
- Group members: Stavya Arora, Isha Goswami, Ivy Brainard
- Team Name: SII
- Colab link: https://colab.research.google.com/drive/1zGOfSoeajEl7f_LV-F8ue6EMINOaA1Q9?usp=sharing
- Piazza link: https://piazza.com/class/m4xf8vvl4206zc/post/218
- Division of labor: Each of the group members contributed equally to this project. We each tried creating various visualizations and analyzed key concepts and findings together.
- Packages Used: surprise, pandas, numpy, seaborn, matplotlib, scipy

1 Introduction

Matrix factorization is a dimensionality reduction technique used to break down large matrices into smaller ones that reflect hidden patterns in the data, usually used with recommendation systems. In this project, we explored a few different matrix factorization techniques for the MovieLens Dataset that has ratings from 943 different users for 1682 different movies, as a precursor to a recommendation system for movies. Each user has rated at least 20 movies and we allotted 90,000 ratings for our training set and 10,000 for our test set. We start by visualizing the data with histograms, then we apply a traditional matrix factorization with stochastic gradient descent, then SGD that incorporates bias terms, and finally the *SVD surprise* package in Python. For each of the results from these methods, we project the movies into 2 dimensions and plot a random selection of 10 movies, the 10 most popular movies, the 10 best movies, and 10 randomly selected movies from the thriller, comedy, and mystery genres.

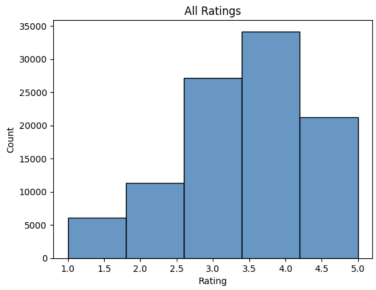
2 Basic Visualizations

Analysis

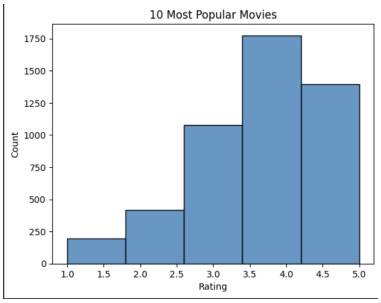
The overall distribution of all of the ratings is right skewed, with the majority of ratings falling between 3 and 4. In general, this shows that the individuals preferred to give above average ratings rather than very low ratings, since there was a peak at 4.0. The 10 most popular movies, as determined by the number of ratings, had a distribution very similar to the full dataset. These ratings also peaked around 3.5 to 4.0, meaning that while these movies had a larger amount of ratings, they weren't particularly the highest rated. However, the 10 best movies, which we determined from highest average ratings, showed that all ratings were a 5, indicating that all the individuals rated these movies very highly. However, we realized that this could be because there were simply very few ratings for many of these movies (max count was 16). To provide more clarity, we decided to plot the 10 best movies that had at least 20 ratings, which showed a more reasonable histogram with an obvious peak at 5.0, but still less commonly rated than the popular movies (based on the y-axis).

Now, for comparing the 3 genres, with comedy movies we have a similar distribution to the overall dataset, with a peak at 4 and a medium number of lower ratings. Mystery movies have fewer total ratings, with a peak around 3.5, indicating that they are less watched but tend to receive median ratings. Finally, the thriller movies had a distribution closer to comedy, with a strong peak at 4 but also quite a bit of lower ratings. In general, some conclusions that we made were that the most popular movies do not necessarily have the highest ratings, which is expected because widely watched movies tend to receive a broader range of ratings, including some negative ones. Our 10 best movies distributions aligned with the expectation that niche, critically acclaimed films get fewer but more passionate ratings. Finally, we noticed that Mystery movies had fewer ratings than the Comedy and Thriller movies, potentially due to it being less of a popular genre.

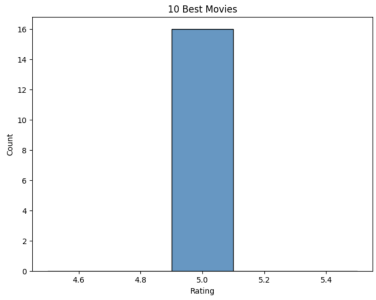
Images



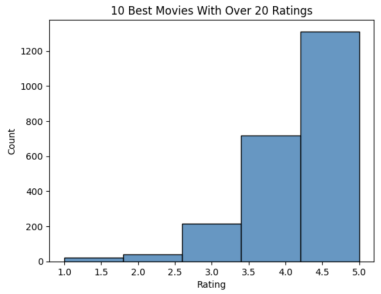
(a) All Ratings



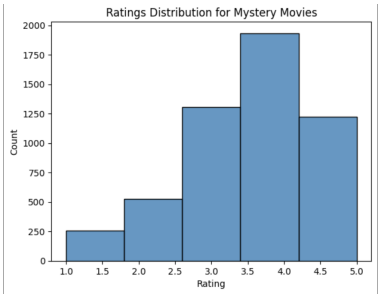
(b) 10 Most Popular Movies



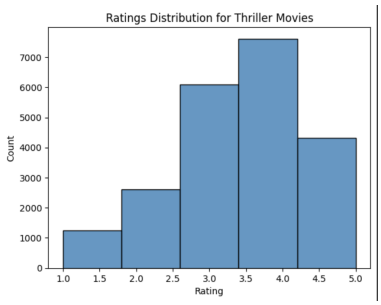
(c) 10 Best Movies



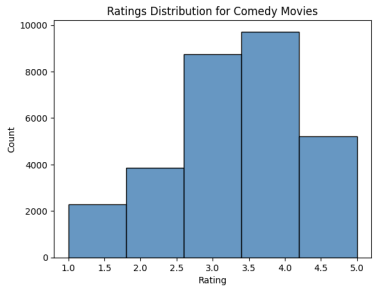
(d) 10 Best Movies with Over 20 Ratings



(e) Rating Distribution for Mystery Movies



(f) Rating Distribution for Thriller Movies



(g) Rating Distribution for Comedy Movies

3 Matrix Factorization Methods

We mainly used three different methods for matrix factorization as outlined in the project guide: matrix factorization with stochastic gradient descent from Homework 5, matrix factorization with bias terms, and finally an off-the-shelf implementation with a package in Python.

Method 1: Stochastic Gradient Descent (from Homework 5)

Traditional matrix factorization with stochastic gradient descent (SGD) works by updating the matrices U and V based on the gradient of the regularized squared error given by:

$$\arg\min_{U,V} \lambda \left(\frac{1}{2}\|U\|_F^2 + \|V\|_F^2 \right) + \frac{1}{2} \sum_{i,j} \left(y_{ij} - u_i^T v_j \right)^2$$

The gradient with respect to u_i is:

$$\partial_{u_i} = \lambda U_i - V_j \left(Y_{ij} - V_j^T U_i \right)$$

and the gradient with respect to v_j is:

$$\partial_{v_j} = \lambda V_j - U_i \left(Y_{ij} - V_j^T U_i \right)$$

During each epoch, all of the points in the training set are processed in a random order and the corresponding indices in U and V are updated with the learning right times their respective gradients. Our implementation stops when the ratio of the change in error from the current update to the change in error from the very first update is at most ϵ (we used 0.0001). We used k = 20 latent factors, η = 0.03, and λ = 0.01 based on trial and error that resulted in the lowest training and test error. This method achieved an RMSE of 0.300 on the training set and 0.446 on the test set.

Method 2: Stochastic Gradient Descent with Bias

The second method we tried was similar to method 1, but it incorporated bias terms to represent global trends in the rating of movies and how users rate on average across all of their ratings. Instead of relying entirely on latent factors, the bias terms allowed for some universal trends throughout all users or movies to be captured as well. The regularized squared error we used in our implementation is given by:

$$\arg\min_{U,V,a,b} \lambda \left(\frac{1}{2}\|U\|_F^2 + \|V\|_F^2 + \|a\|_F^2 + \|b\|_F^2 \right) + \sum_{i,j} \left((y_{ij} - \mu) - (u_i^T v_j + a_i + b_j) \right)^2$$

When using the bias terms, the gradient with respect to u_i is:

$$\partial_{u_i} = \lambda U_i - V_j \left(Y_{ij} - (V_j^T U_i + a_i + b_j) \right)$$

and the gradient with respect to v_j is:

$$\partial_{v_j} = \lambda V_j - U_i \left(Y_{ij} - (V_j^T U_i + a_i + b_j) \right)$$

We see that the main difference between methods 1 and 2 is the constant bias term, one for each user and one for each movie, that are present throughout the calculations. We used the same hyperparameters as method 1, resulting in a RMSE of 0.261 on the training set and 0.420 on the test set.

Method 3: Surprise SVD

The third and final matrix factorization we implemented was using the *SVD surprise* package in Python. The surprise SVD works similarly to method 2's implementation of SGD with bias terms. There are some nuanced differences which will be apparent, yet the regularized squared error minimized by surprise SVD is the same as the one optimized in our implementation of SVD with bias, meaning that the gradient updates to the bias matrices and U and V are the same as well. We use 50 latent factors, 100 epochs, $\eta = 0.05$, and $\lambda = 0.1$. The training error for this method was 0.666, while the test error was 0.918, making this our worst-performing matrix factorization method.

SVD with bias clearly performs better than SVD without, probably because users' ratings of movies are not entirely dependent on factors in the movies or of the users themselves, but also universal trends across movies or users. This is exactly what SVD with bias allows to be incorporated. For example, the best movies may not have features that particularly stand out, but it is still rated highly by all users, which is picked up on with the bias term. *Surprise SVD* may be the worst model because it initializes the entries for U and V by sampling from a standard normal distribution, while our implementation initializes the values from a uniform distribution, limited to values from -0.5 to 0.5. This type of initialization can lead to larger or more varied initial values, which might require more training iterations to converge effectively, especially if the model uses regularization. Initializing to values from -0.5 to 0.5 provides more controlled and bounded initializations, which can help the model converge more quickly and avoid issues with excessively large updates during training. The normal distribution overall has a more unstable nature than the uniform distribution, as reflected in higher RMSE from *Surprise SVD*.

4 Matrix Factorization Visualizations

Analysis

First, for the matrix factorization model we used trial and error to tune the learning rate, and regularization by testing different values and observing their impact on training and test errors. Additionally, we used our knowledge from the previous homework assignment. Starting with a wider range, we adjusted eta incrementally, finding that values around 0.03 provided stable convergence without overshooting, while lower values made training too slow. For regularization, we tested values across an order of magnitude and found that numbers around 0.1 effectively balanced preventing overfitting while still allowing the model to learn meaningful patterns, which is why we selected similar values in this range for our final parameter choice.

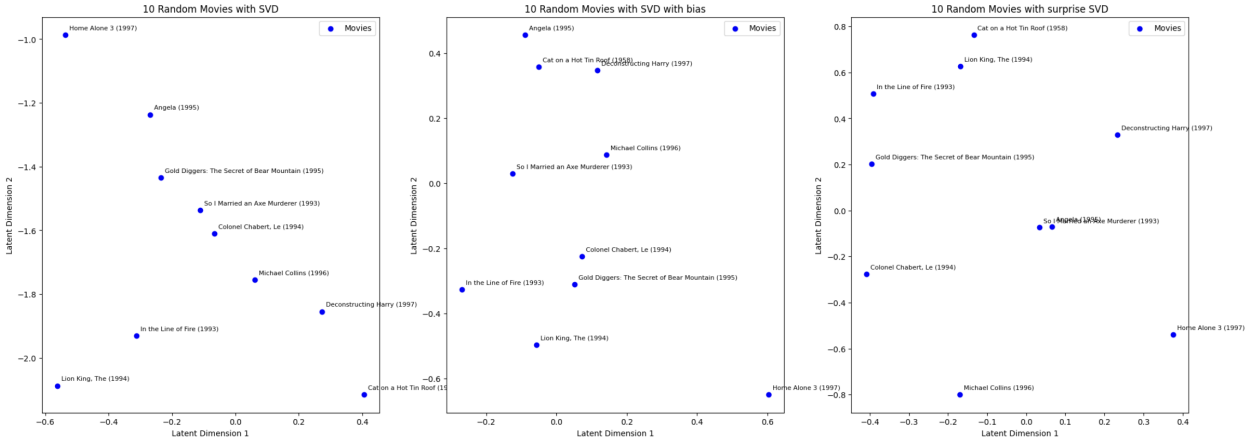
The matrix factorization visualizations revealed several interesting trends about movie preferences and latent space structures. The visualization of the most popular movies –based on the highest number of ratings– tended to cluster closely in the latent space, suggesting that widely watched movies share common appeal characteristics. In contrast, the visualization of the best-rated movies –based on average ratings– showed a more dispersed spread, likely because highly rated movies span different genres and niche interests. Comparing genre-based visualizations, mystery and thriller movies were more closely clustered together, indicating shared thematic or stylistic elements, while comedy movies were more scattered, reflecting their broader variety in humor and audience reception.

Comparing the visualizations produced by different matrix factorization methods, the SVD method without bias showed more compact clustering, while the biased SVD approach showed slight variations in spread, potentially due to its incorporation of user and item biases. The surprise SVD implementation yielded a more refined latent space, with distinct separation among movie groups, suggesting improved predictive power. Across all methods, the 2D projections effectively captured relationships between movies, with genre-based patterns and rating-based distributions aligning well with expectations. The results highlight the strength of matrix factorization in revealing underlying structures in user preferences and movie characteristics.

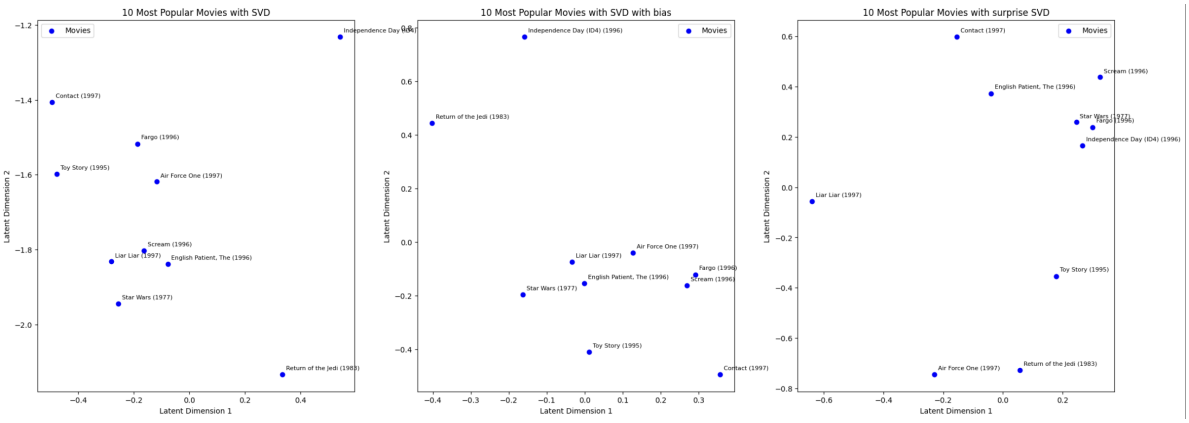
Another interesting observation is how the latent space encodes similarities between movies beyond just their numerical ratings. For example, in the plots of random movies, we can see that movies with similar themes or styles often appear closer together, even if they have different levels of popularity. This suggests that factorization captures deeper latent features, such as mood, style, or target audience, rather than just user preferences in isolation. Additionally, the separation of movies in the latent space varies depending on the factorization method used. The standard SVD method tends to spread movies more uniformly, while the biased SVD and Surprise SVD methods show more defined clusters, potentially due to their enhanced ability to account for systematic rating biases. This is particularly evident when visualizing movies within specific genres, where distinct genre-based groupings emerge more clearly in the Surprise SVD results. Moreover, when comparing the distribution of best-rated versus most popular movies in latent space, it becomes evident that popularity does not necessarily correlate with high ratings. Popular movies tend to form a more cohesive cluster, indicating that many users rate them similarly, whereas the best-rated movies are often more dispersed, reflecting their more niche appeal. This highlights an important insight: while popular movies are widely viewed, the highest-rated movies may not be universally well-known but are highly appreciated by a specific subset of users. These findings reinforce the strength of

matrix factorization techniques in uncovering meaningful movie relationships and user preferences that go beyond simple rating averages.

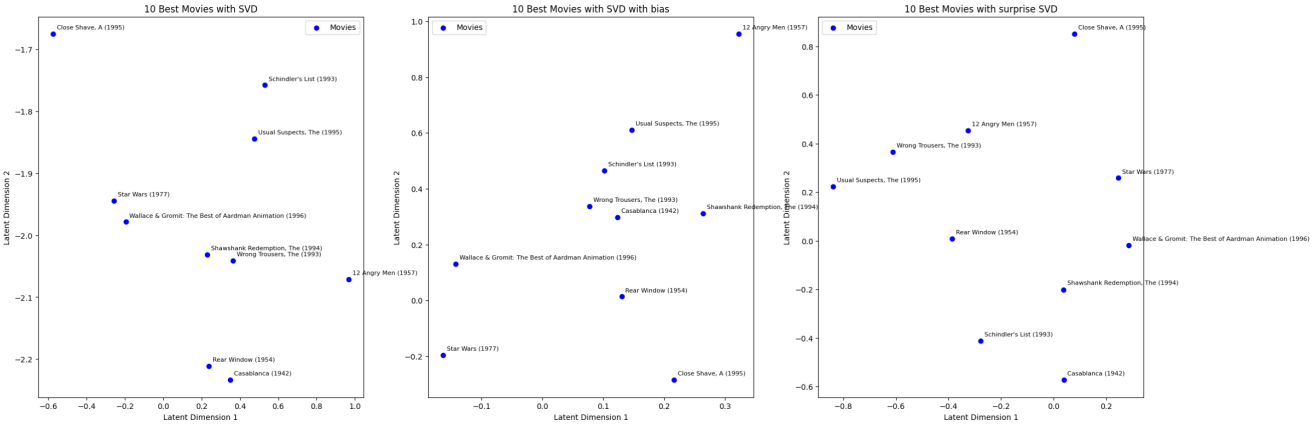
Images



(h) 10 Random Movies with All Methods

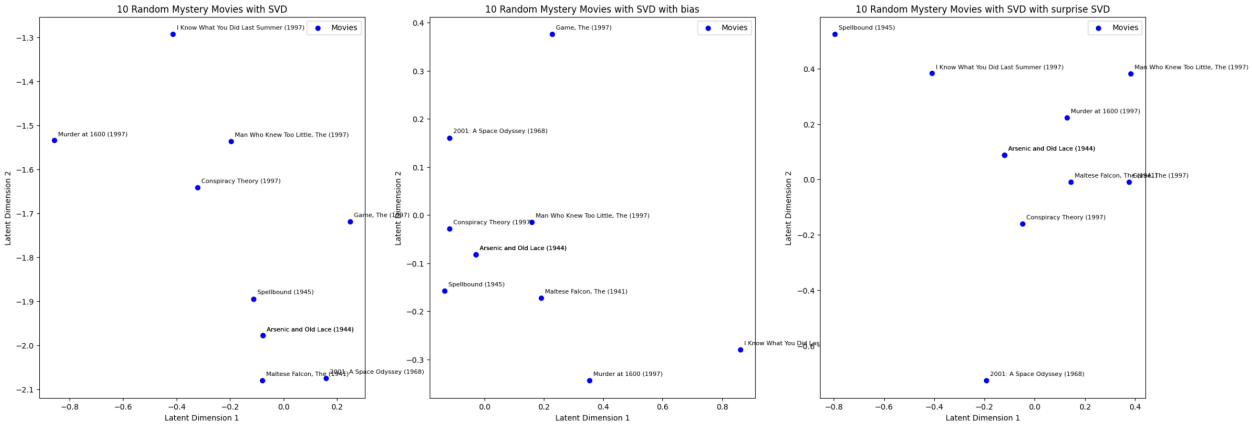


(i) 10 Most Popular Movies with All Methods

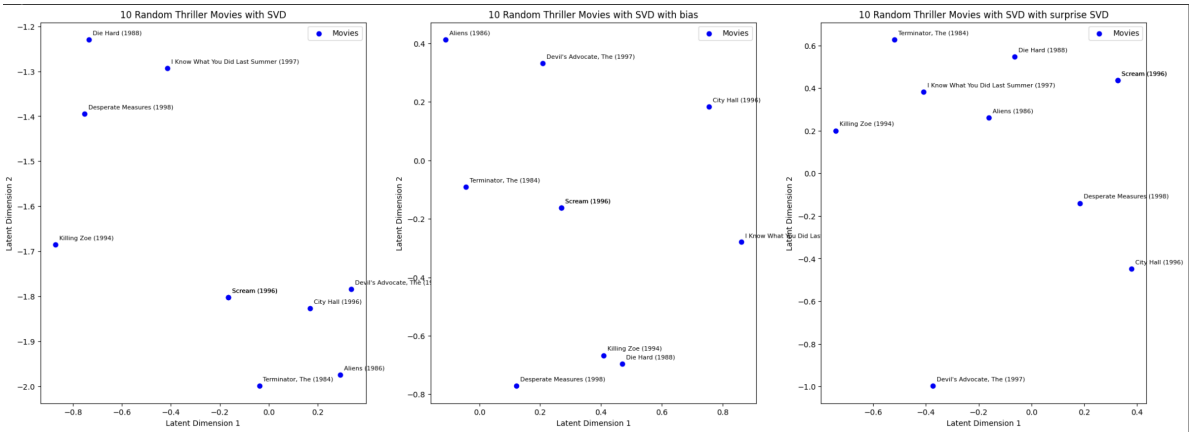


(j) 10 Best Movies with All Methods

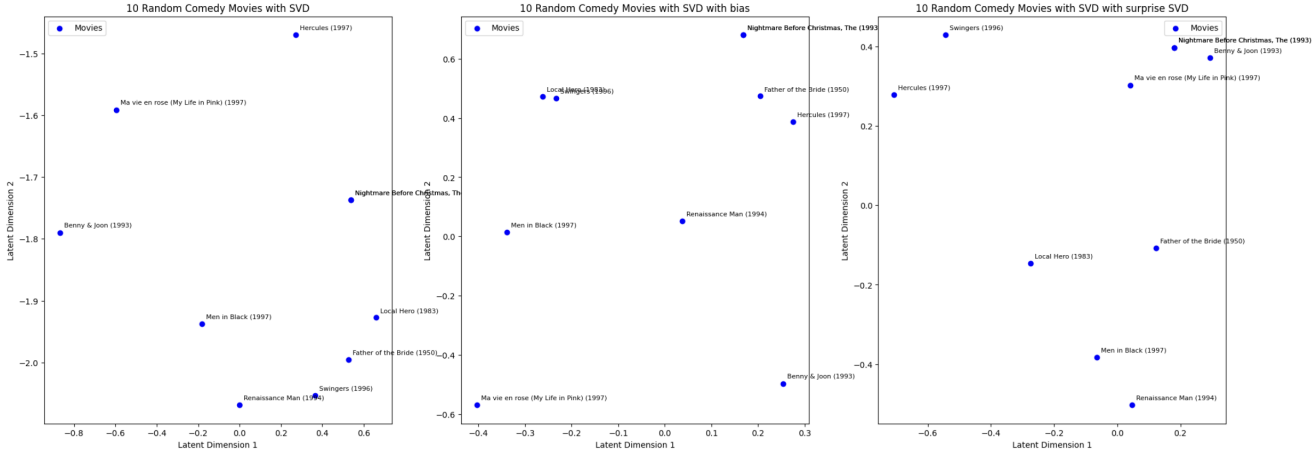
Images (continued)



(k) 10 Random Mystery Movies with All Methods



(l) 10 Random Thriller Movies with All Methods



(m) 10 Random Comedy Movies with All Methods

5 Conclusion, Challenges, and Discussion

Based on our analysis of various matrix factorization methods for the MovieLens data set, we can conclude that user preferences do not always follow a pattern, but there are some global trends that model preferences better. This is evidenced by the better performance of SVD with bias. Since our lowest test error was around 0.4, latent factors alone may not be sufficient for user recommendation systems for movies. Incorporating factors like user demographics, language, budget and production studio could prove to be more impactful. Additionally, alternative methods that combine collaborative filtering with content-based methods Moreover, reducing the 20 latent factors to two dimensions made it difficult to clearly interpret the data and what the clusters actually mean.

6 LLM Usage

Explanation for LLM

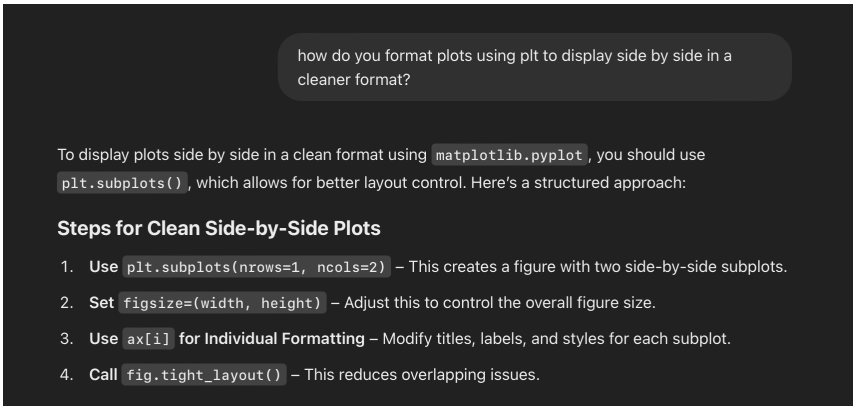


Figure 1: Clean Plotting

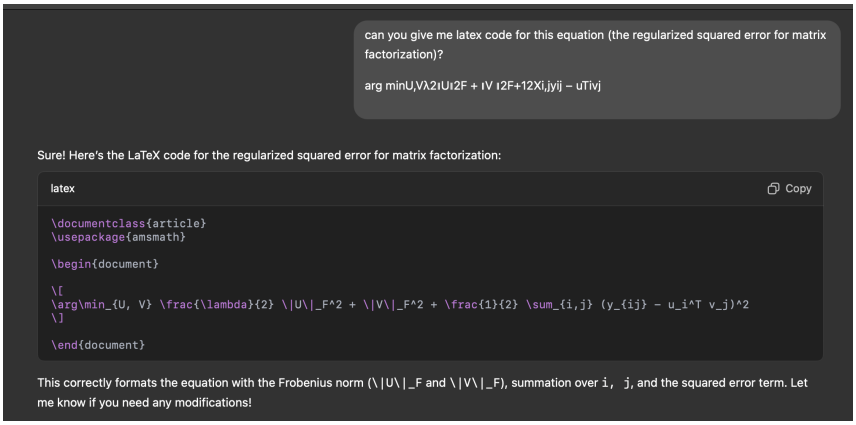


Figure 2: LateX Equation