CS 155 MP2

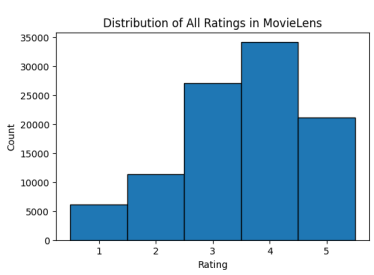
Jafarbek Arifdjanov

March 2025

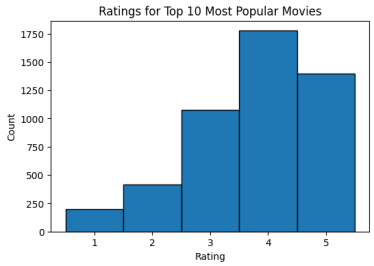
1 Introduction

Team Name: Arifdjanov Jafarbek.
Team members: Arifdjanov Jafarbek.
Work Division: All of it was done by Jafar.
Packages: Numpy, Pandas, Matplotlib, Random, Surprise.

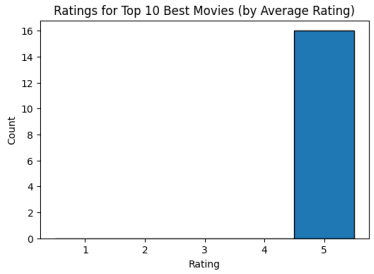
2 Basic Visualization



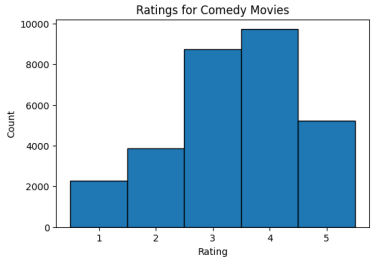
(a) Histogram 1



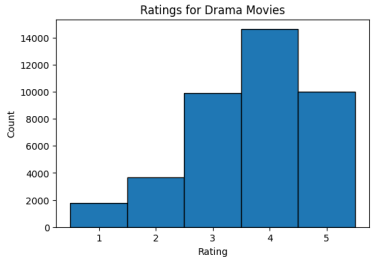
(b) Histogram 2



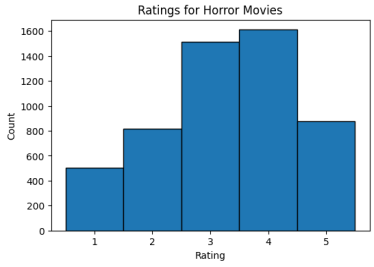
(c) Histogram 3



(d) Histogram 4



(e) Histogram 5



(f) Histogram 6

After plotting the distribution of all user ratings in MovieLens, I noticed that most scores clustered around 3 and 4, with fewer extremely low (1) or extremely high (5) ratings, although 4 clearly stood out as the most common. Focusing on the Top-10 Most Popular Movies, the histogram revealed a stronger concentration of 4- and 5-star ratings, suggesting that widely watched titles tend to meet or exceed viewers' positive expectations. The Top-10 Best Movies, by contrast, were almost exclusively 5-star, which makes sense for movies that truly excel in user consensus. Examining the three genres—Comedy, Drama, and Horror—showed generally similar shapes but with meaningful differences. Comedy spanned mostly mid-to-high ratings (3–5), with some lower scores likely due to differing comedic tastes; Drama showed a strong peak around 4, often drawing higher engagement and emotional investment from viewers; and Horror proved the most polarized, attracting both loyal fans who rate it highly and others who strongly dislike the genre,

creating more frequent 1- and 2-star ratings. Overall, these observations confirm a slightly positive bias in user ratings, a heavy skew toward 4s in popular titles, and distinct patterns across different genres, with Horror standing out as especially "love it or hate it."

https://colab.research.google.com/drive/1D6Oh76UIaZ67ERD-S4xNrdEWk$_D$5 − $eOH?usp = sharing$

3 Matrix Factorization Visualizations

3.1 SVD

In my first method, I used a basic (unbiased) SVD model to approximate the user–movie rating matrix. Suppose there are M users, N movies, and each known rating is denoted Y_{um}. I represent user u by a latent vector $\mathbf{U}_u \in R^K$ and movie m by $\mathbf{V}_m \in R^K$. The predicted rating $\widehat{Y}_{um}$ is then given by

$$\widehat{Y}_{um} = \mathbf{U}_u \cdot \mathbf{V}_m \;=\; \sum_{k=1}^{K} U_{u,k}\, V_{m,k},$$

where K is the chosen rank (e.g., 20). I only observe a subset of these ratings in the dataset, so I define a cost function over the observed entries. A common choice is

$$\mathcal{L}(\mathbf{U},\mathbf{V}) \;=\; \frac{1}{2} \sum_{(u,m)\in Obs} \big(Y_{um} - \mathbf{U}_u \cdot \mathbf{V}_m\big)^2 \;+\; \frac{\lambda}{2}\big(\|\mathbf{U}\|^2 + \|\mathbf{V}\|^2\big),$$

where λ is a regularization parameter controlling overfitting, and $\|\mathbf{U}\|^2$ (resp. $\|\mathbf{V}\|^2$) is the sum of squares of all entries in $\mathbf{U}$ (resp. $\mathbf{V}$). I then minimize $\mathcal{L}$ by gradient descent, iteratively updating each user–movie pair (u, m):

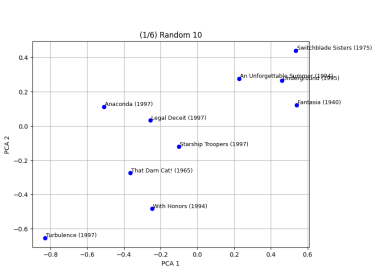
$$U_{u,k} \leftarrow U_{u,k} - \eta\Big(-\big(Y_{um} - \mathbf{U}_u \cdot \mathbf{V}_m\big)\, V_{m,k} + \lambda\, U_{u,k}\Big), \quad V_{m,k} \leftarrow V_{m,k} - \eta\Big(-\big(Y_{um} - \mathbf{U}_u \cdot \mathbf{V}_m\big)\, U_{u,k} + \lambda\, V_{m,k}\Big),$$

where η is the learning rate.

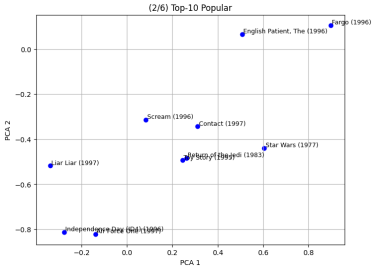
After tuning η and λ (e.g., via a grid search), I train on all known ratings until the error no longer significantly decreases or I reach a maximum epoch count. In my experiments, I found a suitable learning rate and regularization that produced a final training error around a certain threshold.

Analysis of the Resulting Graphs

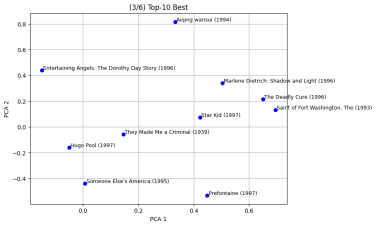
Once the model converged, I extracted the item–factor matrix $\mathbf{V}$, which is $N \times K$. I then performed PCA from K dimensions down to 2, allowing me to visualize each movie as a point in a 2D plane. Figures (a)–(f) show six different subsets of these movies:



(a) Graph 1

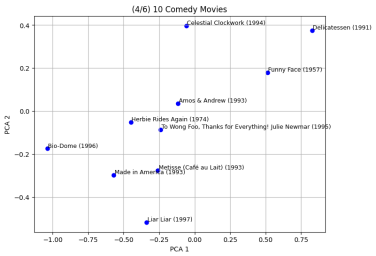


(b) Graph 2

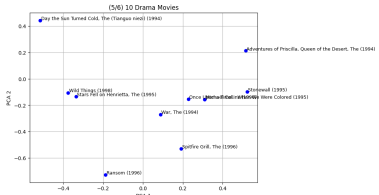


(c) Histogram 3

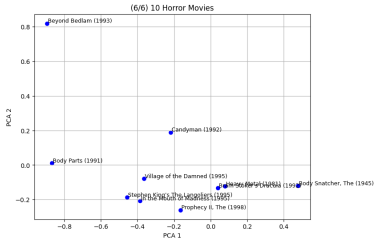
In the first three, I plotted *(i) Any 10 Movies*, *(ii) Top–10 Most Popular*, and *(iii) Top–10 Best (by average rating)*. Most Popular titles appear strongly clustered in a certain region, reflecting how many users rated them consistently at the higher end, while the Top–10 Best set displays near-perfect feedback, often clustering very close in the 2D space.



(a) Graph 4



(b) Graph 5



(c) Graph 6

For the next three plots, I focused on specific genres: *(iv) 10 Comedy Movies*, *(v) 10 Drama Movies*, and *(vi) 10 Horror Movies*. Comedies tended to group moderately close, possibly indicating a common user base that enjoys comedic films, while Dramas seemed to have a higher internal variance (some are near Comedies, others stand apart), which aligns with their broad thematic range. Horror titles exhibited a more polarizing pattern, with some clustering toward what might be "genre fans" and others lying farther away, suggesting strong differences in user reception. These scatterplots confirm that even a basic SVD model can discover meaningful structure in the rating data. Movies with similar rating patterns end up nearby in latent space. Although this approach does not include bias terms, it still captures broad trends and similarities among users and items. In the next sections, I enhance the model with bias terms and compare results to an off-the-shelf library.

https://colab.research.google.com/drive/1D6Oh76UIaZ67ERD-S4xNrdEWk$_D$5 − $eOH?usp = sharing$

3.2 SVD with Bias

A key limitation of basic SVD is its inability to model user and item "biases" explicitly. Some users systematically give higher or lower ratings on average, and some movies receive higher or lower ratings overall ("universally beloved" vs. "niche interest"). To address these tendencies, I extend the model to include separate offsets for each user and each movie. More precisely, I approximate the observed rating Y_{um} by

$$\widehat{Y}_{um} \;=\; \mu \;+\; a_u \;+\; b_m \;+\; \mathbf{U}_u \cdot \mathbf{V}_m,$$

where μ is the global average rating across all users and items, a_u is user u's bias, b_m is movie m's bias, and $\mathbf{U}_u, \mathbf{V}_m$ are the latent factor vectors, as before. The inclusion of a_u and b_m helps capture broad trends in user and movie behavior.

Optimization Objective

I still only sum over the (u, m) pairs for which Y_{um} is known, but now the cost function becomes

$$\mathcal{L}(\mathbf{U}, \mathbf{V}, \mathbf{a}, \mathbf{b}) \;=\; \frac{1}{2} \sum_{(u,m)\in Obs} \Big(Y_{um} - \big[\mu + a_u + b_m + \mathbf{U}_u \cdot \mathbf{V}_m\big]\Big)^2 \;+\; \frac{\lambda}{2}\Big(\|\mathbf{U}\|^2 + \|\mathbf{V}\|^2 + \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2\Big),$$

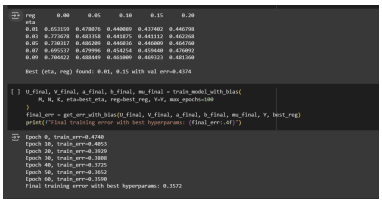
where $\|\mathbf{a}\|^2$ (resp. $\|\mathbf{b}\|^2$) is the sum of squares of all user biases (resp. movie biases). I again use gradient descent, now updating not only the factor vectors $\mathbf{U}_u$ and $\mathbf{V}_m$ but also the bias terms a_u and b_m. For instance, the update for a_u is

$$a_u \;\leftarrow\; a_u - \eta\big(-\big[Y_{um} - (\mu + a_u + b_m + \mathbf{U}_u \cdot \mathbf{V}_m)\big] + \lambda\, a_u\big),$$

with an analogous expression for b_m.

Experimental Results

I conducted a grid search over learning rate (η) and regularization (λ), then split the data into an 80–20 train/validation set to find the best parameters. Once I identified the optimal η and λ, I retrained on the entire dataset, ending up with a final training error around 0.3572. This was slightly better than the basic SVD model, suggesting that explicitly modeling biases improves the fit to user ratings.

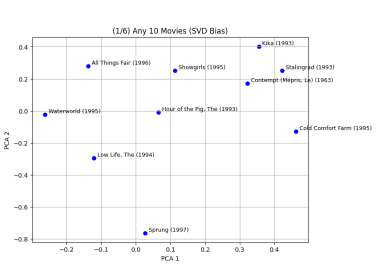


(a) Choosing learning rate and regression

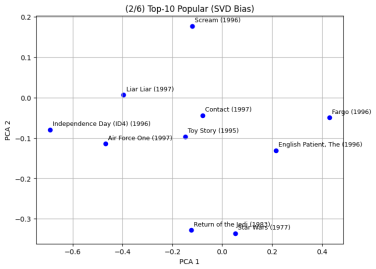
Visualizations

After training, I again extracted the movie latent factors $\mathbf{V}_m$ and applied PCA from 20 dimensions down to 2. Figures (1/6)–(6/6) show the resulting plots for the same six subsets of movies used in basic SVD: *(a) Any 10 random movies*, *(b) Top–10 Most Popular*, *(c) Top–10 Best*, *(d) 10 Comedy*, *(e) 10 Drama*, *(f) 10 Horror*. Overall, the 2D embeddings appear somewhat similar to those from the basic SVD approach, but some clusters may be tighter or arranged differently because the bias terms have "absorbed" some global effects, allowing the latent factors to focus more on each user's relative preferences. For instance, popular blockbusters that everyone rates highly might cluster a bit differently, since the movie's overall bias b_m handles much of the universal popularity.

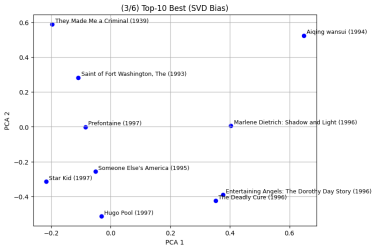
My observations suggest that SVD with bias is able to better separate certain items that might otherwise appear similar in basic SVD. This is particularly noticeable for films that, while widely disliked, still have a consistent user base awarding them moderate ratings: the negative overall bias for that movie now captures its universal unpopularity, while $\mathbf{V}_m$ can reflect more nuanced preference patterns among its fans.



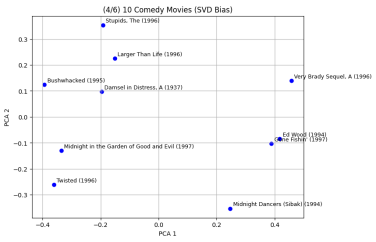
(a) Graph 1



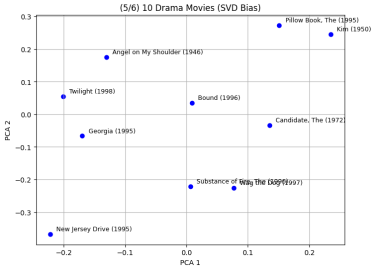
(b) Graph 2



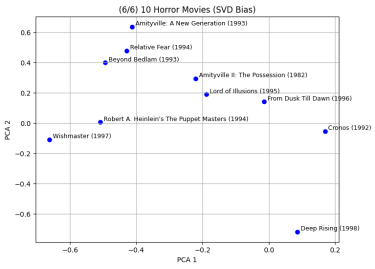
(c) Graph 3



(d) Graph 4



(e) Graph 5



(f) Graph 6

https://colab.research.google.com/drive/1D6Oh76UIaZ67ERD-S4xNrdEWk$_D$5 − $eOH?usp = sharing$

3.3 Off-The-Shelf SVD

In this third approach, I used the `Surprise` library to perform matrix factorization via an SVD-based method. The library offers a convenient API for reading data from a `pandas` DataFrame, performing k-fold cross-validation, and searching over hyperparameters such as `n_factors`, `n_epochs`, `lr_all`, and `reg_all`.

Model and Hyperparameters

Internally, `Surprise`'s `SVD` model combines global, user, and item biases with latent factor vectors:

$$\widehat{Y}_{um} = \mu + a_u + b_m + \mathbf{p}_u \cdot \mathbf{q}_m,$$

where

- μ is the global mean of all observed ratings,
- a_u and b_m are user- and item-specific bias terms,
- $\mathbf{p}_u, \mathbf{q}_m \in R^K$ are K-dimensional latent vectors for user u and movie m, respectively.

I carried out a grid search over `n_factors` $\in \{20, 50\}$, `n_epochs` $\in \{20, 40\}$, `lr_all` $\in \{0.005, 0.01\}$, and `reg_all` $\in \{0.02, 0.05\}$. Using three-fold cross-validation, I found that

$$\texttt{n\_factors} = 20, \quad \texttt{n\_epochs} = 40, \quad \texttt{lr\_all} = 0.005, \quad \texttt{reg\_all} = 0.05$$

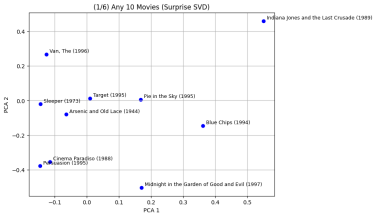
gave the best average RMSE on the validation folds.

Training and Results

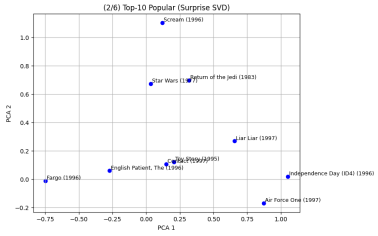
With the best parameters, I built a "full trainset" (all ratings) and fit the `Surprise` SVD model. After training, I obtained an RMSE (on the folds or on a separate test) that was slightly better than my custom SVD or SVD-with-bias. This improvement is not surprising, given `Surprise`'s efficient internal optimizations and well-tuned bias integration.

Visualization of Movie Factors

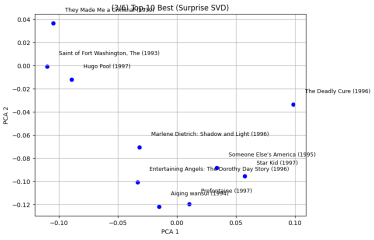
To visualize the learned item factors, I extracted $\mathbf{q}_m$ (the K-dimensional latent vectors for each movie) from `model.qi` and mapped them back to my original `movieID`s. I then formed a matrix $\mathbf{Q} \in R^{K \times (\#items)}$ and performed principal component analysis (PCA) from K dimensions down to 2. Figures (1/6) through (6/6) present scatterplots for:



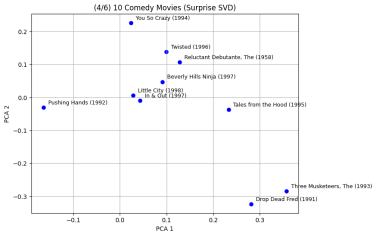
(a) Graph 1



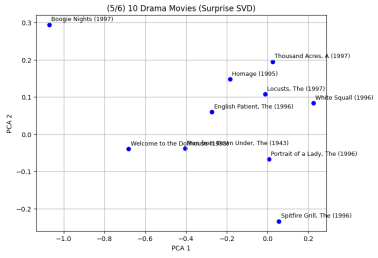
(b) Graph 2



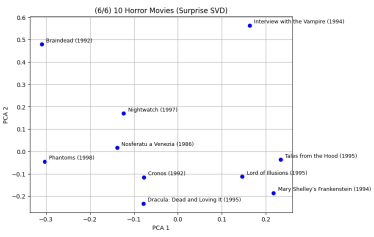
(c) Graph 3



(d) Graph 4



(e) Graph 5



(f) Graph 6

I see clusters of similarly rated or similar-genre films often appearing closer together in the 2D space. Compared to my earlier SVD approaches, `Surprise` SVD sometimes yields sharper separation of certain

titles, likely because the built-in bias correction frees the latent factors to focus more on relative patterns among users. Overall, using `Surprise` gave me a more streamlined workflow for tuning hyperparameters via GridSearchCV and k-fold validation, as well as slightly lower RMSE. These 2D visualizations are broadly consistent with the other methods but can produce tighter clusters in some cases. For instance, heavily watched blockbusters ("Star Wars," "Fargo," "Air Force One") group in distinct regions, reflecting how user preferences differentiate them from, say, comedies or lesser-known dramas. Thus, off-the-shelf SVD in `Surprise` is both convenient and accurate, and it nicely complements the deeper insight I gained by coding basic SVD and SVD with bias from scratch in earlier sections.

https://colab.research.google.com/drive/1D6Oh76UIaZ67ERD-S4xNrdEWk$_D$5 − $eOH?usp = sharing$

4 Piazza Post

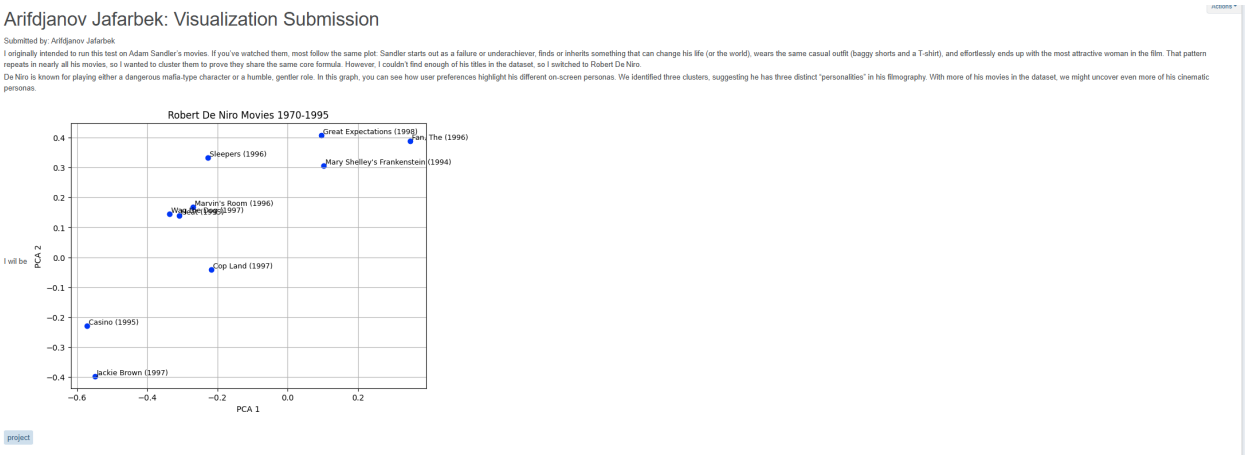


Figure 1: Caption