Data Description and Insights Analysis

Description of structure of content

movieId: ID of the movie being rated (integer).
rating: User's rating for the movie (integer, e.g., 1-5 scale).
userId: ID of the user who gave the rating (integer).

top 10 movies w/ highest avg ratings

movieId	avg_rating	num_ratings
32	2.9166666666666665	12
90	2.8125	16
30	2.5	14
94	2.473684210526316	19
23	2.466666666666667	15
49	2.4375	16
29	2.4	20
18	2.4	15
52	2.357142857142857	14
53	2.25	12

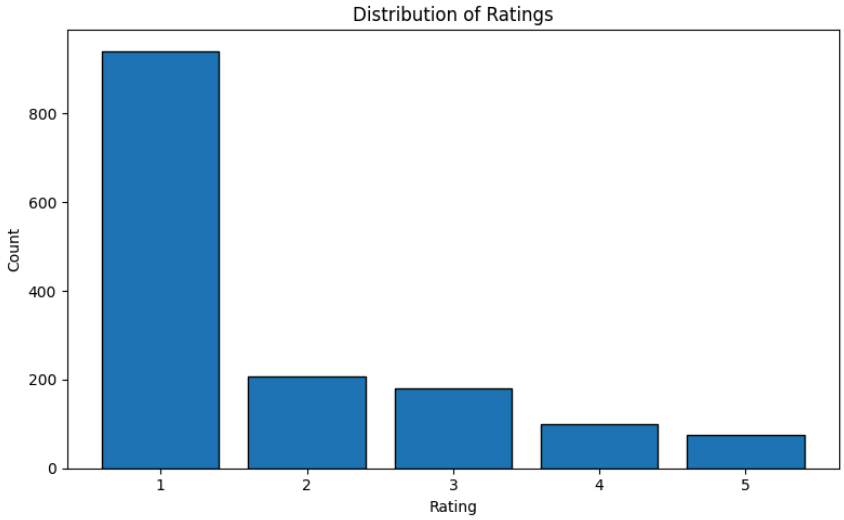
top 10 Users Who Gave the Most Ratings

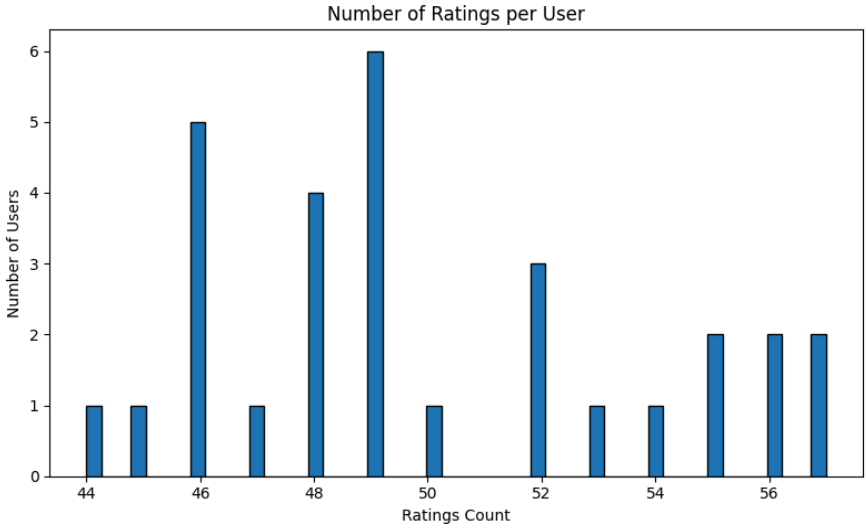
userId	count
6	57
14	57
22	56
11	56
12	55
4	55
7	54
9	53
23	52
24	52

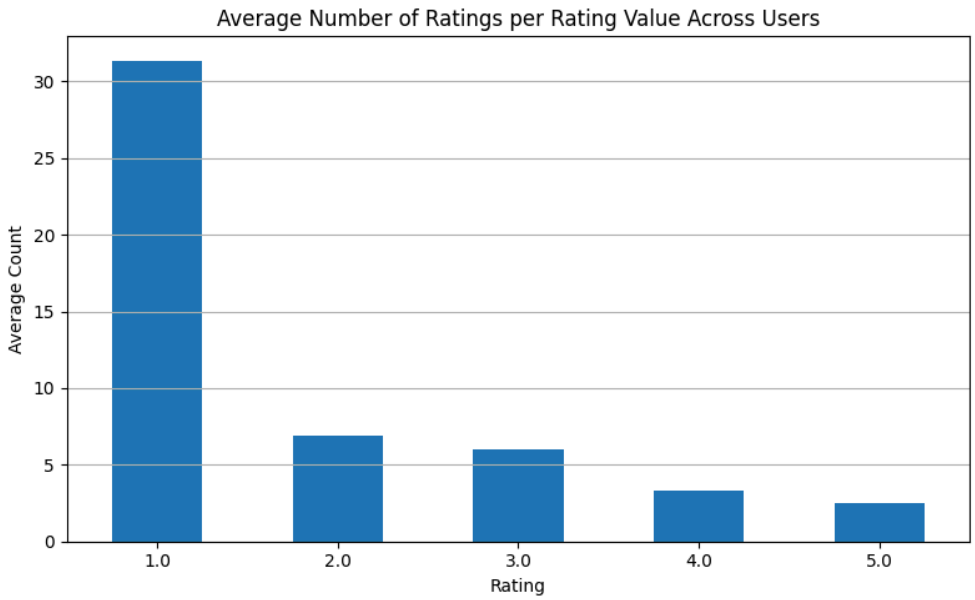
influence of top10 users who gave the most ratings

- dataset statistics (e.g. mean rating) can be biased: inflated or deflated by these users' ratings.
- Users who rate less frequently may get less tailored recommendations, as the model prioritizes high-frequency users' influence.

distribution of ratings, number of ratings per user







marketing implications based on user engagement and rating tendencies

the above plots reveal two user behavioral patterns and implications

Observation1:

- A majority of ratings are 1-star, with very few 4 or 5 ratings.
- Implications:Investigate why most ratings are 1: is it due to a few lower-rated movies, or people just don't like movies on this platform in general.

Observation2:

- Most users gave between 44 to 57 ratings, the spread of number of ratings per user is not too wide.
- Implications: since not wide spread, the recommendation system is in general not biased (not heavily influenced by few users who give a lot of ratings)

Split Dataset and Performance Assessment

compare performance of different splits

RMSE for 70/30 split: 1.0172
RMSE for 80/20 split: 1.0663

Identify most effective config

70/30 split model is better, because it has a lower RMSE (1.0172 < 1.0663)

In-Depth Evaluation of Error Metrics

Define metrics

- RMSE (root mean squared error)
- MSE (Mean Squared Error)
- MAE (Mean Absolute Error)
- Precision@k: Of the top-k recommended movies i.e. rated>=4, what fraction are actually liked i.e. rated>=4
- Recall@k: Of the movies a user actually liked, what fraction are captured in the top-k recommendations

Note for Precision and Recall, use k=5, since from the plot “Average Number of Ratings per Rating Value Across User”, note on average #ratings=4 + #ratings=5 is around 5. Also, pick ratings equal and above 4 as “liked”.

Strength and weakness of each metrics

RMSE computes root mean square error between predicted and true ratings, penalizing larger errors more heavily.
Weakness is that it may not reflect ranking quality.

- E.g. Assume for movie 1,2,3; true rating: 5,4,1. Model A predicts: 4.7, 4.1, 1.2; Model B predicts 4.1, 4.7, 1.2. Note RMSE for model A and B are the same, but A would recommend movies in order: 1, 2, 3 which matches the true rating; while B would recommend movies in order 2, 1, 3, which NOT match the true rating.

MSE computes root mean square error between predicted and true ratings.
MAE computes the mean absolute difference between predicted and true ratings.

- MSE and MAE both have the same limitation as RMSE: measure only *how close each predicted number is to its true value*, not the **ordering** which is needed for recommendation;
- MAE also lacks emphasis on large errors (e.g. abs error between rating 5 and 3 is 2; while square error is 4).

Precision@k computes the fraction of top-k recommended items that are actually liked (here I define “liked” as rated ≥ 4).
Its weakness is that it only cares about the first k recommendations and ignores other predictions.

Recall@k computes how many of the relevant (liked) items are successfully included in the top-k recommendations.
It may be unfairly low when users have many liked items.

F1@k is computed from both precision and recall, providing a balanced score of model relevance and coverage.

Trade-off when selecting metrics

RMSE / MSE / MAE
-Sparse Data: These metrics may misrepresent model quality (example as above)
- Imbalanced Ratings: If most ratings in the dataset are high, these metrics can be low even when the model fails to predict rare low ratings (e.g. predicts all high ratings).
- Trade-off: Good for rating prediction tasks, but risk overlooking ranking quality and underrepresented cases.

Precision@k
- Sparse Data: works well, since it focuses only on top-k predictions
- Imbalanced Ratings: Precision can be inflated if very few items are relevant (e.g., rated ≥ 4), as even random guesses may appear precise.
- Trade-off: High precision doesn't guarantee broad relevance or fairness; it's biased toward users with few high ratings.

Recall@k
- Sparse/imbalanced Data: could be very low if very few high ratings per user.

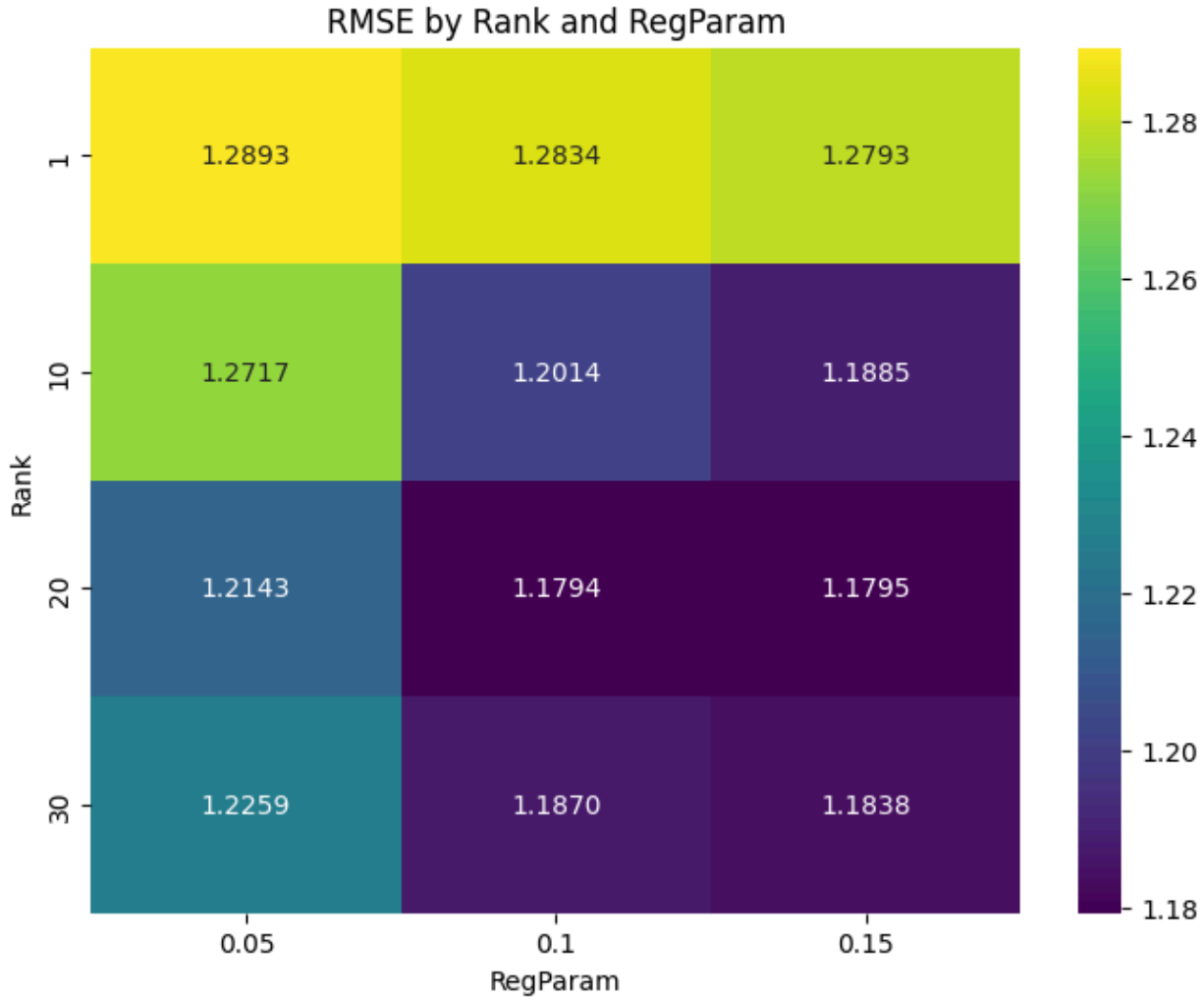
F1@k
- harmonic mean of Precision@k and Recall@k.
- In sparse or imbalanced settings, it provides a more reliable metric.

Model performance

Metrics for 70/30 split:
Precision@k: 0.0545
Recall@k: 0.1591
F1@k: 0.0812
RMSE: 1.0172
MSE: 1.0347
MAE: 0.7124

Metrics for 80/20 split:
Precision@k: 0.0588
Recall@k: 0.1961
F1@k: 0.0905
RMSE: 1.0663
MSE: 1.1370
MAE: 0.7442

Hyperparameters Tuning Using Cross-Validation Techniques



Best model RMSE on test set: 1.0472
Best model parameters:
  rank: 20
  regParam: 0.1

hyperparams impact on performance & training time; how to balance

Rank

- Controls the dimensionality of the latent user/item vectors
- Higher rank can capture more nuanced patterns, but training time grows roughly linearly with rank

regParam

- Controls how much the model penalizes complex user/item vectors, Prevents overfitting, especially on sparse data
- lower values, model tends to overfit; higher value, tends to underfit (i.e. predict average ratings for all)

Personalized Recommendations and Analysis for Selected Users

Recommendations for user 21, 11

```
cv's best model recommends
+------+--------------------------------------------------------------------------------------+
|userId|recommendations                                                                       |
+------+--------------------------------------------------------------------------------------+
|21    |[{29, 4.0893693}, {53, 3.9491887}, {52, 3.8032875}, {63, 3.3225036}, {2, 3.2723577}]|
|11    |[{32, 4.8393064}, {18, 4.685277}, {30, 4.5811334}, {27, 4.5199957}, {23, 4.1939397}]|
+------+--------------------------------------------------------------------------------------+
```

How ALS uses user ratings to generate recommendations

ALS uses user ratings i.e. R, to learn U and P s.t.
- Each u_i best explains the ratings user i gave
- Each p_j best explains the ratings item j received

e.g. if R_ij=5 i.e. user i rates item j with 5, ALS tries to make dotprod(u_i, p_j=5)

Once ALS learns U and P, to recommend new item for a given user i:
- ALS compute predicted scores for all items
- Exclude items user already rated
- Sort the remaining by predicted score, and recommend top-k items, where k is specified by user

ALS for dataset with limited features

- ALS works well with limited features — it learns patterns directly from user-item ratings.
- No metadata needed (e.g., no genres or user info required).
- May struggle with very sparse or small datasets i.e. very few available ratings for it to train on.

Compare base v.s. Refined models, features to add for better personalized recommendations

1. base model (70/30 split, maxIter=10, regParam=0.1, rank=10)

```
baseline model recommends
+------+--------------------------------------------------------------------------------------+
|userId|recommendations                                                                       |
+------+--------------------------------------------------------------------------------------+
|21    |[{53, 3.7534876}, {29, 3.5829422}, {52, 3.5488617}, {76, 3.5134597}, {2, 3.4579816}]|
|11    |[{32, 4.7528567}, {30, 4.6563816}, {27, 4.5116687}, {46, 4.368993}, {18, 4.3575163}]|
+------+--------------------------------------------------------------------------------------+
```

Metrics:
Precision@k: 0.0545
Recall@k: 0.1591
F1@k: 0.0812
RMSE: 1.0172
MSE: 1.0347
MAE: 0.7124

2. refined model (70/30 split, maxIter=10, regParam=0.1, rank=20)

```
cv's best model recommends
+------+--------------------------------------------------------------------------------------+
|userId|recommendations                                                                       |
+------+--------------------------------------------------------------------------------------+
|21    |[{53, 4.056582}, {52, 3.659092}, {29, 3.6516068}, {2, 3.4786358}, {74, 3.2658987}]|
|11    |[{32, 4.772703}, {18, 4.6721554}, {30, 4.5145817}, {27, 4.452958}, {23, 4.270186}]|
+------+--------------------------------------------------------------------------------------+
```

Metrics:
Precision@k : 0.0455
Recall@k : 0.1061
F1@k : 0.0636
RMSE : 1.0472
MSE : 1.0965
MAE : 0.7037

Features to add:
- User features: demographics (age, gender, jobs, etc.), location
- Movie features: tags or keywords