14, 2025, from https://www.kaggle.com

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785

McKinney, W. (2010). Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference* (pp. 51–56). SciPy. https://doi.org/10.25080/Majora-92bf1922-00a

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html

Van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering, 13*(2), 22–30. https://doi.org/10.1109/MCSE.2011.37