Movie Recommendations using Movielens 100 K

Mohini Nath, Iris, Rishita Mylavarapu, Poojan Prerak Shah, Jeya Krishna Ganapathy Raman

Key Questions

1

Millions of users engage daily with content on social media, streaming platforms, and shopping sites.

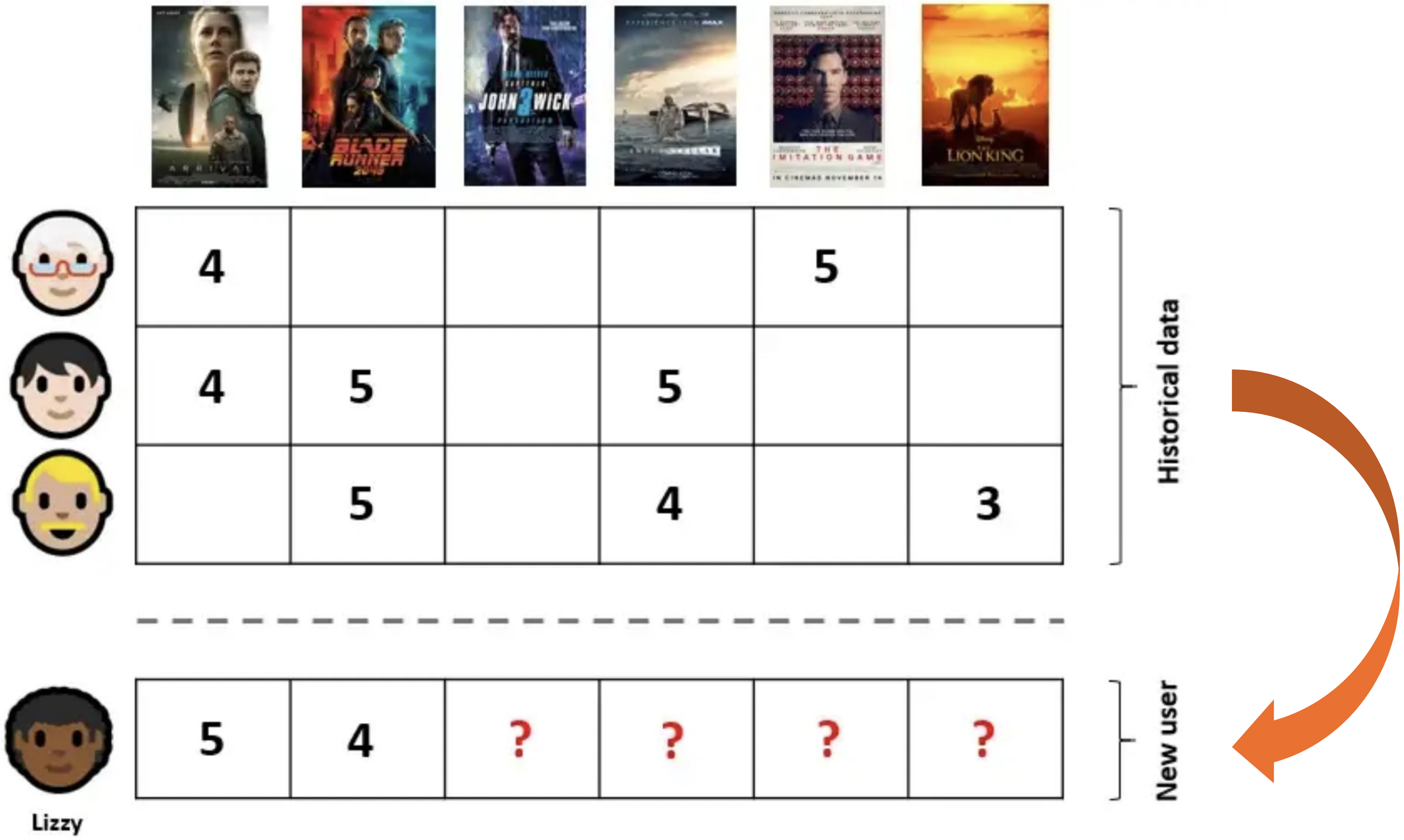
How can we accurately predict which unseen items they are most likely to enjoy, to drive higher engagement and boost profitability?

2

Which machine learning algorithms are most effective for predicting user preferences for unseen items?

What are the advantages and limitations of each approach, and in what contexts does each algorithm perform best?

Problem Definition



<u>Use historical data to predict new user behavior</u>

We are given a sparse matrix where:

- Rows = Users
- Columns = Movies
- Entries = Known ratings (e.g., on a scale of 1 to 5)

Goal: Predict the unknown rating Rij that user i would give to movie j

Analysis Procedure

Data preprocessing

- Combined different datasets/information about users, movies and user/movie interactions
- Transformed user ratings per user to a mean centered rating to remove bias
- Created a test train split where 20% of each user's ratings were withheld for testing
- No other data cleaning was needed

Analysis

- Tried different algorithms to predict user ratings namely,
  - SVD (Singular value decomposition)
  - Soft/Hard impute
  - Two tower model using NNs
- Parameter trained each model to ensure best results - done by comparing change in RMSE/MAE with change in parameter

Evaluation

- Evaluated the performance of each model using MAE/RMSE
- Further compared the outputs of each model by comparing predictions
- Discussed why and in what cases is one model superior to the other

Model 1 SVD : Description

Goal

We decompose a user-item rating matrix into lower-dimensional latent factors to reconstruct or predict missing entries

We approximate the (filled-in) user-item rating matrix $R \in \mathbb{R}^{m \times n}$ as:

$$R \approx U \Sigma V^T$$

Where:

- $U \in \mathbb{R}^{m \times k}$: user latent factor matrix
- $\Sigma \in \mathbb{R}^{k \times k}$: diagonal matrix of singular values
- $V \in \mathbb{R}^{n \times k}$: item latent factor matrix
- k: number of latent dimensions (rank)

- Each user i is represented by a latent vector $u_i \in \mathbb{R}^k$
- Each item j is represented by a latent vector $v_j \in \mathbb{R}^k$
- Predicted rating is:

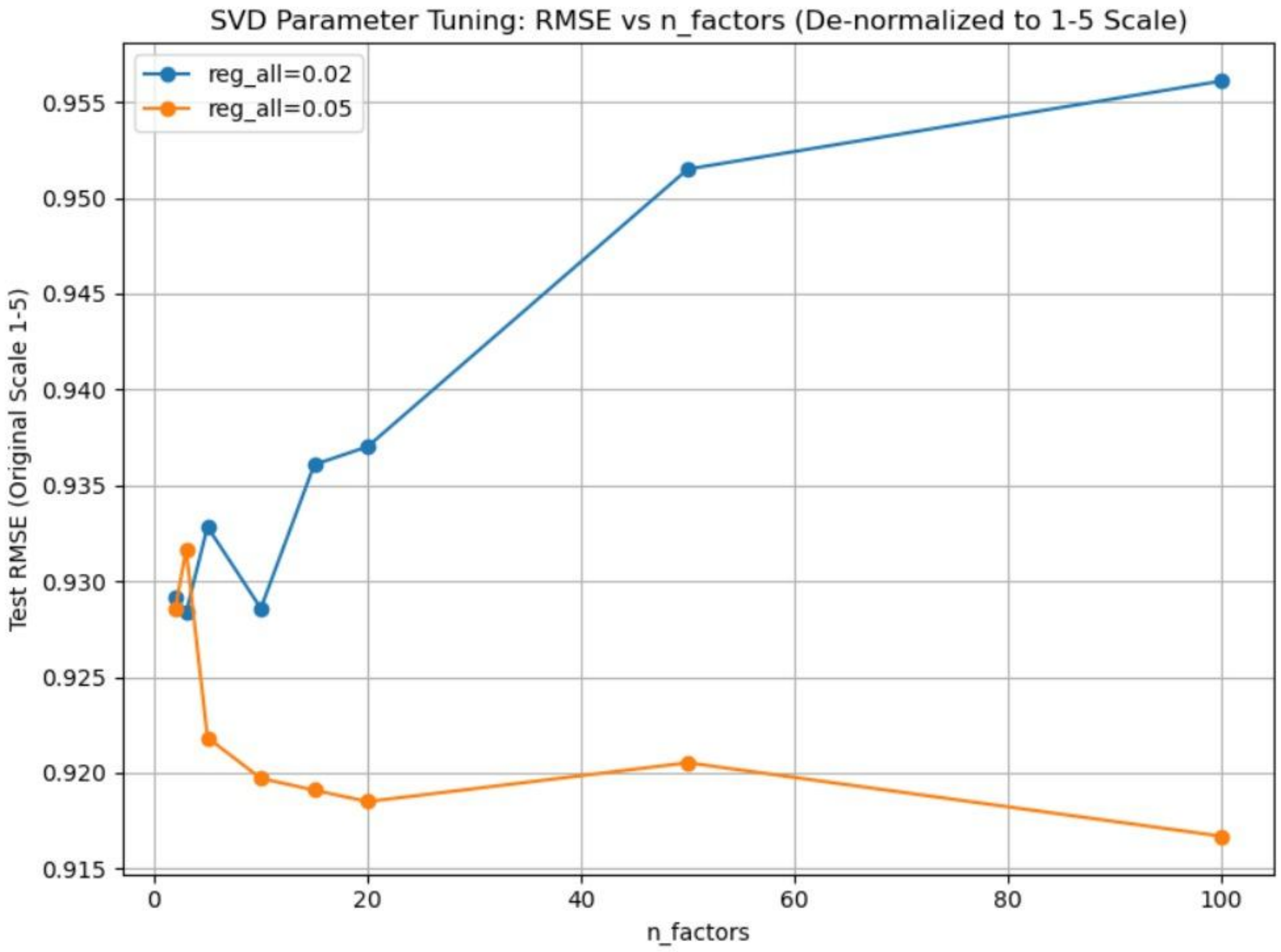
$$\hat{R}_{ij} = u_i^T \Sigma v_j$$

This factorization <u>helps identify latent features</u> representing user preferences and item characteristics.

<u>The dot product of these factors estimates missing ratings</u>, allowing us to generate recommendations even for sparse datasets.

Model 1 SVD : Outputs

SVD Model Performance Across Hyperparameters



SVD Parameter Tuning – Manual Results

- reg_all = 0.05 consistently outperforms reg_all = 0.02 across all n_factors.
- For reg_all = 0.05, RMSE improves sharply from n_factors = 2 to 15.
- Beyond n_factors = 15, RMSE stabilizes with very minimal improvement.
- For reg_all = 0.02, RMSE shows an initial drop till n_factors = 5–10 but then increases significantly as n_factors increase, indicating overfitting.
- The best RMSE ≈ 0.917 was achieved at n_factors = 15 with reg_all = 0.05.
- Increasing n_factors beyond 15 yields diminishing returns, with slight fluctuations.

Conclusion:

Stronger regularization (reg_all = 0.05) helps control overfitting effectively.
The optimal balance is found at n_factors = 15, beyond which complexity adds little benefit.
The trend confirms the necessity of exploring small n_factors to avoid misleading tuning choices.

Model 2 Soft/Hard Impute : Description

Goal

Optimization Problem

Algorithm

Hard impute

Find a **low-rank matrix approximation** of the observed matrix by minimizing the reconstruction error, while **enforcing a rank constraint.**

Given partially observed matrix M and observation mask Ω, find a **low-rank matrix** X that matches M at observed entries.

$$\min_X \quad \|P_\Omega(X - M)\|_F^2 \quad \text{subject to } \operatorname{rank}(X) \leq k$$

Where:

- $P_\Omega(\cdot)$: Projection operator that keeps observed values and masks others

⭐ Rank K is tuned = 3

Soft Impute

Instead of setting a hard rank limit, penalize the nuclear norm (sum of singular values), which **promotes low-rank solutions but in a soft, continuous way.**

$$\min_X \quad \frac{1}{2}\|P_\Omega(X - M)\|_F^2 + \lambda\|X\|_*$$

⭐ Lambda is tuned = 10

Repeat Until Convergence (for both Soft and Hard Impute):

1. **Compute SVD:**

$$X^{(t)} = U\Sigma V^T$$

2. **Apply Thresholding to Singular Values:**

- For **Soft Impute** (shrink):

$$\sigma_i' = \max(\sigma_i - \lambda, 0)$$

- For **Hard Impute** (truncate):

$$\sigma_i' = \begin{cases} \sigma_i, & \text{if } i \leq k \\ 0, & \text{if } i > k \end{cases}$$

3. **Reconstruct** updated matrix:

$$X_{\text{new}} = U\Sigma' V^T$$

4. **Project back the observed entries:**

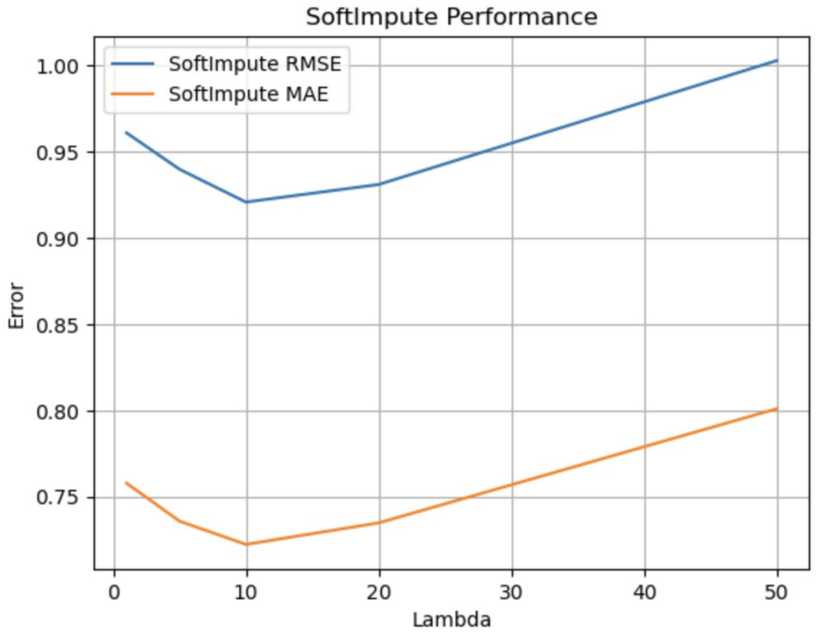
$$P_\Omega(X^{(t+1)}) = P_\Omega(M)$$

⭐ *Indicates parameters we parameter tuned*

Model 2 Soft/Hard Impute :Tuning Curves

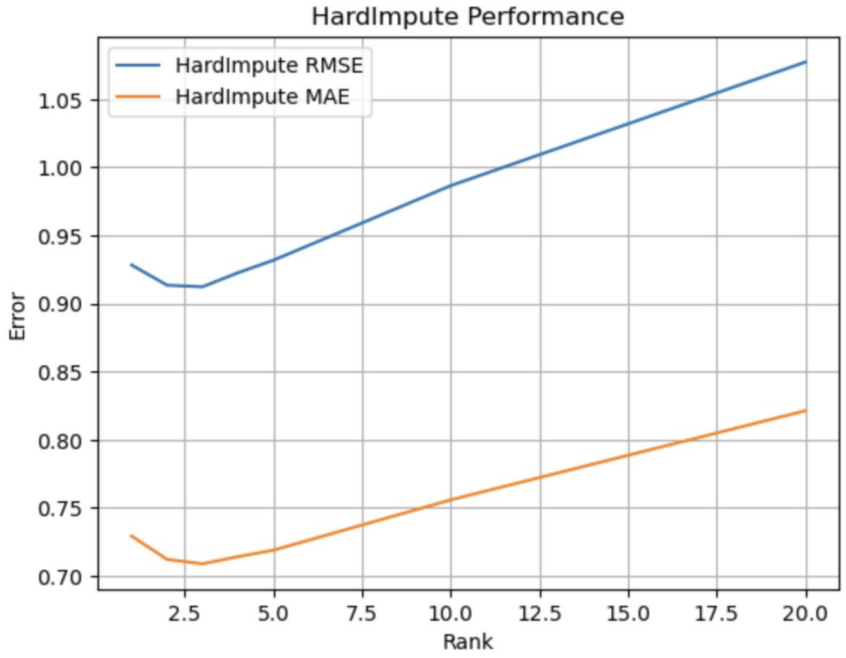
SoftImpute Tuning Curve: Lambda vs. Test RMSE

- Shows how **test RMSE** changes as the **shrinkage parameter λ** increases.
- **Observation**:
  - Test RMSE **decreases** until λ = 10, then begins to **rise** again.
  - This confirms λ = 10 is near-optimal for your data, balancing bias and variance.



HardImpute Tuning Curve: Rank vs. Test RMSE

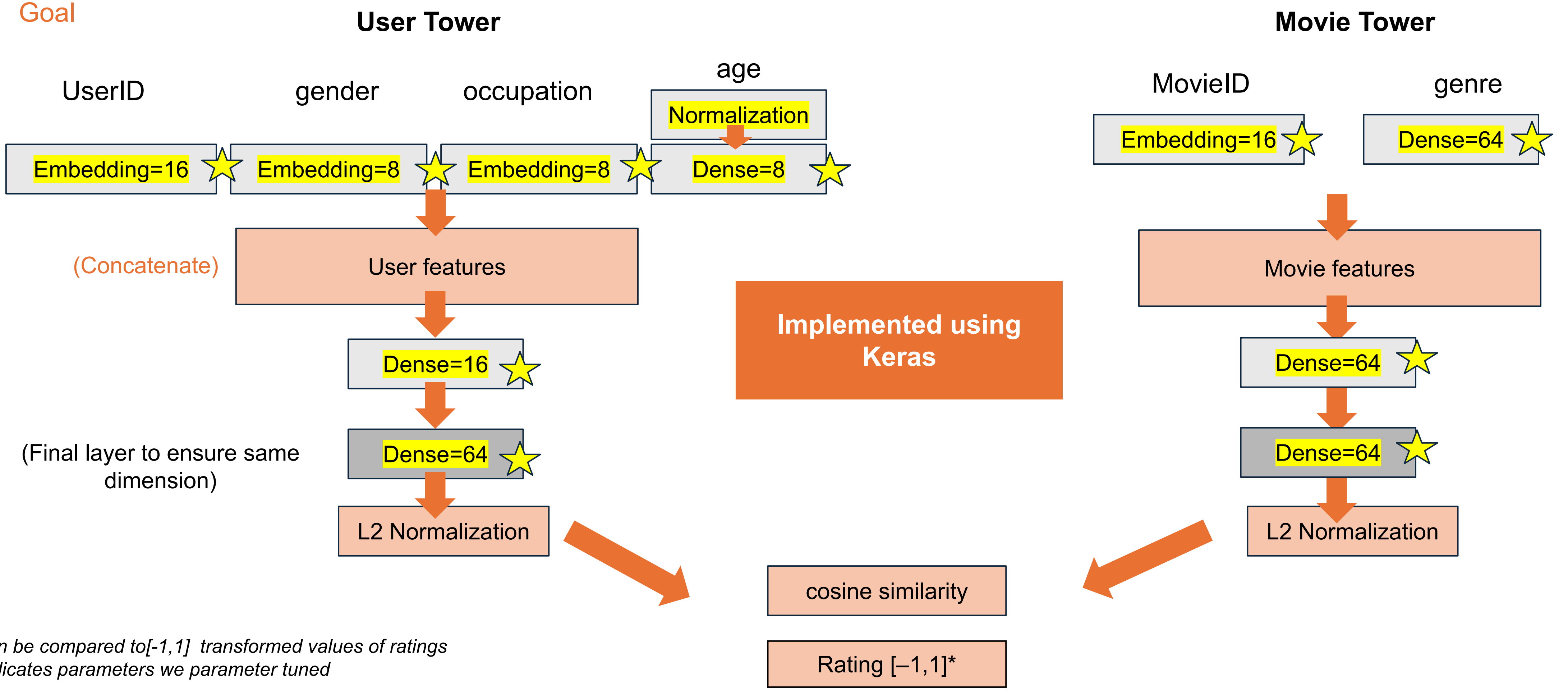
- Shows how **test RMSE** varies with increasing **rank truncation** level.
- **Observation**:
  - RMSE drops until rank = 3, then increases again.
  - Rank = 2-4 appears optimal; using too many singular vectors may overfit.



Model 3 Two Tower Model : Description

Goal

Embed user and movie features using neural networks to learn latent representations, and predict user ratings by combining all available user demographics and movie content features



** Can be compared to[-1,1] transformed values of ratings*

☆Indicates parameters we parameter tuned

Performance

Model 3 Two Tower Model : Outputs

There is very little difference between MAE/RMSE between the top 5-10 architectures and any could be used

RMSE

```
Top 5 by Train RMSE:
emb64_user128_movie128_final64: Train RMSE=0.3099, Test RMSE=0.3944
emb64_user128_movie64_final64: Train RMSE=0.3157, Test RMSE=0.3957
emb64_user64_movie128_final128: Train RMSE=0.3172, Test RMSE=0.3932
emb64_user64_movie128_final64: Train RMSE=0.3174, Test RMSE=0.3977
emb64_user64_movie64_final64: Train RMSE=0.3189, Test RMSE=0.3983
```

```
Top 5 by RMSE:
emb16_user16_movie64_final64: Test RMSE=0.3908, Train RMSE=0.3481
emb16_user32_movie16_final128: Test RMSE=0.3917, Train RMSE=0.3502
emb16_user8_movie64_final128: Test RMSE=0.3919, Train RMSE=0.3628
emb64_user8_movie64_final128: Test RMSE=0.3920, Train RMSE=0.3462
emb32_user16_movie64_final64: Test RMSE=0.3920, Train RMSE=0.3406
```

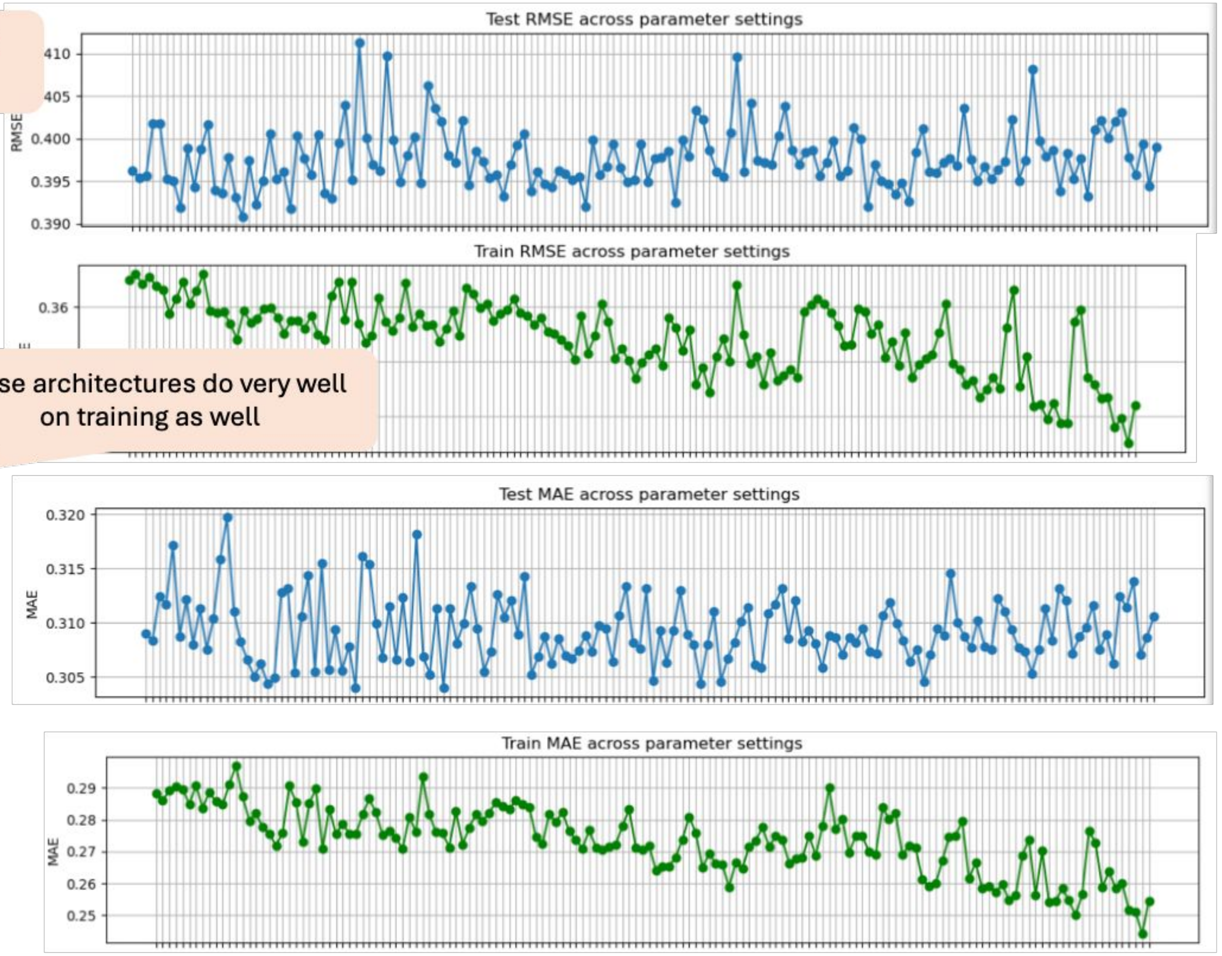
MAE

```
Top 5 by Train MAE:
emb64_user128_movie128_final64: Train MAE=0.2441, Test MAE=0.3086
emb64_user64_movie128_final64: Train MAE=0.2499, Test MAE=0.3087
emb64_user128_movie64_final128: Train MAE=0.2511, Test MAE=0.3070
emb64_user128_movie64_final64: Train MAE=0.2516, Test MAE=0.3138
emb64_user64_movie32_final64: Train MAE=0.2541, Test MAE=0.3083
```

```
Top 5 by MAE:
emb16_user128_movie32_final64: Test MAE=0.3040, Train MAE=0.2713
emb16_user64_movie8_final128: Test MAE=0.3040, Train MAE=0.2816
emb16_user16_movie128_final64: Test MAE=0.3043, Train MAE=0.2719
emb32_user64_movie16_final64: Test MAE=0.3043, Train MAE=0.2648
emb32_user64_movie32_final128: Test MAE=0.3046, Train MAE=0.2660
```

Similar Architectures

These architectures do very well on training as well



Final Test RMSE :~0.4 & MAE :~0.31

Different Parameter Combinations

RMSE and MAE are calculated on mean centered [-1,1] ratings

Comparison of Performance and Discussion*

	SVD	Soft/Hard Impute	Two-Tower
MAE	~ 0.723	~ 0.72	~ 0.75
RMSE	~ 0.921	~ 0.92	~ 0.96

Critique

SVD

X Performs poorly on **highly sparse matrices** and we had ~93% missing values

X Couldn't incorporate **user or item metadata** (age, gender, genres)
X Couldn't incorporate **non-linear interactions** if any are present (e.g. User A watches drama movies if they get an Oscar)

Soft/Hard Impute

While **SVD requires a fully observed matrix, soft/hard impute is inherently designed to deal with missing data**

- In SVD we're forced to fill missing entries (i.e. impute) before running SVD.
- These imputed values distort the structure, especially when >90% of entries are missing and SVD treats those as true signals (i.e.It tries to approximate values you never observed, which leads to poorer predictions)

Two-Tower

X high computational complexity and initial training time especially for large data set
X Hard to tune as it has many possible viable architectures/parameters (~200K parameters)

Performs better than SVD/imputation **even when most ratings are missing** because of the additional features used

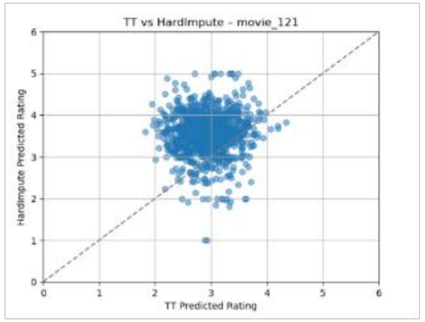
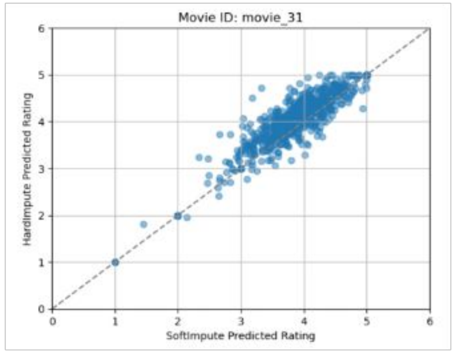
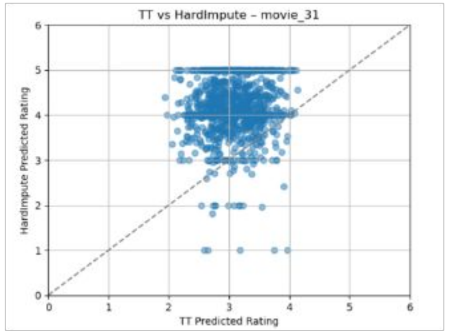
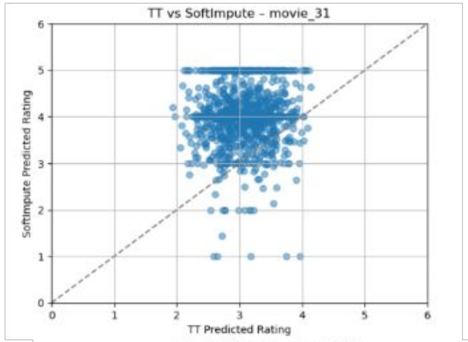
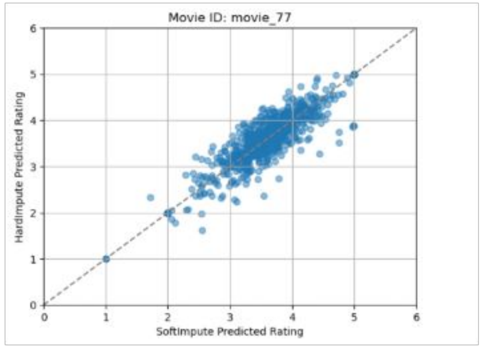
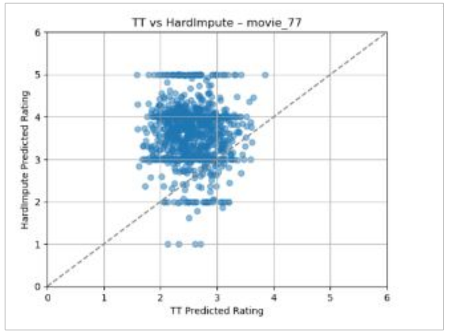
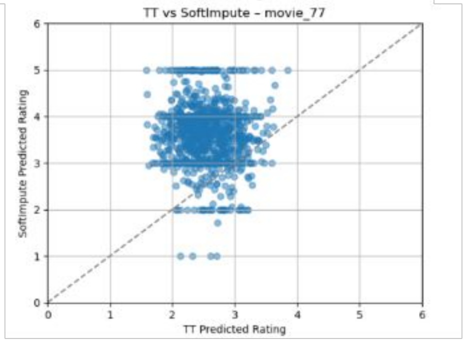
- **Side information** such as user demographics, movie genres etc
- **Can model nonlinear interactions**

No need for side information/simple model: it works best when only user–item ratings are available.

Fairly low time complexity (OMN^2 for SVD and OKMN for soft/hard impute) and could work fast for large number of users
Fewer parameters to tune (only ridge penalty/n_factors)

RMSE and MAE are calculated on 1-5 ratings

Comparison of Performance and Discussion – Directional comparison of predictions

Movie ID	Hard vs Soft Impute	Hard vs NN	Soft vs NN
121			
31			
77			

There are high correlations between the predictions of soft and hard impute while predictions with the two tower model seem uncorrelated

Comparison of Performance and Discussion – Top 5 recommendations

	Hard Impute	Soft Impute	Two Tower
 121	20%		
 31	40%		
 77	20%		

For these 3 users there is no overlap between the recommendations of soft and hard impure and the two tower model

This lack of overlap between the imputes and two tower model is likely because of very different input data that has gone into the models

As expected there are no overlaps between the two tower model and Soft/Hard imputes.
There is some overlap between the recommendations given by soft and hard impute

Conclusion

- Given the **similar RMSE/MAE** across methods and the **low overlap in top-k recommendations or correlations between predicted ratings**, we conclude that each method captures different aspects of user preference.
  - In particular, the two-tower model likely performs better for users with additional available features (e.g., age, gender, occupation), leveraging its capacity to incorporate side information.
- This diversity in strengths suggests that overall performance could be improved by **ensemble approaches** that combine predictions from multiple models, tailoring to the strengths of each for different user segments

Learnings

Challenges/Reflections

X Comparing outputs of a prediction algorithm like the two tower model with matrix completion (SVD/Soft/Hard impute)

X How should the test train split be done? & Cold-start issues

X Data sparsity is a serious challenge

X Missing Not at Random Data

X Simple models work well

Solution/Reflections

✓ Brought **cosine similarities and ratings to have the same scale** so cosine similarities could act as ratings

✓ Done at a user level for users with >5 ratings rather than a sample of 20% to **ensure real world like behavior** also helps address the issue with cold start. In the real world these users will be given random recommendations until they rate movies

✓ Given the sparsity of the dataset, **regularization techniques were crucial to prevent overfitting** and likely why high regularization in SVD was preferred

✓ Movielens data was not missing at random data (**which is an assumption for SVD/Soft/Hard Impute**) because more popular movies get rated more. **The two tower model was less affected by this fact.**

✓ Despite inability to generate additional features, SVD/soft/hard impute work very well (almost comparable to NNs) – **likely because it is a low rank latent space**

-> Can further look into ensemble models / temporal dynamics to better performance