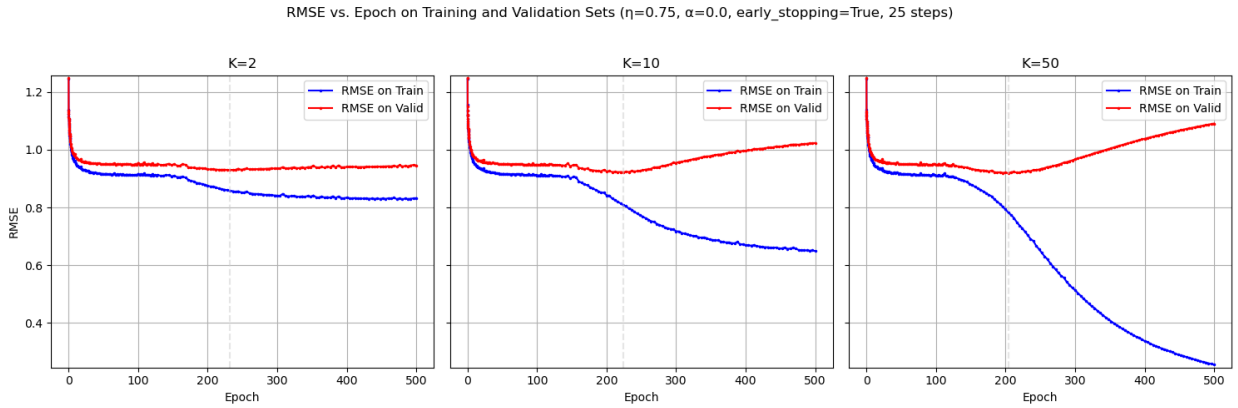
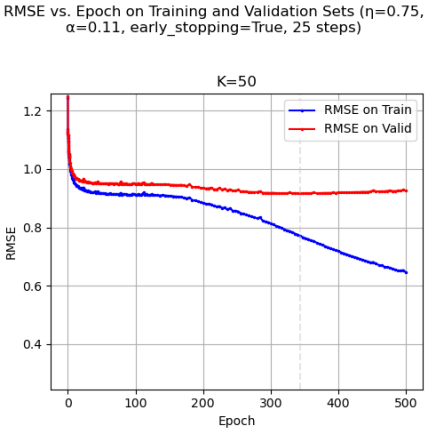
Problem 1

1a



(i) As K increases so does the rate at which overfitting occurs. This is because increases in K drive model complexity, causing it to overfit & poorly generalize. (ii) The 'best' validation-set performance, in terms of RMSE, improved from 0.930 (K = 2) to 0.922 (K = 10) to 0.920 (K = 50). However, there are diminishing improvements in RMSE as K increases. (iii) The models were trained on a step size of η = 0.75. This value was selected because it lies at the upper range of step size values that still consistently converge, thus reducing train time without sacrificing accuracy.

1b



(i) α = 0.11. This alpha value was determined through a three-round grid search that refined the interval range (e.g. 1.0E-3 → 1.0E1) to converge on the best alpha value based on validation RMSE. (ii) η = 0.75. The step size was retained from section **1a** to control for the other hyperparameters and because it still represented a high and stable learning rate. (iii) Yes, this training run, with a non-zero alpha, returned a better RMSE - 0.916 (K = 50, α = 0.11) - compared to that in **1a** - 0.920 (K = 50, α = 0.0). This is because the alpha value curbed variance in the model, regularizing latent factors and improving generalization.

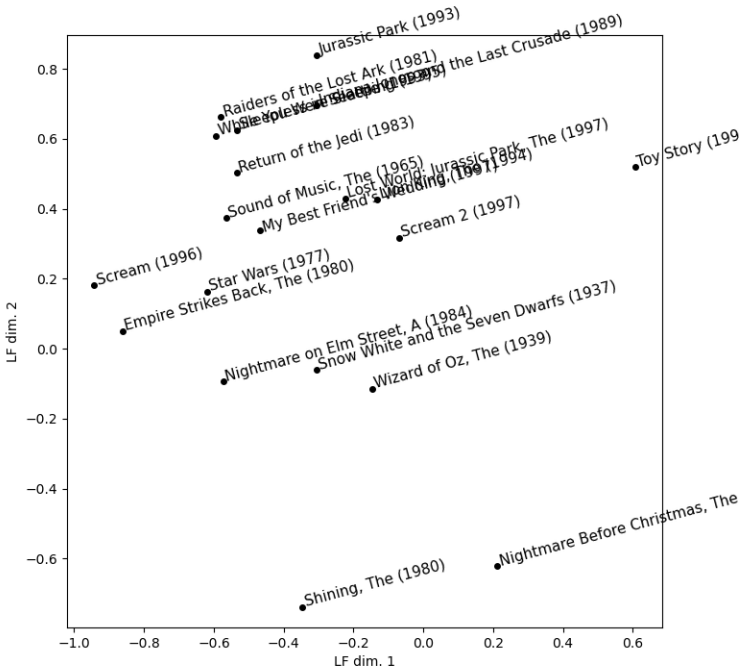
1c

Model	Train (RMSE, MAE)	Valid (RMSE, MAE)	Test (RMSE, MAE)
M_1 (K = 02, α = 0.00)	0.858, 0.675	0.930, 0.732	0.931, 0.727
M_2 (K = 10, α = 0.00)	0.807, 0.632	0.922, 0.724	0.922, 0.718
M_3 (K = 50, α = 0.00)	0.781, 0.611	0.920, 0.723	0.917, 0.715
M_4 (K = 50, α = 0.11)	0.769, 0.606	0.916, 0.723	0.912, 0.713

(i) Based only on RMSE, the optimal number of latent factors is $K = 50$ (M_4), which achieves the lowest RMSE on train, validation, and test sets. However, the improvement from $K = 10$ (M_2) to $K = 50$ (M_4) is marginal across sets (0.65% RSME & 0.14% MAE gains on the validation set and 1.08% RSME & 0.70% MAE gains on the test set) yet quintuples the latent-factor count, increasing computational resources required to train the model and the risk of overfitting. Thus, for practical purposes, $K = 10$ is the optimal number of latent factors. (ii) When ranking by MAE, the validation set ties M_4 and M_3 for first (0.723), then M_2 (0.724) and M_1 (0.732), while the test-set MAE ordering exactly matches the RMSE ranking ($M_4 > M_3 > M_2 > M_1$).

1d

While there are some clusters of movies that do bunch in an interpretable way - scary movies like the *The Shining (1980)* and *The Nightmare Before Christmas (1993)*, early-era fantasy films like *The Wizard of Oz (1939) and Snow White and the Seven Dwarfs (1937)*, the majority of the vector embeddings do not make sense based on their genres or release dates. For example, *The Empire Strikes Back (1980)* and *Star Wars (1977)* are both closer to *Scream (1996)* than to their own sequel, *Return of the Jedi (1983)*. This is likely because the LF model is driven only by the co-rating patterns and does not consider user or item metadata (e.g. age or release date). Thus, it cannot capture *why* some films may 'belong' together.



Problem 2

2a

The final **Problem 2** solution, $M_{E\text{-}SVD}$ adopts an ensemble of 27 Surprise SVD++ models, each perturbed ±10% around the optimal hyperparameters, with predictions averaged[1]. A three-stage hyperparameter search was executed against *n_factors*: the number of latent factors, *lr_all*: the learning rate for all parameters, *reg_all*: the regularization rate for all parameters, and *n_epochs*: the number of epochs in the model. Each stage executed a grid search over a seven-point grid with values lin- or log-spaced around the current best, progressively narrowing the interval range on the optimal hyperparameter. All grid searches used 5-fold cross-validation and fixed seeds.

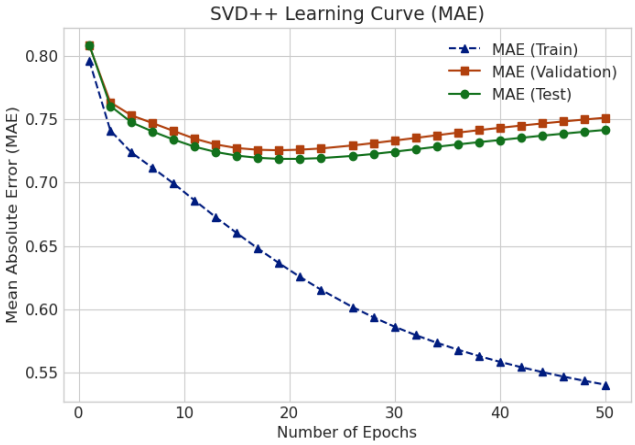
This approach leverages some important characteristics about Latent Factor models and Ensembling. Primarily, SVD (i.e. latent vector matrix factorization) was chosen because it scales well on sparse data, balances bias-variance well, and easily translates to the 1-5 rating scale[1,2]. Further, Surprise's SVD++ algorithm improved over SVD

[1] Surprise. "Matrix Factorization-Based Algorithms."
[2] A., Elie. "Deep Dive into Matrix Factorization for Recommender Systems: From Basics to Implementation."

because it also captures implicit ratings (i.e. the fact that a user *i* rated some item *j*), and thus accounts for the set of rated items and so lowers variance in the learned factors (Equation 1)[1,3,4].

Lastly, introducing the $\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right)$ (1) 27-part ensemble based on hyperparameter perturbations reduces model variance while preserving the low bias of matrix factorization, improving generalizability[5].

2b



The figure illustrates the SVD++ learning curve (MAE vs. Epochs), which was used to determine the optimal training duration. As epochs increase, the model fits onto more of the underlying patterns in the data. Thus, training and validation MAE decrease, with validation error reaching its minimum at *n_epochs* = 19. At this point the model begins to overfit on the dataset and it no longer generalizes well. Thus, *n_epochs* = 19 was selected as the optimal stopping point because it has the lowest held-out error with decent generalization performance.

2c

Model	Test MAE	Leaderboard MAE
M_2 ($K = 10$, $\alpha = 0.00$)	0.718	—
M_4 ($K = 50$, $\alpha = 0.11$)	0.712	—
$M_{E\text{-}SVD}$ ($K = 9 \pm 10\%$, *lr_all* = 0.00668 ± 10%, *reg_all* = 0.00499 ± 10%, *n_epochs* = 19, *n_trees* = 27)	0.630	0.705

(i) The final ensemble model, $M_{E\text{-}SVD}$, achieves a test MAE of 0.630 and a leaderboard MAE of 0.705. The slight increase suggests reasonable generalization (standard deviation = 0.053) but with room for improvement as the gap means overfitting. The differences in MAE are likely the result of changes to the distribution of ratings (i.e. sparsity or cold-start items). (ii) Compared to the models in **Problem 1**, the **Problem 2** solution is far better. The

[3] Li, Gai, and Qiang Chen. "Exploiting Explicit and Implicit Feedback for Personalized Ranking."
[4] Chen, Shulong, et al. "Matrix Factorization for Recommendation with Explicit and Implicit Feedback."
[5] Siddhartha Pramanik. "Handling Sparse and High-Dimensional Data in Machine Learning: Strategies and Techniques for Improved Model Performance."

combination of a more extensive hyperparameter search, ensembling, and the SVD++ algorithm reduce variance and better manage the bias-variance tradeoff.

2d

Overall, the ensemble SVD++ approach has strong predictive accuracy because it compresses high-dimensional user–item interactions into compact latent embeddings while also incorporating implicit feedback signals. It scales efficiently to highly-sparse, large datasets like MovieLens 100K and ensembling helps reduce variance across hyperparameter runs. However, combining multiple SVD++ models (*n_trees* = 27) requires substantial computational resources, making it unsuitable for real-time recommendations, and the current pipeline ignores temporal and demographic dynamics. (See **Appendix A** and **Appendix B**, together they demonstrate that the model fails to capture temporal bias within the movies.)

If I had another week, I would spend more time exploring factorization methods, like LightFM, in order to weave in user and item metadata alongside collaborative signals. I'd also try stacking or ensembling in different ways. I attempted to build a logistic regression on top of SVD++ and LightFM outputs to map continuous scores back to the 1–5 rating scale, but struggled to implement that pipeline. With more time I believe this pipeline could result in better accuracy.

A major lesson I took away is how crucial it is to understand your data before you start modeling. Profiling rating distributions by user activity, item popularity, and decade buckets flagged all the biases that the latent-factor models struggled with and would've otherwise swept under the rug. As a result, I know what next steps I should take to improve the output. I also recognized that logging every single experiment - whether that be notes on metrics, hyperparameters, or decisions made - makes writing the report and retracing my steps so much smoother.

Acknowledgements

Use of Artificial Intelligence (AI)

AI was used in this project to do the following:

- Display information about the MovieLens 100K dataset (population min/max, mean, median, top 5, etc.)
- Graph information about the dataset, **Problem 1** RMSE vs. Epoch trends, and **Problem 2** learning curves and movie rating distributions by decade.

Bibliography

A., Elie. “Deep Dive into Matrix Factorization for Recommender Systems: From Basics to Implementation.” *Medium*, 31 Aug. 2024, medium.com/%40eliasah/deep-dive-into-matrix-factorization-for-recommender-systems-from-basics-to-implementation-79e4f1ea1660. Accessed 28 Apr. 2025.

Chen, Shulong, and Yuxing Peng. “Matrix Factorization for Recommendation with Explicit and Implicit Feedback.” *Knowledge-Based Systems*, vol. 158, Oct. 2018, pp. 109–117, https://doi.org/10.1016/j.knosys.2018.05.040. Accessed 28 Apr. 2025.

Funk, Simon. “Netflix Update: Try This at Home.” *Sifter.org*, 11 Dec. 2006, sifter.org/simon/journal/20061211.html. Accessed 28 Apr. 2025.

Ghai, Bhavya, et al. “Multi-Level Ensemble Learning Based Recommender System.” https://www3.cs.stonybrook.edu/~bghai/docs/project/Ensemble.pdf. Accessed 28 Apr. 2025.

Jain, Aarshay. “Quick Guide to Build a Recommendation Engine in Python & R.” *Analytics Vidhya*, June 2016, www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python/. Accessed 28 Apr. 2025.

Klimashevskaia, Anastasiia, et al. “A Survey on Popularity Bias in Recommender Systems.” *User Modeling and User-Adapted Interaction*, 1 July 2024, https://doi.org/10.1007/s11257-024-09406-0. Accessed 28 Apr. 2025.

Kula, Maciej. “Quickstart — LightFM 1.16 Documentation.” *Lyst.com*, 2016, making.lyst.com/lightfm/docs/quickstart.html. Accessed 28 Apr. 2025.

Li, Gai, and Qiang Chen. “Exploiting Explicit and Implicit Feedback for Personalized Ranking.” *Mathematical Problems in Engineering*, vol. 2016, 2016, pp. 1–11, https://doi.org/10.1155/2016/2535329.

“NumPy V2.2 Manual — Numpy.einsum .” *Numpy.org*, numpy.org/doc/stable/reference/generated/numpy.einsum.html. Accessed 28 Apr. 2025.

Ohtsuki, Tomoki. “A Basic Tutorial with Movielens 100K — MyFM 0.2.1 Documentation.” *Readthedocs.io*, 2023, myfm.readthedocs.io/en/stable/movielens.html. Accessed 28 Apr. 2025.

Siddhartha Pramanik. “Handling Sparse and High-Dimensional Data in Machine Learning: Strategies and Techniques for Improved Model Performance.” *Medium*, 24 Jan. 2025, medium.com/@siddharthapramanik771/handling-sparse-and-high-dimensional-data-in-machine-learning-strategies-and-techniques-for-34516e88efff. Accessed 28 Apr. 2025.

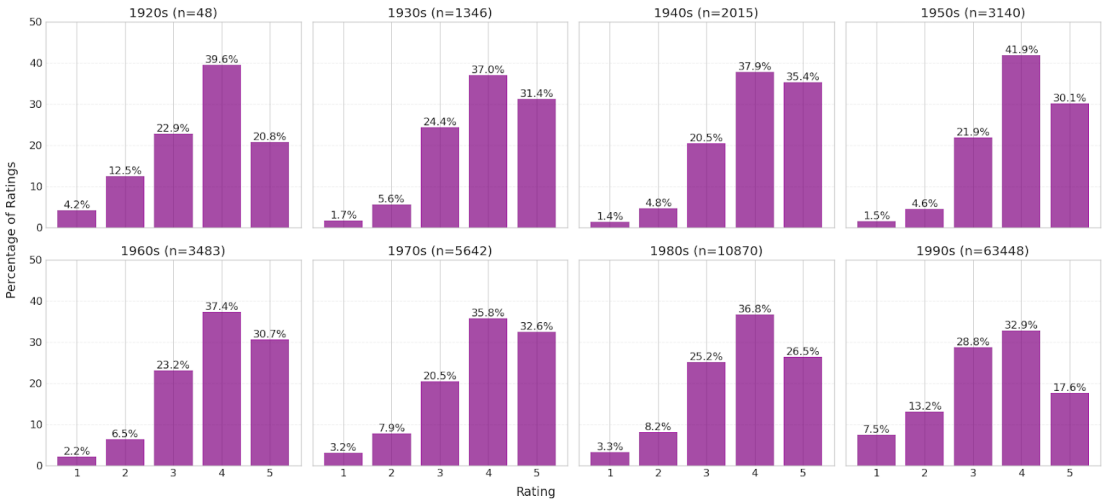
Surprise. “Matrix Factorization-Based Algorithms.” *Surprise.readthedocs.io*, 2015, surprise.readthedocs.io/en/stable/matrix_factorization.html. Accessed 28 Apr. 2025.

tufts-ml-courses. “Day20-RecommenderSystems.ipynb.” *GitHub*, 2025, github.com/tufts-ml-courses/cs135-25s-assignments/blob/main/labs/day20-RecommenderSystems.ipynb. Accessed 28 Apr. 2025.

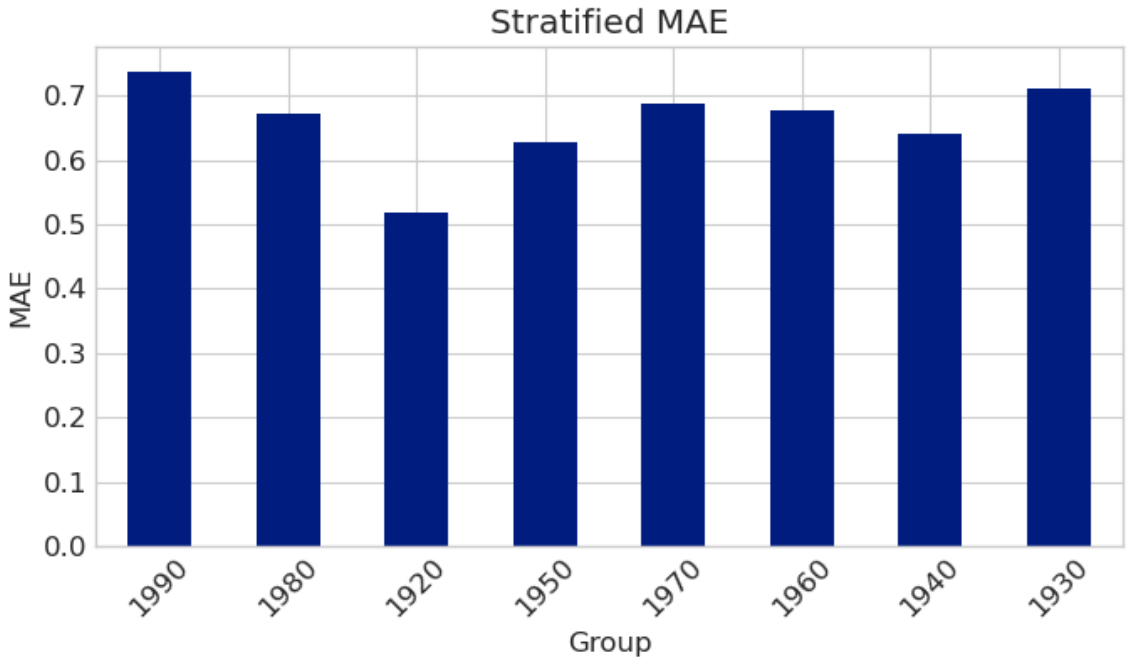
Wegmeth, Lukas. "The Impact of Feature Quantity on Recommendation Algorithm Performance: A Movielens-100K Case Study." *ArXiv.org*, 13 July 2022, arxiv.org/abs/2207.08713. Accessed 28 Apr. 2025.

Appendix

Appendix A. MovieLens Rating Distribution by Decade (1920s–1990s).



Appendix B. Validation MAE of Ensemble SVD++ by Movie Release Decade (1920s-1990s).*



*Not in chronological order.