HarvardX PH125.9x — MovieLens Capstone

Fernando Marcelo Parodi

December 10, 2025

Contents

Executive Summary

This project successfully developed a predictive recommendation system for the MovieLens 10M dataset, achieving the stringent RMSE target of < 0.86490. The core approach was an additive effects model that progressively decomposed the overall rating variance into four empirical biases: Movie Appeal (b_i), User Tendency (b_u), Genre Preference (b_g), and Review Date Drift (b_t). The final model utilized Penalized Least Squares (Regularization), with the optimal strength ($lambda = 5.15$) tuned exclusively on the edx development set. This methodology ensured strict separation between the tuning process and the final, single evaluation on the held-out final_holdout_test set. The final model achieved an Official Validation RMSE of 0.86430.

Key decisions:

- Keep the model easy to read and progressive (effects for items, users, genres, and time).
- une lambda via a coarse-to-fine grid using an internal split of **edx** only.
- Avoid any use of **final_holdout_test** for training or tuning (strict separation).

Exploratory Data Analysis

In this analysis, **edx** (a data.frame) has 9,000,055 rows and 6 columns. The Cartesian product of users and movies would yield approximately 746 million possible ratings; the gap to the actual row count highlights substantial sparsity.

Data model

The MovieLens 10M dataset is composed of two source files:

- `ratings.dat`: `(userId, movieId, rating, timestamp)` — one row per user–movie rating event.
- `movies.dat`: `(movieId, title, genres)` — movie metadata; `genres` is a pipe-separated list.

After joining on `movieId`, the working table used in this analysis contains:

- `userId` (integer): unique user identifier.
- `movieId` (integer): unique movie identifier.
- `rating` (numeric): explicit rating in $\{0.5, 1.0, \ldots, 5.0\}$.
- `timestamp` (integer): Unix time for the rating event.
- `title` (character): movie title (often includes year in parentheses).
- `genres` (character): one or more genres separated by `|` (e.g., `"Drama|Romance"`).

Engineered variables created later for modeling/EDA include a weekly time key (`week`) and atomic genre labels (exploding `genres`).

Table 1: High-level dataset dimensions

metric	value
Rows (ratings)	9000055
Unique users	69878
Unique movies	10677

Ratings / Global mean

The overall average rating in the dataset (mu) was found to be approximately 3.51. Starting with this global mean as a baseline yielded an RMSE of 1.060. This high initial error immediately signals that predicting accurately requires accounting for systematic deviations from the mean, necessitating the inclusion of additional empirical effects.

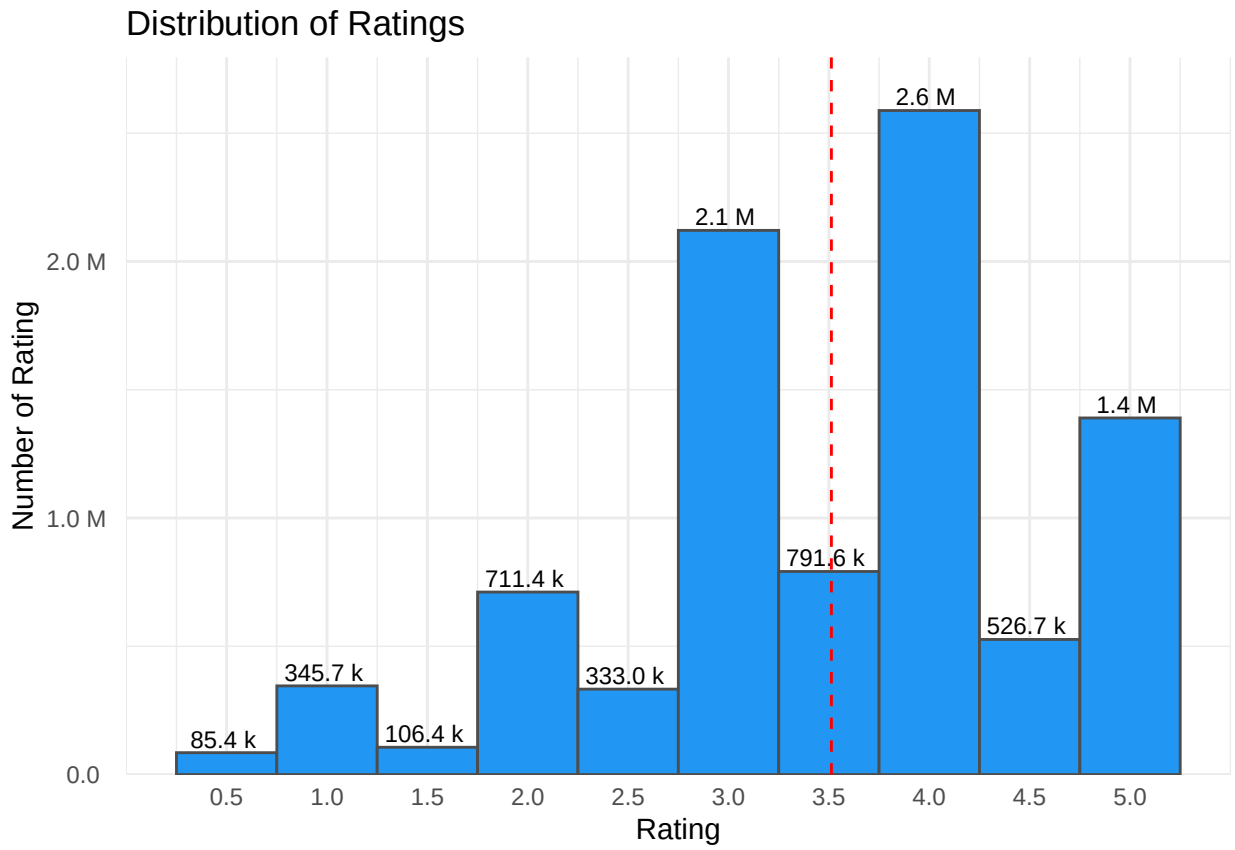

Figure 1: Distribution of Ratings

Movies ($movieId$) - Users ($userId$)

Ratings are uneven across titles: the most-rated film — Pulp Fiction (1994) — has 31,362 ratings, while 126 movies are rated only once; this motivates a movie-specific bias (`b_i`). User activity is similarly skewed: the most active user submitted 6,616 ratings, whereas 0 users recorded fewer than 10; a user bias (`b_u`) accounts for these tendencies.

Figure 2: Variation average rating per movie (b_i) / user (b_u)

Movie Genre ($genres)

Genre differences. Mean ratings vary generally across genre combinations.
Among groups with at least 50,000 ratings (31 combinations), the highest mean is 4.00 for Comedy|Crime|Drama, and the lowest is 2.88 for Horror.
The 1.12-point spread (visible with the 95% CIs in the figure) supports adding a genre bias term `b_g` to the model.

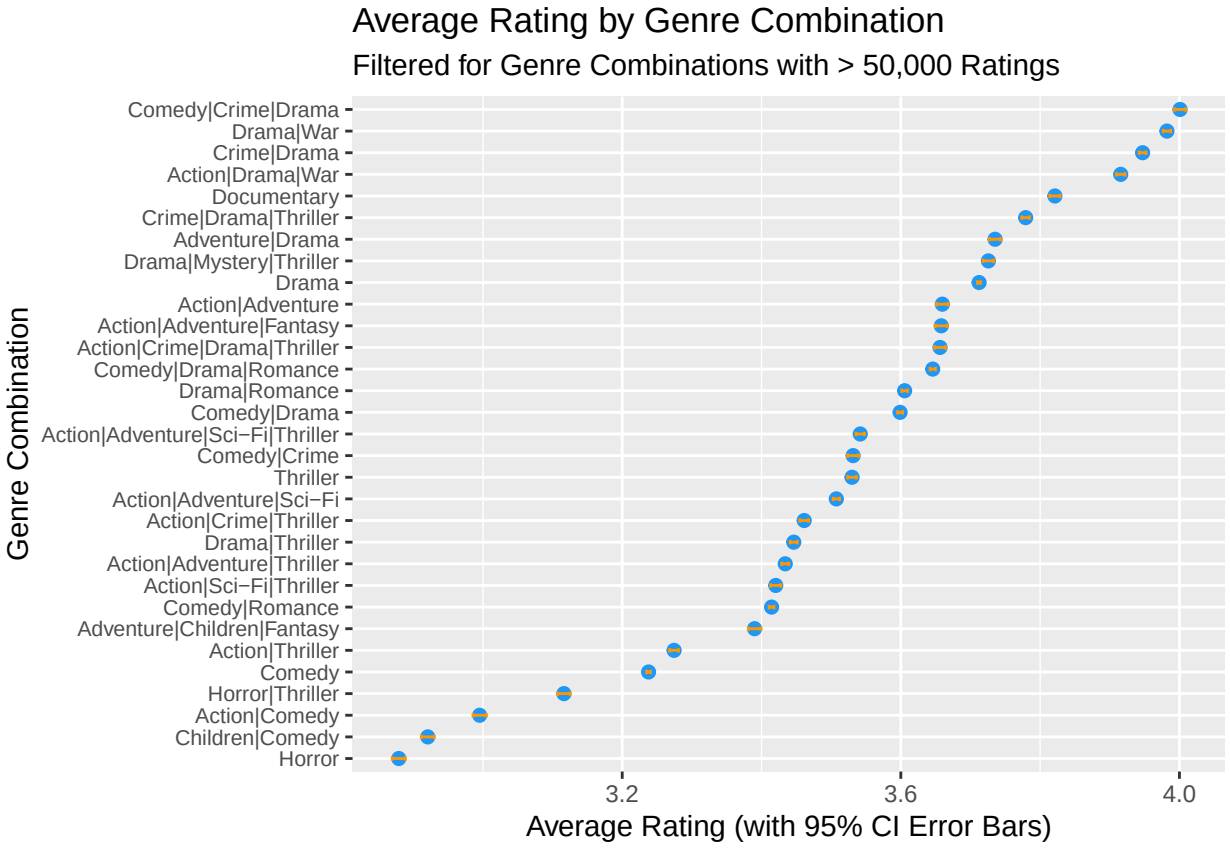

Figure 3: Average Rating by Genre Combination

Date of review ($timestamp)

Ratings begin in 1995. The yearly mean drops from 4.00 to a local low around 2004 (3.43), then recovers gradually through 2009 (see Figure 4). The temporal pattern is modest compared with the movie (`b_i`) and user (`b_u`) effects, but it is present, so a time bias term (`b_t`) is included in the model.

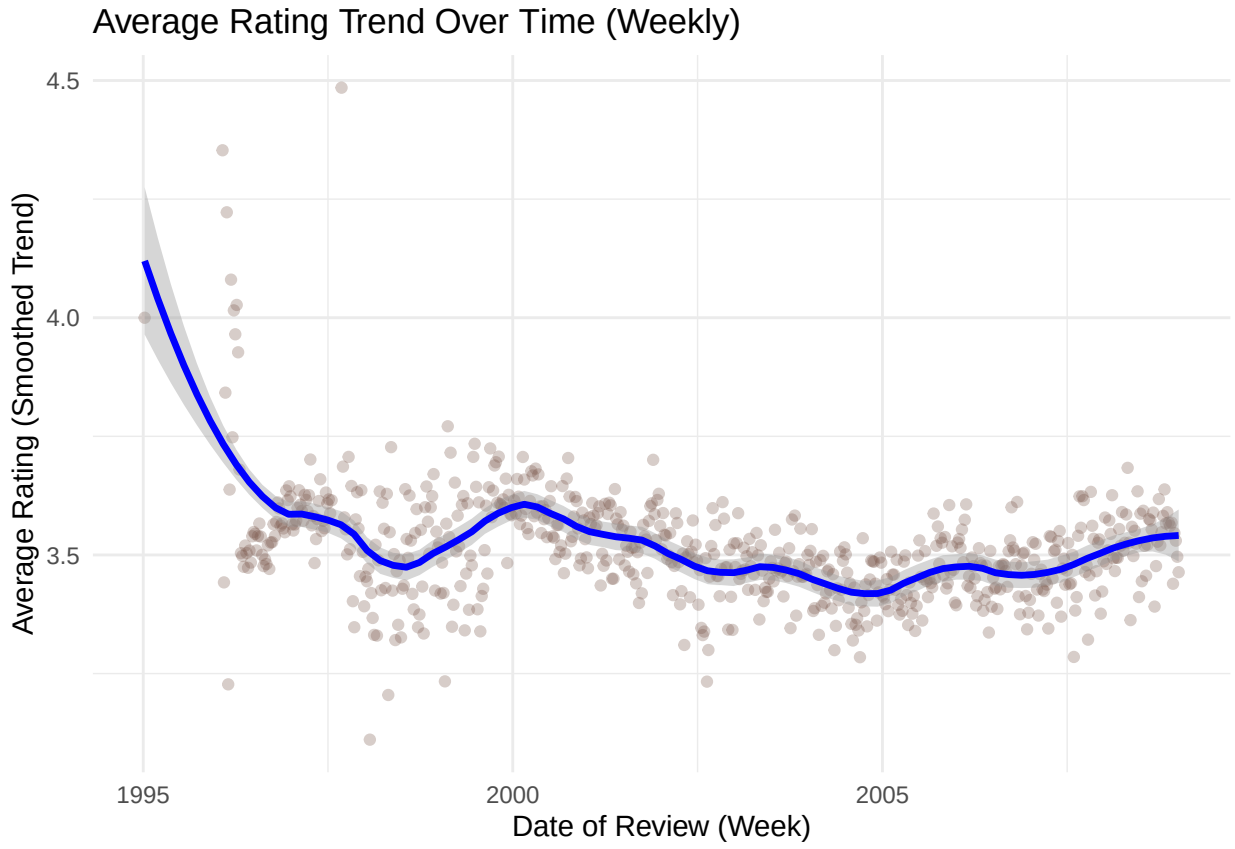

Figure 4: Average Rating Over Time (weekly) (b_t)

Methods and Modeling

Splitting edx out into train and test sets

This crucial first step established the internal tuning environment. The edx set was split into a 90% training partition (train_set) and a 10% test partition (test_set). The use of semi_join was a strict measure to guarantee that every movie and user present in the test_set had at least one observation in the train_set, thereby avoiding the "Cold Start" problem during development and ensuring meaningful RMSE calculations

Developing the Algorithm (Progressive Bias)

The algorithm was built progressively by introducing one additive effect at a time. This step-by-step approach not only simplifies the model but also empirically measures the contribution of each factor to the total error reduction. The most significant performance jump was observed upon the inclusion of the b_u (User) term, confirming that individual rating habit is the dominant source of residual variation after accounting for the inherent quality of the film (b_i).

Regularising the algorithm

Regularization was the critical final step required to enhance the model's generalizability and prevent overfitting. By applying a penalty$\hat{b} = \frac{\sum \text{residual}}{n+\lambda}$ to the magnitude of the effect estimates, regularization shrinks estimates that are based on small sample sizes (noisy data) toward the population mean. This is crucial for improving out-of-sample prediction. The optimal *lambda* was determined via cross-validation, where the model was tested across a sequence of lambda values to find the one that minimized the RMSE on the internal test_set. The value **lambda = 5.15** was identified as the optimal balance point between bias and variance.

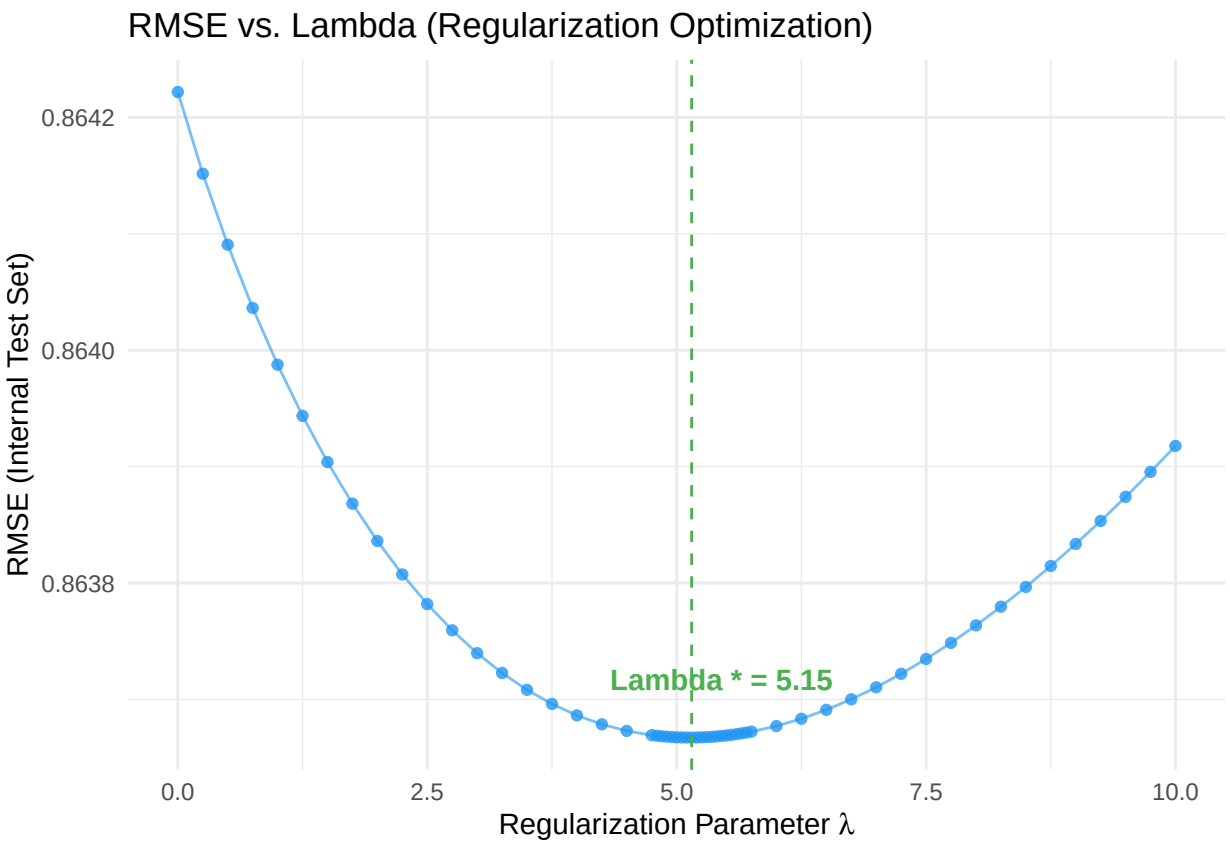

Figure 5: Selecting the Optimal Tuning Parameter (Lambda) via Coarse and Fine Grid Search

Results

The internal split inside **edx** shows monotonic improvement as effects are added, followed by a further reduction in RMSE after regularization. Finally, the model trained on the full **edx** with the tuned `lambda*` is evaluated once on the **final_holdout_test**.

Table 2: Official RMSE on final_holdout_test.

method	RMSE
Regularized Movie + User + Genre + Time Effect Model	0.86367
Movie + User + Genre + Time Effect Model	0.86422
Official (final_holdout_test)	0.86430
Movie + User + Genre Effect Model	0.86432
Movie + User Effect Model	0.86468
Movie Effect Model	0.94296
Just the Average	1.06005

Method	RMSE	final vs objective
Project objective	0.86490	-
Validation of latest model	0.86430	-0.00060

Conclusion

- The incremental effects address major sources of bias: movie popularity (`b_i`) and user strictness (`b_u`).
- Regularization reduces variance for movies and users with few ratings, improving generalization.
- Genre and time effects provide small but consistent gains by capturing structural patterns in the data.
- The strict separation between tuning and final evaluation prevents leakage and aligns with the project instructions.

Limitations and Future Work

The primary constraint of this final model is its linearity. By design, the additive effects model does not capture complex, non-linear user-item interactions (e.g., a specific user who hates only sci-fi directed by filmmaker X). This results in a residual error that is not truly independent. Future work should focus on Matrix Factorization techniques (e.g., Funk SVD). This approach would decompose the residual error into latent features (unseen characteristics of movies and users) and is significantly more scalable and memory-efficient for real-world production environments than the linear bias model presented here.

Reproducibility Notes

- **Code:** The R code used in this project is fully reproducible and contained in the attached Script.R file.
- **Translation:** All English translations within this report were conducted using the Gemini Large Language Model (Google, 2024).

```
## R version 4.5.1 (2025-06-13 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 11 x64 (build 26200)
##
## Matrix products: default
##   LAPACK version 3.12.1
##
## Random number generation:
##  RNG:     Mersenne-Twister
##  Normal:  Inversion
##  Sample:  Rounding
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## time zone: America/Buenos_Aires
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] knitr_1.50        gridExtra_2.3     scales_1.4.0      data.table_1.17.8
##  [5] caret_7.0-1       lattice_0.22-7    lubridate_1.9.4   forcats_1.0.0
##  [9] stringr_1.5.2     dplyr_1.1.4       purrr_1.1.0       readr_2.1.5
## [13] tidyr_1.3.1       tibble_3.3.0      ggplot2_4.0.0     tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.6         xfun_0.53            recipes_1.3.1
##  [4] tzdb_0.5.0           vctrs_0.6.5          tools_4.5.1
##  [7] generics_0.1.4       stats4_4.5.1         parallel_4.5.1
## [10] ModelMetrics_1.2.2.2 pkgconfig_2.0.3      Matrix_1.7-4
## [13] RColorBrewer_1.1-3   S7_0.2.0             lifecycle_1.0.4
## [16] compiler_4.5.1       farver_2.1.2         codetools_0.2-20
## [19] htmltools_0.5.8.1    class_7.3-23         yaml_2.3.10
## [22] prodlim_2025.04.28   crayon_1.5.3         pillar_1.11.1
## [25] MASS_7.3-65          gower_1.0.2          iterators_1.0.14
## [28] rpart_4.1.24         foreach_1.5.2        nlme_3.1-168
## [31] parallelly_1.45.1    lava_1.8.1           tidyselect_1.2.1
## [34] digest_0.6.37        stringi_1.8.7        future_1.67.0
## [37] reshape2_1.4.4       listenv_0.9.1        labeling_0.4.3
## [40] splines_4.5.1        fastmap_1.2.0        grid_4.5.1
## [43] cli_3.6.5            magrittr_2.0.4       utf8_1.2.6
## [46] survival_3.8-3       future.apply_1.20.0  withr_3.0.2
```

```
## [49] bit64_4.6.0-1        timechange_0.3.0     rmarkdown_2.29
## [52] globals_0.18.0       bit_4.6.0            nnet_7.3-20
## [55] timeDate_4041.110    hms_1.1.3            evaluate_1.0.5
## [58] hardhat_1.4.2        mgcv_1.9-3           rlang_1.1.6
## [61] Rcpp_1.1.0           glue_1.8.0           pROC_1.19.0.1
## [64] ipred_0.9-15         vroom_1.6.6          rstudioapi_0.17.1
## [67] R6_2.6.1             plyr_1.8.9
```