Book Recommendation System

CP421 Data Mining Project

Arsalan Khan

Asil Khan

Waleed Ghufran

Spencer Robson

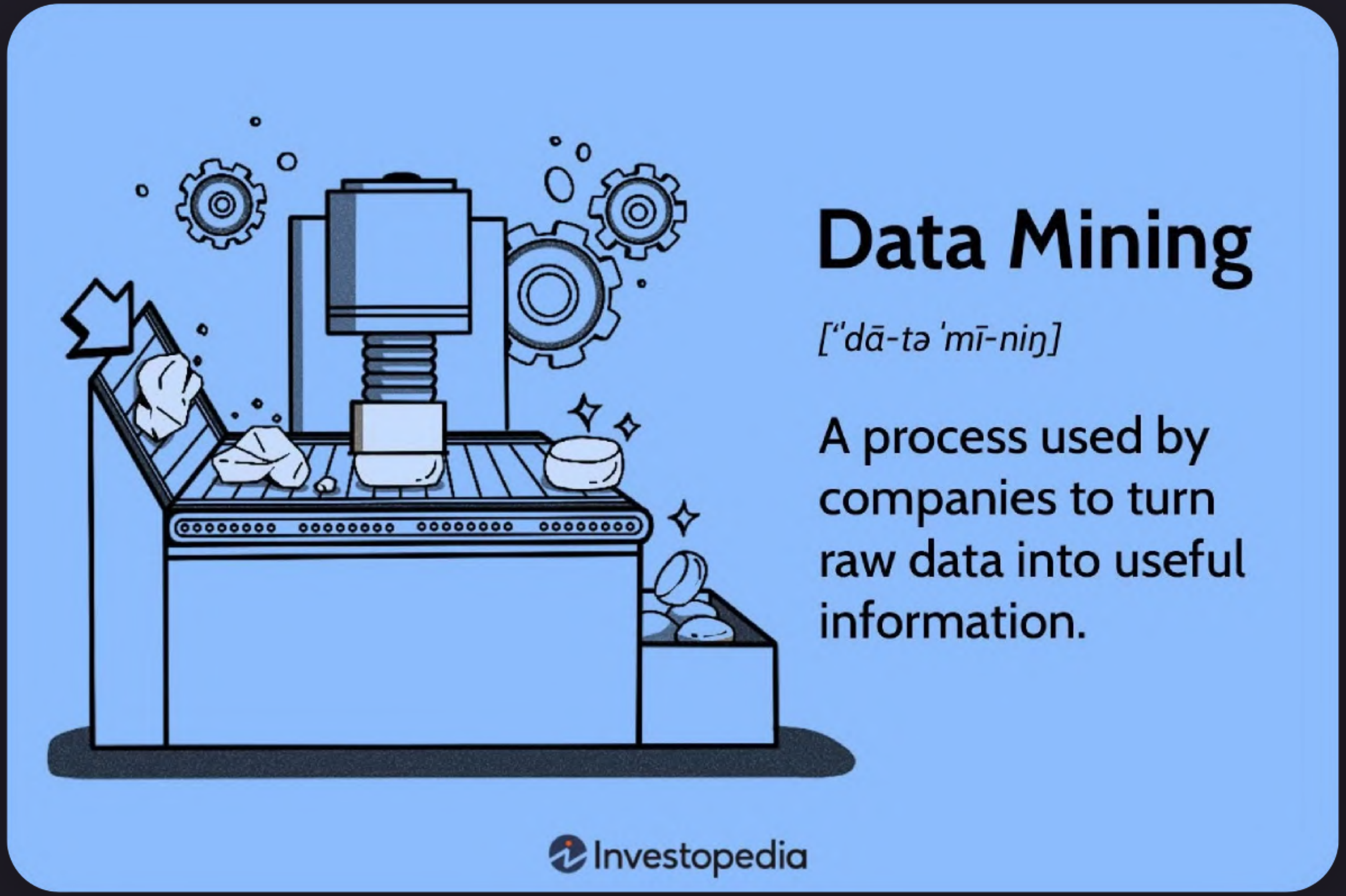
Aliha Ali

Qusai Almasri

Table of Contents

Content:

1. Introduction
2. Dataset Description
3. Data Preprocessing
4. Methodologies
5. RMSE Comparison
6. Discussion
7. Conclusion
8. Questions & Answers



Data Mining

[“dā-tə ˈmī-niŋ]

A process used by companies to turn raw data into useful information.

Book Recommendation

Introduction

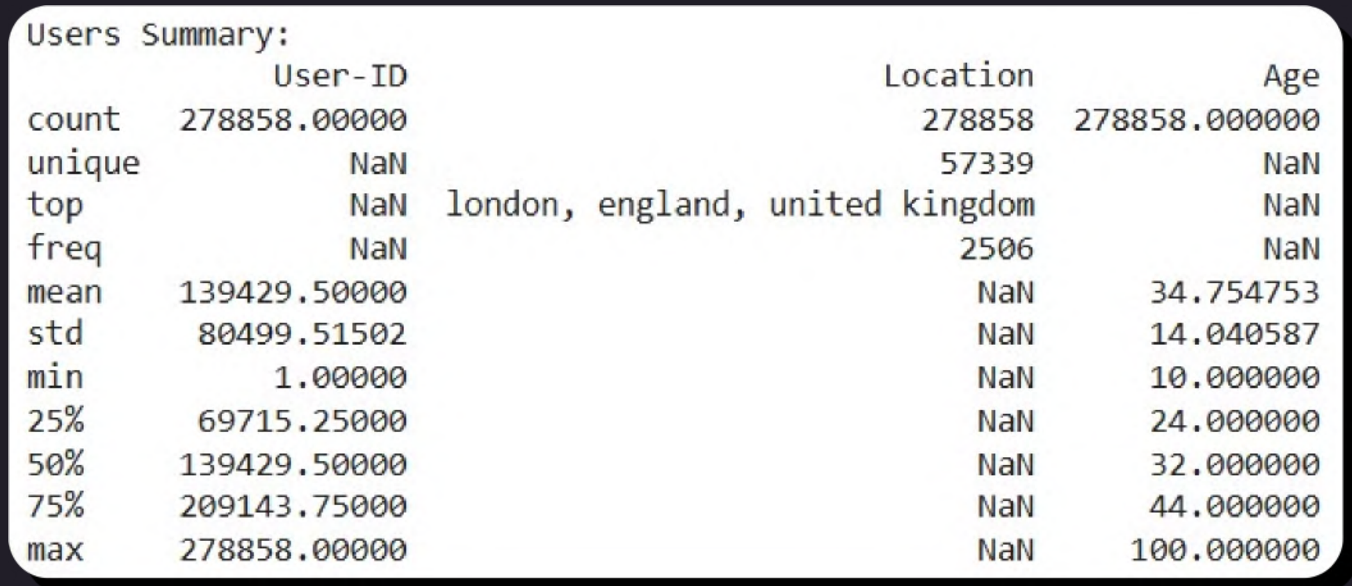
□ Importance of recommendation systems in the information age.

□ Project objectives:

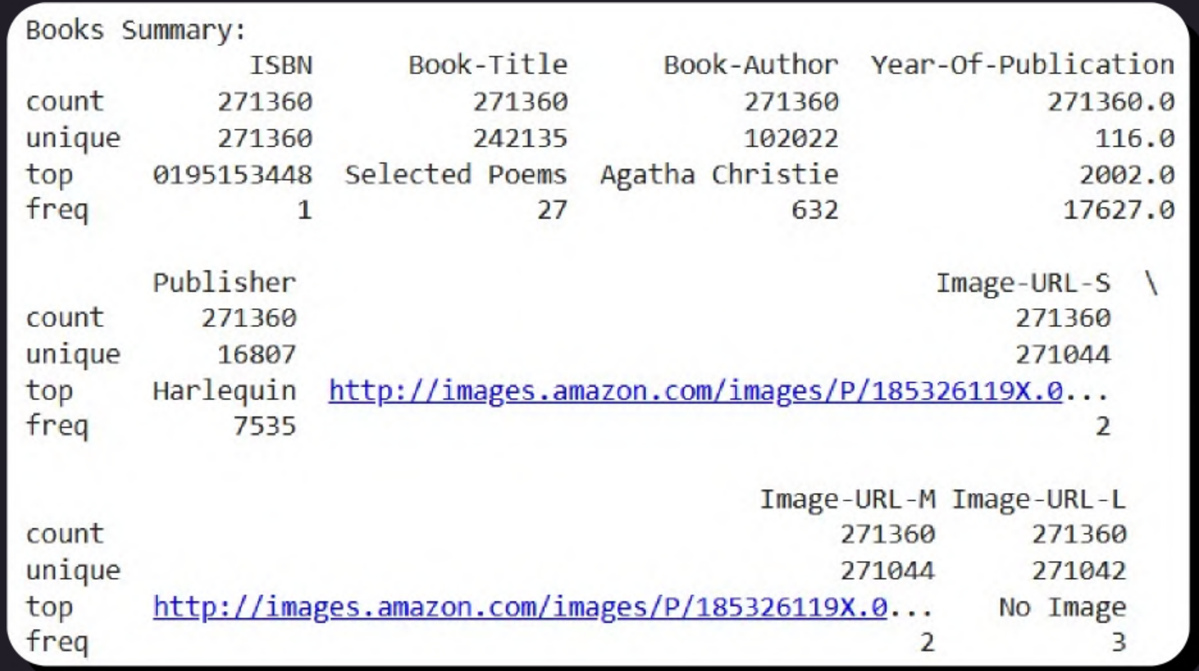
- Implement various recommendation approaches.
- Compare their performance using evaluation metrics.
- Enhance system accuracy with advanced methods.

Dataset Description

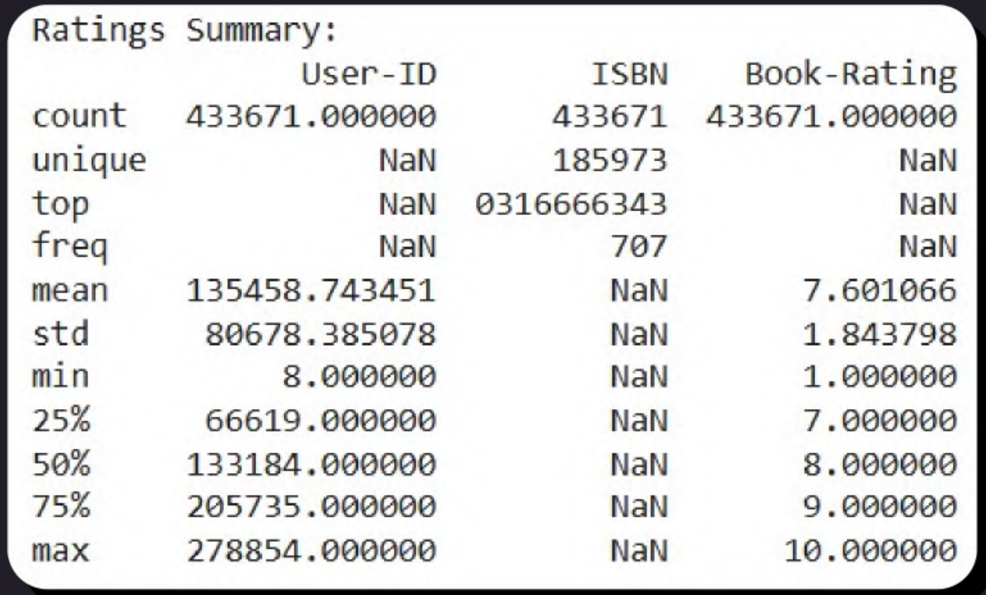
- □ **Book-Crossing Dataset**:
  - Users.csv: User demographics.
  - Books.csv: Book metadata.
  - Ratings.csv: User ratings.
- □ **Dataset Statistics**:
  - Users: 278,858 entries.
  - Books: 271,360 entries.
  - Ratings: 1,149,780 entries.
  - Ratings scale: 1 to 10.



```
Users Summary:
             User-ID                         Location            Age
count   278858.00000                           278858  278858.000000
unique           NaN                            57339            NaN
top              NaN  london, england, united kingdom            NaN
freq             NaN                             2506            NaN
mean    139429.50000                              NaN      34.754753
std      80499.51502                              NaN      14.040587
min          1.00000                              NaN      10.000000
25%      69715.25000                              NaN      24.000000
50%     139429.50000                              NaN      32.000000
75%     209143.75000                              NaN      44.000000
max     278858.00000                              NaN     100.000000
```



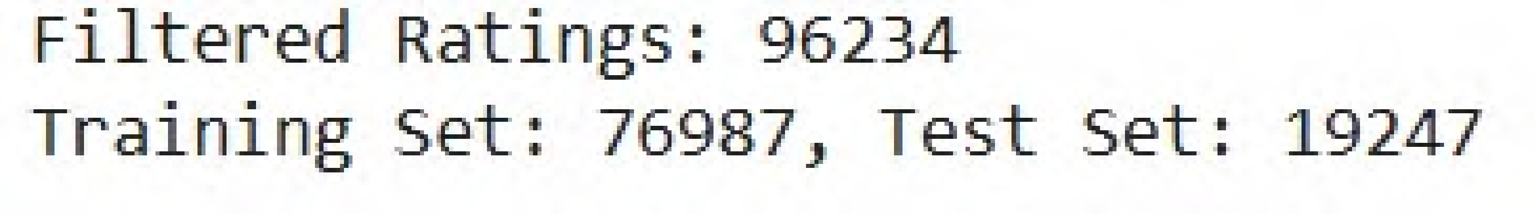
```
Books Summary:
              ISBN      Book-Title      Book-Author  Year-Of-Publication
count       271360          271360           271360             271360.0
unique      271360          242135           102022                116.0
top     0195153448  Selected Poems  Agatha Christie               2002.0
freq             1              27              632              17627.0

        Publisher                                       Image-URL-S  \
count      271360                                            271360
unique      16807                                            271044
top     Harlequin  http://images.amazon.com/images/P/185326119X.0...
freq         7535                                                 2

                                              Image-URL-M Image-URL-L
count                                              271360      271360
unique                                             271044      271042
top     http://images.amazon.com/images/P/185326119X.0...    No Image
freq                                                    2           3
```



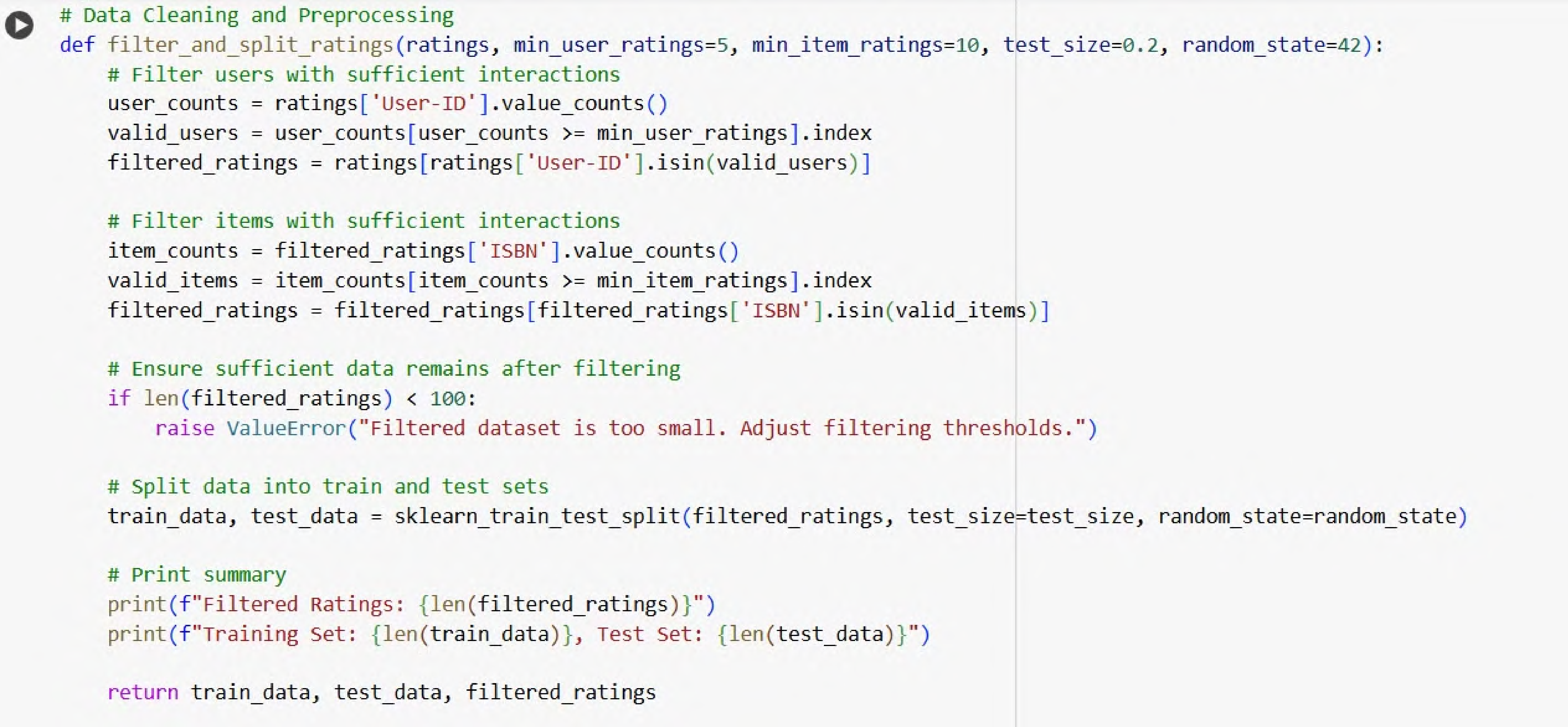
```
Ratings Summary:
              User-ID        ISBN    Book-Rating
count   433671.000000      433671  433671.000000
unique            NaN      185973            NaN
top               NaN  0316666343            NaN
freq              NaN         707            NaN
mean    135458.743451         NaN       7.601066
std      80678.385078         NaN       1.843798
min          8.000000         NaN       1.000000
25%      66619.000000         NaN       7.000000
50%     133184.000000         NaN       8.000000
75%     205735.000000         NaN       9.000000
max     278854.000000         NaN      10.000000
```

Data Preprocessing

- □ Importance of data preprocessing.
- □ Steps undertaken:
  - Handling missing values.
  - Data filtering.
  - Data splitting.



```
Filtered Ratings: 96234
Training Set: 76987, Test Set: 19247
```



```python
# Data Cleaning and Preprocessing
def filter_and_split_ratings(ratings, min_user_ratings=5, min_item_ratings=10, test_size=0.2, random_state=42):
    # Filter users with sufficient interactions
    user_counts = ratings['User-ID'].value_counts()
    valid_users = user_counts[user_counts >= min_user_ratings].index
    filtered_ratings = ratings[ratings['User-ID'].isin(valid_users)]

    # Filter items with sufficient interactions
    item_counts = filtered_ratings['ISBN'].value_counts()
    valid_items = item_counts[item_counts >= min_item_ratings].index
    filtered_ratings = filtered_ratings[filtered_ratings['ISBN'].isin(valid_items)]

    # Ensure sufficient data remains after filtering
    if len(filtered_ratings) < 100:
        raise ValueError("Filtered dataset is too small. Adjust filtering thresholds.")

    # Split data into train and test sets
    train_data, test_data = sklearn_train_test_split(filtered_ratings, test_size=test_size, random_state=random_state)

    # Print summary
    print(f"Filtered Ratings: {len(filtered_ratings)}")
    print(f"Training Set: {len(train_data)}, Test Set: {len(test_data)}")

    return train_data, test_data, filtered_ratings
```

Handling Missing Values

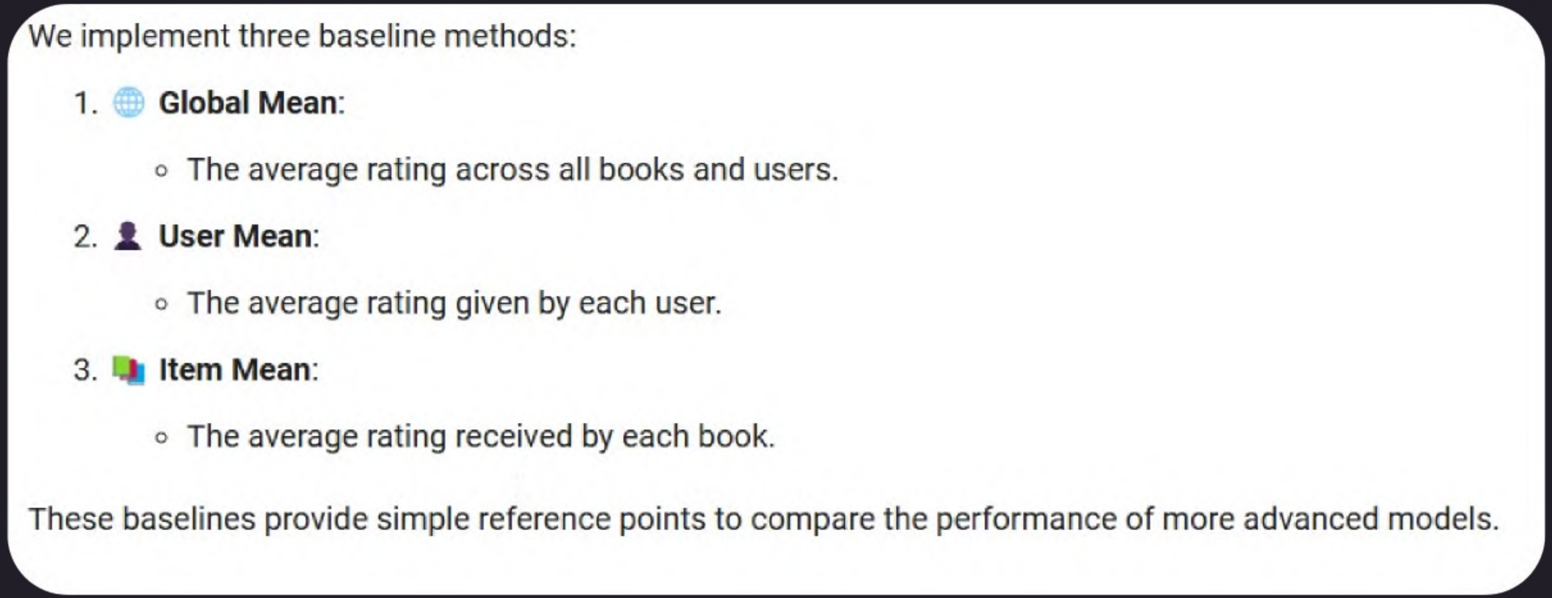
□ **Users Dataset**:
  - Imputed missing ages by sampling valid ages.
  - Clipped ages to 10-100 years.

□ **Books Dataset**:
  - 'Year-Of-Publication': Filled invalid years with 'Unknown'.
  - 'Book-Author' and 'Publisher': Filled with 'Unknown'.
  - 'Image-URL-L': Filled with 'No Image'.

□ **Ratings Dataset**:
  - Removed ratings with a value of 0.

Data Filtering and Splitting

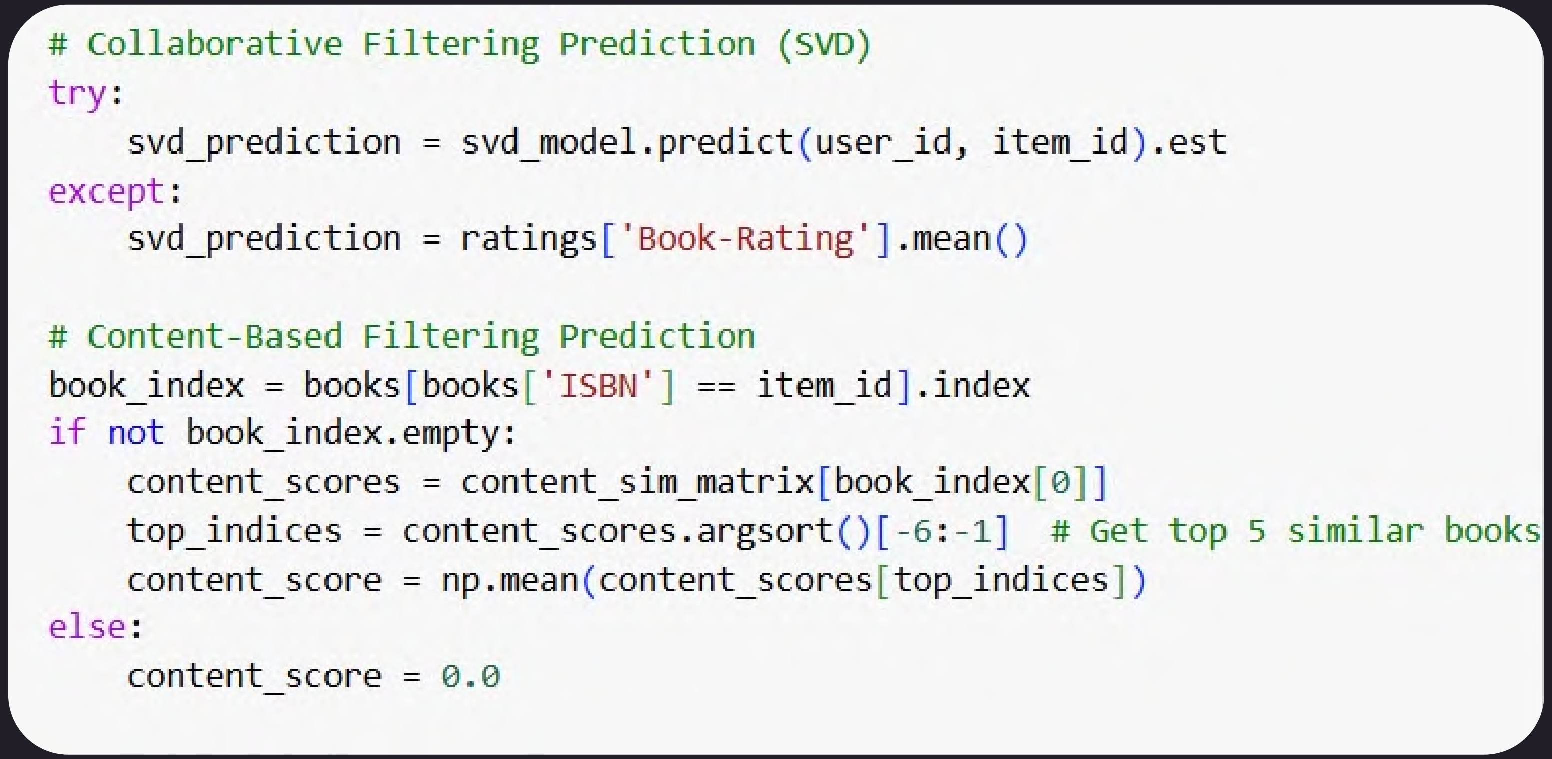
- □ **Data Filtering**:
  - Retained users with at least 5 ratings.
  - Retained books with at least 10 ratings.
- □ **Data Splitting**:
  - 80/20 split for training and testing.
  - Ensures models are evaluated on unseen data.
- □ **Resulting Data Sizes**:
  - Filtered Ratings: 96,234 entries.
  - Training Set: 76,987 entries.
  - Testing Set: 19,247 entries.



We implement three baseline methods:

1. 🌐 **Global Mean**:
   - The average rating across all books and users.
2. 👤 **User Mean**:
   - The average rating given by each user.
3. 📚 **Item Mean**:
   - The average rating received by each book.

These baselines provide simple reference points to compare the performance of more advanced models.

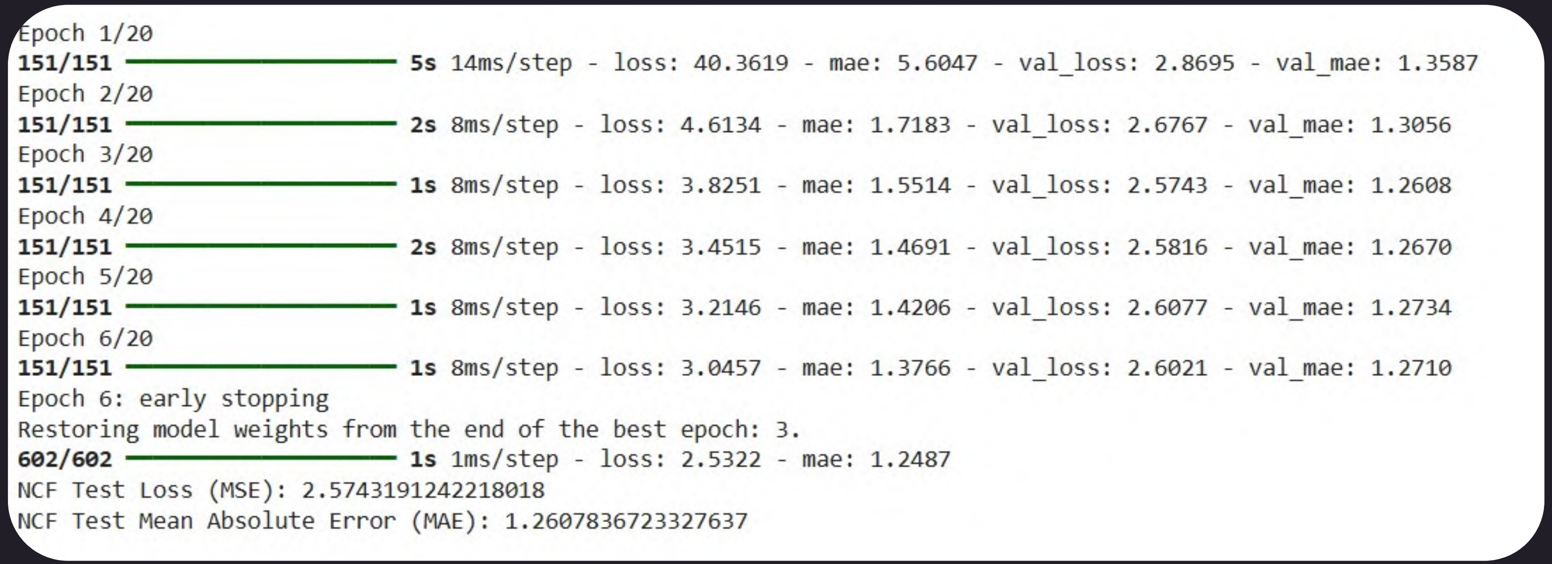


```
# Collaborative Filtering Prediction (SVD)
try:
    svd_prediction = svd_model.predict(user_id, item_id).est
except:
    svd_prediction = ratings['Book-Rating'].mean()

# Content-Based Filtering Prediction
book_index = books[books['ISBN'] == item_id].index
if not book_index.empty:
    content_scores = content_sim_matrix[book_index[0]]
    top_indices = content_scores.argsort()[-6:-1]  # Get top 5 similar books
    content_score = np.mean(content_scores[top_indices])
else:
    content_score = 0.0
```

Methodology Overview

Overview of recommendation techniques implemented:

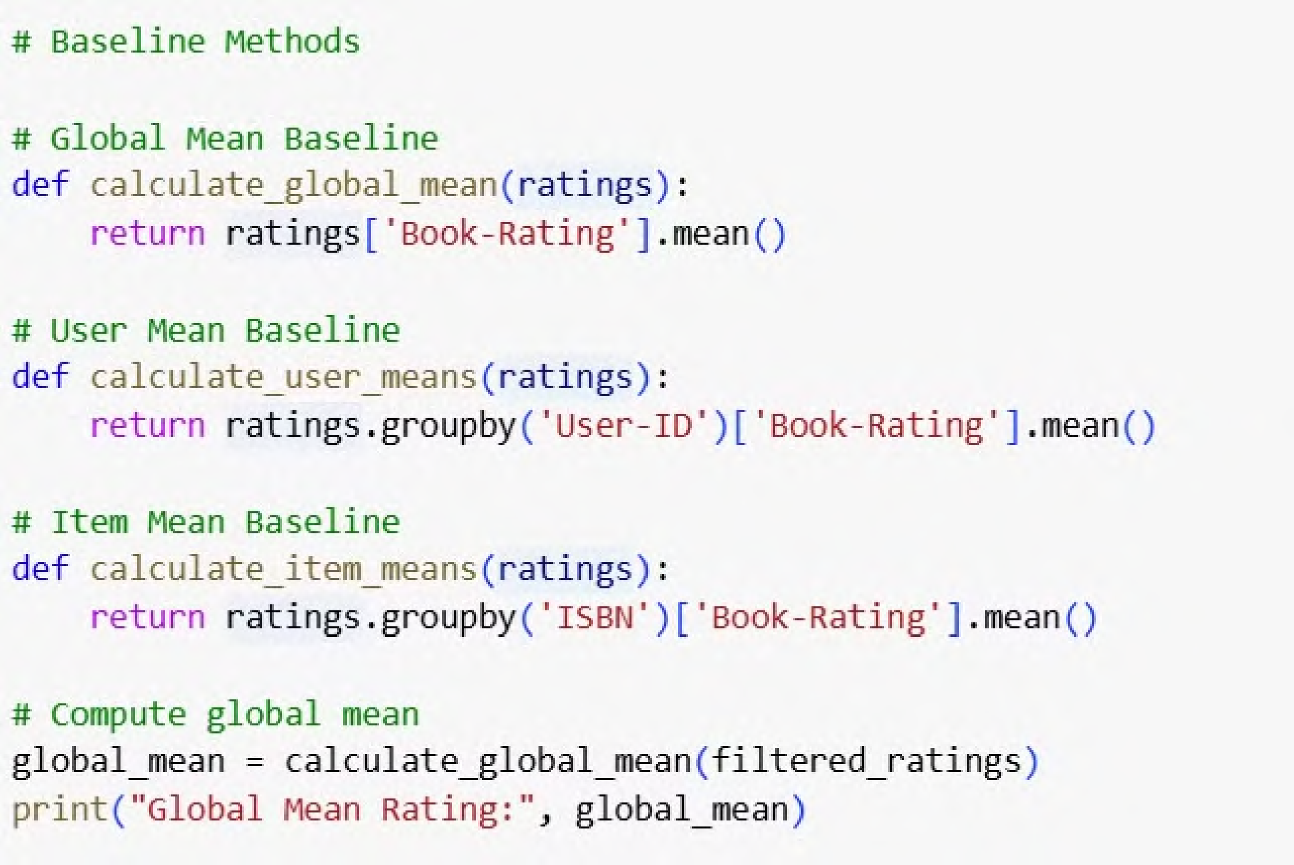
- Baseline Methods
- Content-Based Filtering
- Collaborative Filtering (SVD)
- Neural Collaborative Filtering (NCF)
- Hybrid Recommendation System



```
Epoch 1/20
151/151 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - loss: 40.3619 - mae: 5.6047 - val_loss: 2.8695 - val_mae: 1.3587
Epoch 2/20
151/151 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - loss: 4.6134 - mae: 1.7183 - val_loss: 2.6767 - val_mae: 1.3056
Epoch 3/20
151/151 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 3.8251 - mae: 1.5514 - val_loss: 2.5743 - val_mae: 1.2608
Epoch 4/20
151/151 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - loss: 3.4515 - mae: 1.4691 - val_loss: 2.5816 - val_mae: 1.2670
Epoch 5/20
151/151 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 3.2146 - mae: 1.4206 - val_loss: 2.6077 - val_mae: 1.2734
Epoch 6/20
151/151 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 3.0457 - mae: 1.3766 - val_loss: 2.6021 - val_mae: 1.2710
Epoch 6: early stopping
Restoring model weights from the end of the best epoch: 3.
602/602 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 2.5322 - mae: 1.2487
NCF Test Loss (MSE): 2.5743191242218018
NCF Test Mean Absolute Error (MAE): 1.2607836723327637
```

Baseline Methods

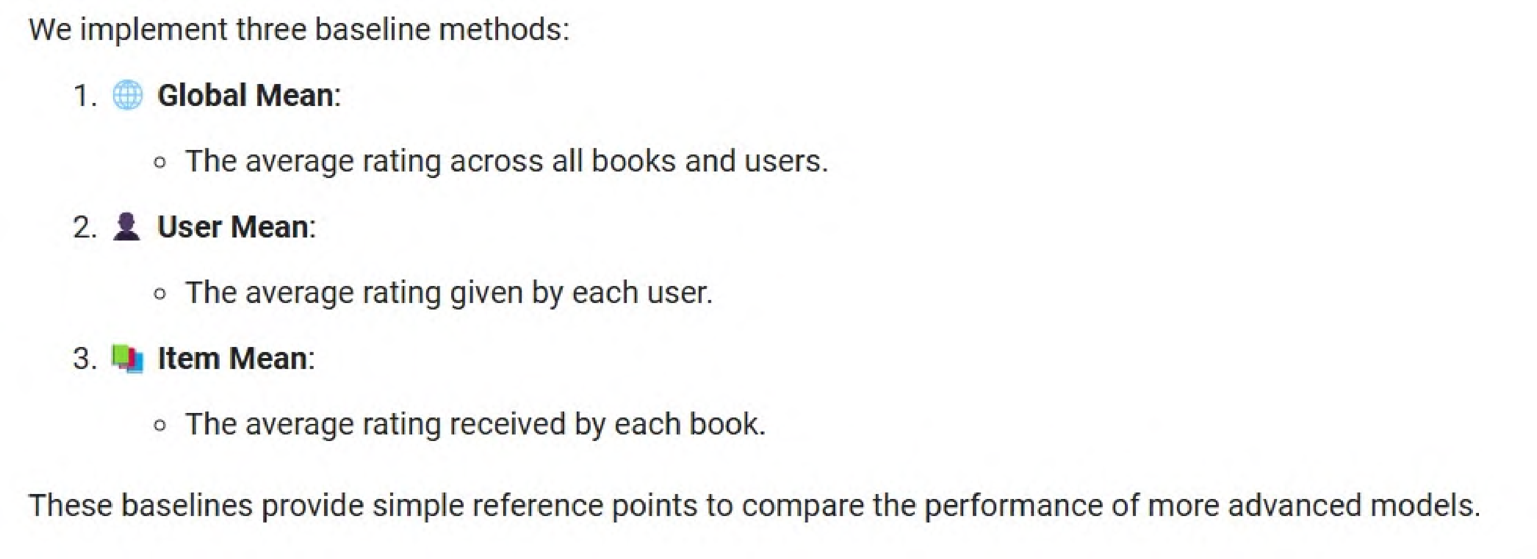
- □ Purpose: Provide reference points for model performance.
- □ Methods:
  - Global Mean
  - User Mean
  - Item Mean



```
# Baseline Methods

# Global Mean Baseline
def calculate_global_mean(ratings):
    return ratings['Book-Rating'].mean()

# User Mean Baseline
def calculate_user_means(ratings):
    return ratings.groupby('User-ID')['Book-Rating'].mean()

# Item Mean Baseline
def calculate_item_means(ratings):
    return ratings.groupby('ISBN')['Book-Rating'].mean()

# Compute global mean
global_mean = calculate_global_mean(filtered_ratings)
print("Global Mean Rating:", global_mean)
```

Baseline Methods Results

- □ **Global Mean Rating: 7.8236**
- □ **User Mean RMSE: 1.6774**
- □ **Item Mean RMSE: 1.7476**



We implement three baseline methods:

1. 🌐 **Global Mean**:
   - The average rating across all books and users.
2. 👤 **User Mean**:
   - The average rating given by each user.
3. 📚 **Item Mean**:
   - The average rating received by each book.

These baselines provide simple reference points to compare the performance of more advanced models.

Content-Based Filtering

□ Recommends items similar to those liked by the user.

□ Based on item features.

Approach

□ **TF-IDF Vectorization**:

- Converts text into numerical features.
- Captures importance of words in titles.

□ **Cosine Similarity**:

- Measures similarity b*etween books.*
- *Based on TF-IDF vect*ors.

□ **Recommendation Function**:

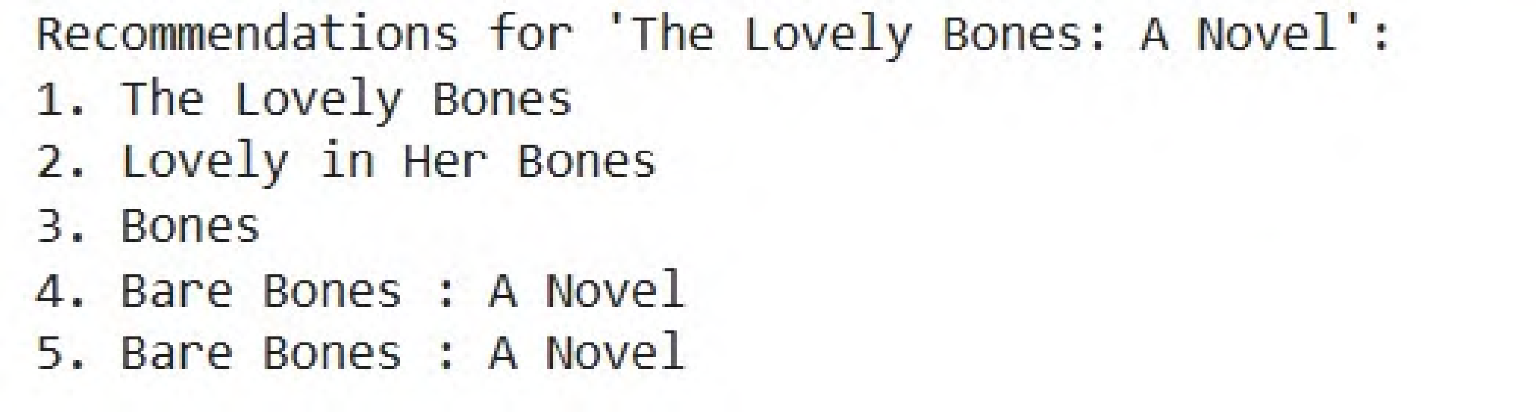
- Inputs a book title.
- Outputs top N similar books.

Example

□ **Input Book**: "The Lovely Bones: A Novel"

□ **Recommendations**:

1. "The Lovely Bones"
2. "Lovely in Her Bones"
3. "Bones"
4. "Bare Bones: A Novel"



```
Recommendations for 'The Lovely Bones: A Novel':
1. The Lovely Bones
2. Lovely in Her Bones
3. Bones
4. Bare Bones : A Novel
5. Bare Bones : A Novel
```

Collaborative Filtering (SVD)

□ Predicts user preferences based on past interactions.

□ Utilizes patterns in user-item interactions.

SVD Approach

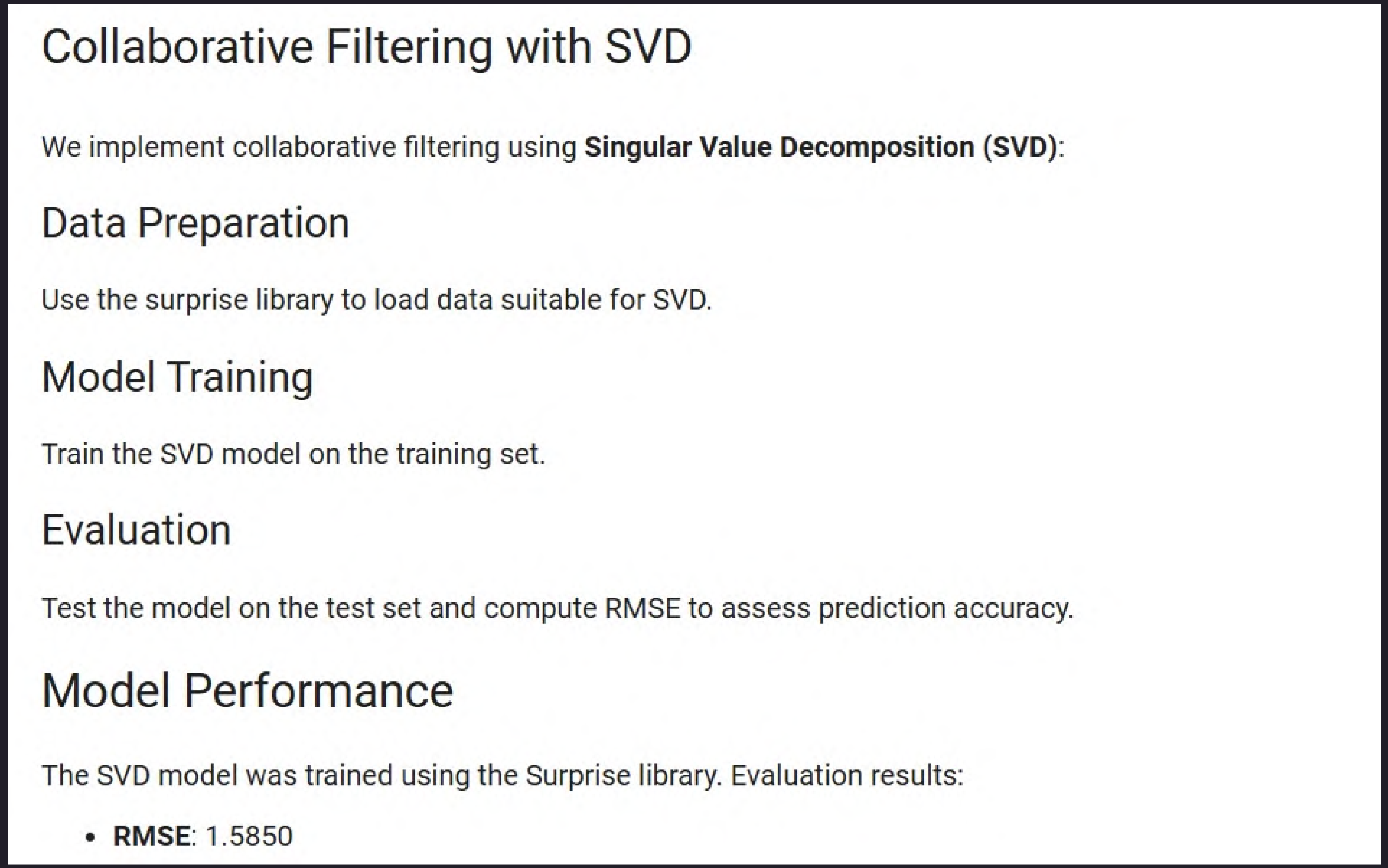
□ **Data Preparation**:
- Used the Surprise library.
- Loaded data suitable for SVD.

□ **Model Training**:
- Trained SVD with 100 latent factors.
- Over 30 epochs.

□ **Evaluation**:
- Tested on the test set.
- Computed RMSE

SVD Performance

SVD RMSE: 1.5828



Collaborative Filtering with SVD

We implement collaborative filtering using **Singular Value Decomposition (SVD)**:

Data Preparation

Use the surprise library to load data suitable for SVD.

Model Training

Train the SVD model on the training set.

Evaluation

Test the model on the test set and compute RMSE to assess prediction accuracy.

Model Performance

The SVD model was trained using the Surprise library. Evaluation results:

- **RMSE**: 1.5850

Neural Collaborative Filtering (NCF)

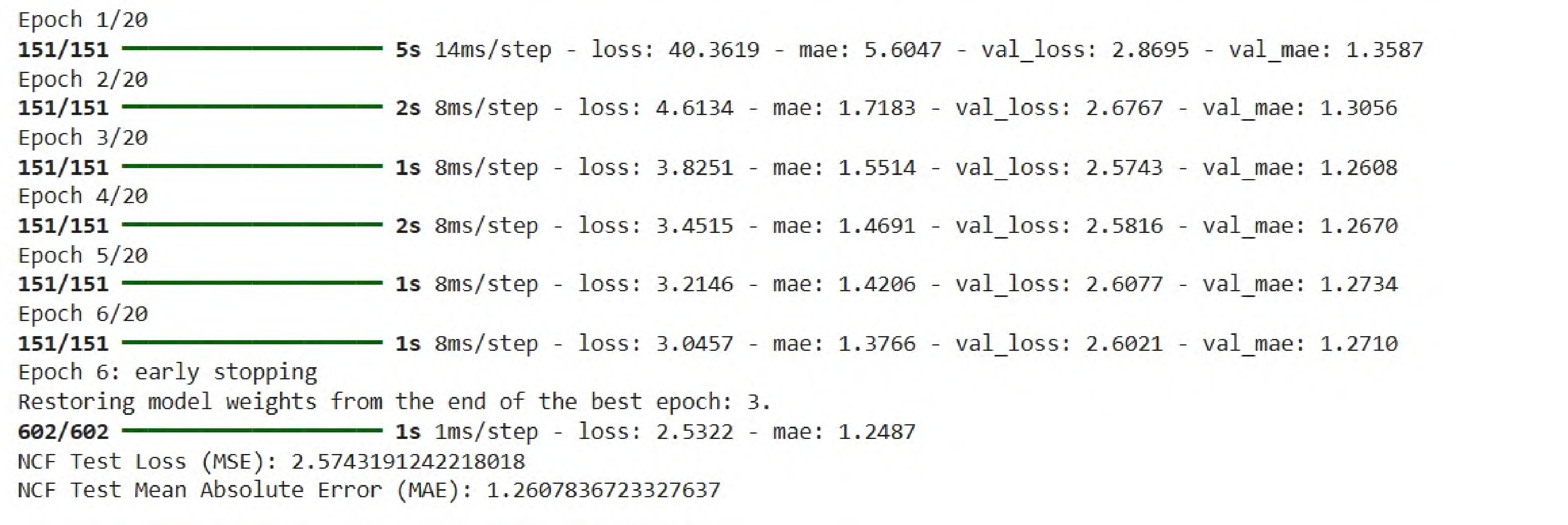
- □ Uses neural networks to model complex interactions.
- □ Captures non-linear relationships.

NCF Performance

- □ **Test Loss (MSE): 2.5271**
- □ **Test MAE: 1.2402**
- □ **NCF RMSE: 1.5897**

NCF Approach

- □ **Data Preprocessing**:
  - Encoded user and item IDs.
  - Split data into training and testing sets.
- □ **Model Architecture**:
  - Embedding layers for users and items.
  - Dense layers to capture interactions.
  - Output layer predicts the rating.
- □ **Training**:
  - Used early stopping to prevent overfitting.



```
Epoch 1/20
151/151 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - loss: 40.3619 - mae: 5.6047 - val_loss: 2.8695 - val_mae: 1.3587
Epoch 2/20
151/151 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - loss: 4.6134 - mae: 1.7183 - val_loss: 2.6767 - val_mae: 1.3056
Epoch 3/20
151/151 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 3.8251 - mae: 1.5514 - val_loss: 2.5743 - val_mae: 1.2608
Epoch 4/20
151/151 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - loss: 3.4515 - mae: 1.4691 - val_loss: 2.5816 - val_mae: 1.2670
Epoch 5/20
151/151 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 3.2146 - mae: 1.4206 - val_loss: 2.6077 - val_mae: 1.2734
Epoch 6/20
151/151 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 3.0457 - mae: 1.3766 - val_loss: 2.6021 - val_mae: 1.2710
Epoch 6: early stopping
Restoring model weights from the end of the best epoch: 3.
602/602 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 2.5322 - mae: 1.2487
NCF Test Loss (MSE): 2.5743191242218018
NCF Test Mean Absolute Error (MAE): 1.2607836723327637
```

Hybrid Recommendation System

□ Combines collaborative and content-based filtering.

□ Leverages strengths of both methods.

Hybrid Performance

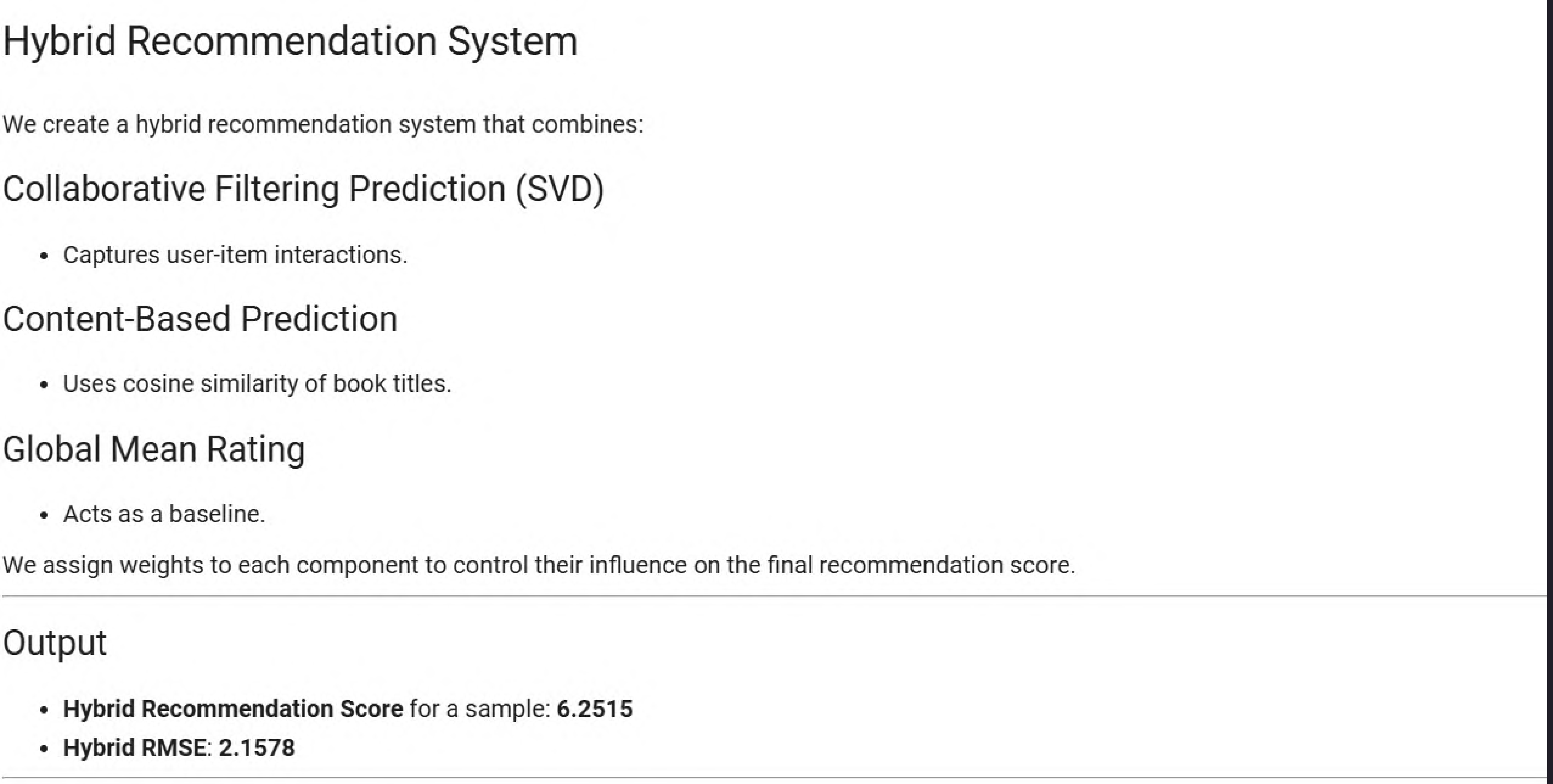
□ **Sample Hybrid Score: 6.3660**

□ **Hybrid RMSE: 2.1550**

Hybrid Approach

□ **Components**:
- SVD Prediction (Weight: 0.7)
- Content-Based Prediction (Weight: 0.2)
- Global Mean Rating (Weight: 0.1)

□ **Hybrid Score**:
- Weighted sum of the components.



Hybrid Recommendation System

We create a hybrid recommendation system that combines:

Collaborative Filtering Prediction (SVD)

- Captures user-item interactions.

Content-Based Prediction

- Uses cosine similarity of book titles.

Global Mean Rating

- Acts as a baseline.

We assign weights to each component to control their influence on the final recommendation score.

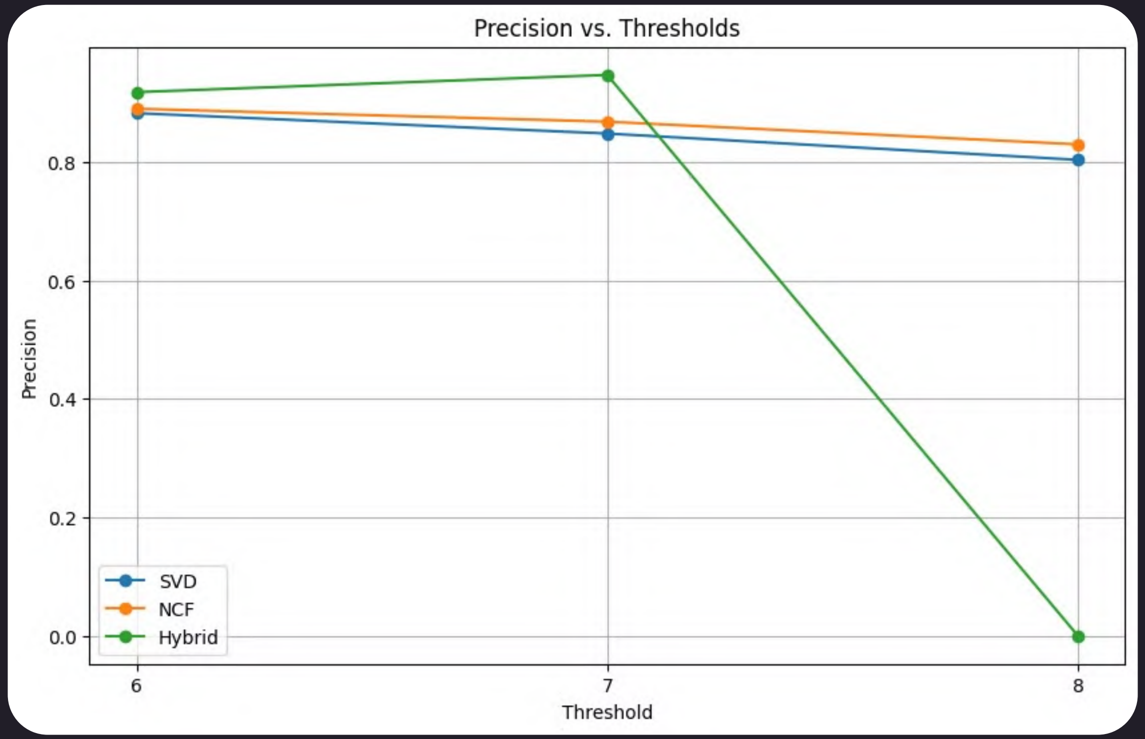
Output

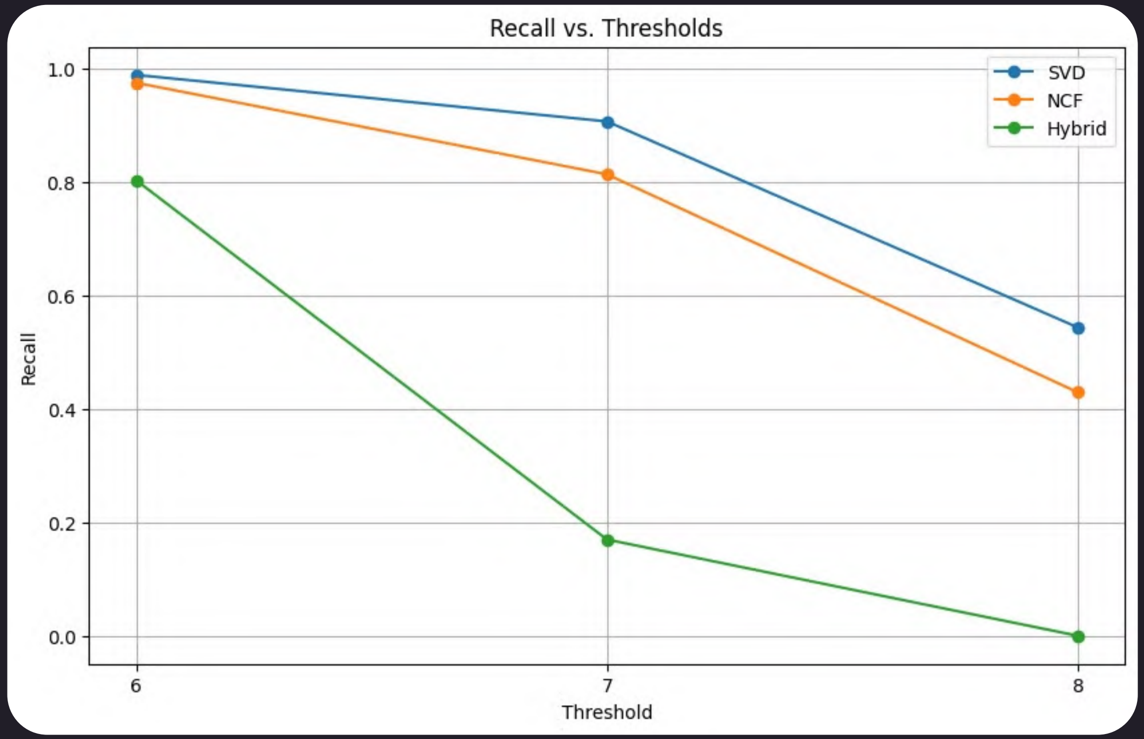
- **Hybrid Recommendation Score** for a sample: **6.2515**
- **Hybrid RMSE**: **2.1578**

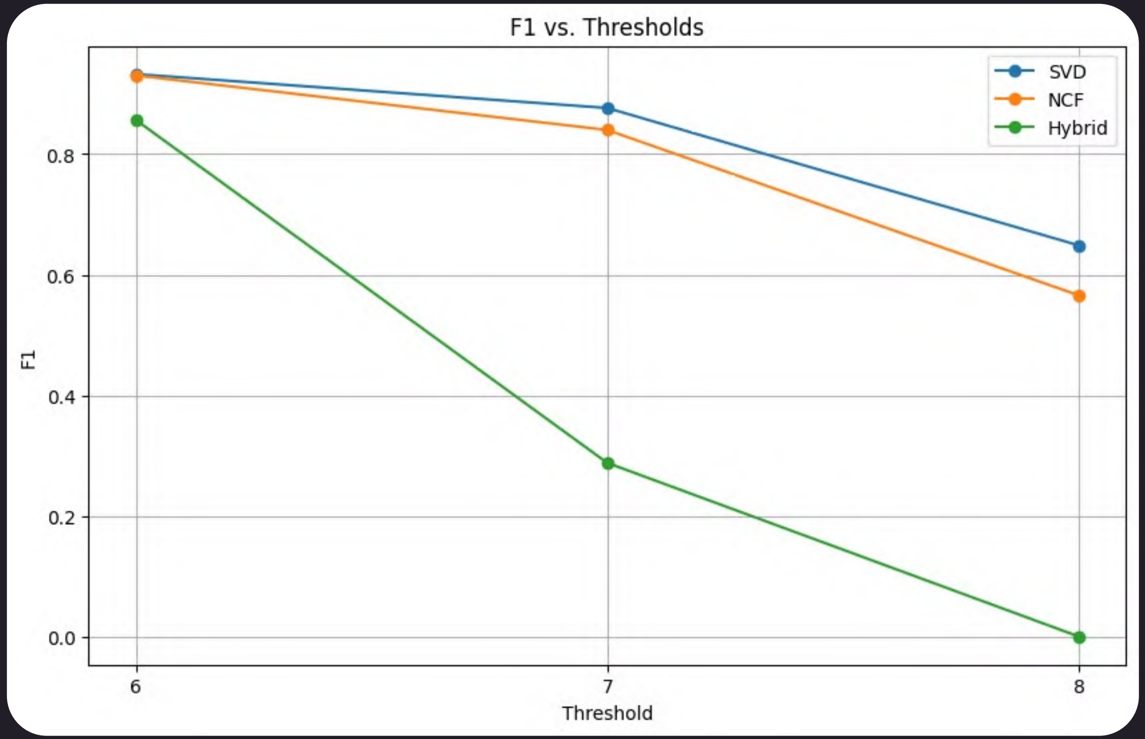
Try Pitch

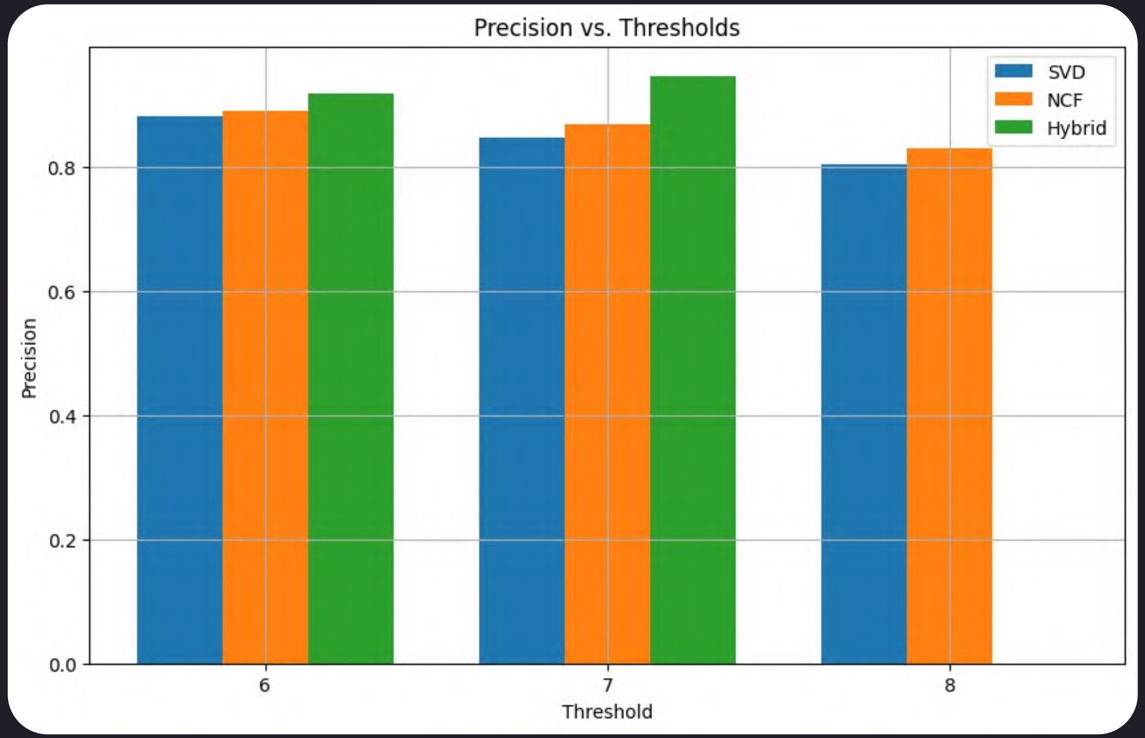
RMSE Comparison

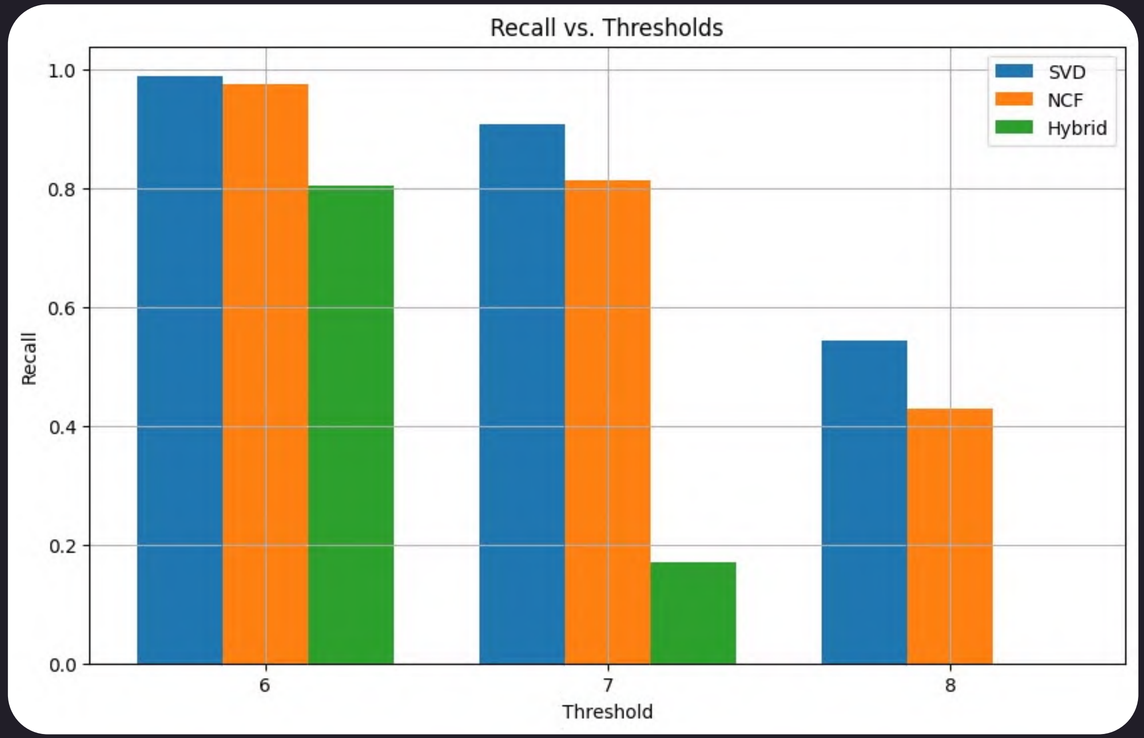
Model	RMSE
SVD	1.5828
NCF	1.5897
Hybrid	2.1550
User Mean Baseline	1.6774
Item Mean Baseline	1.7476

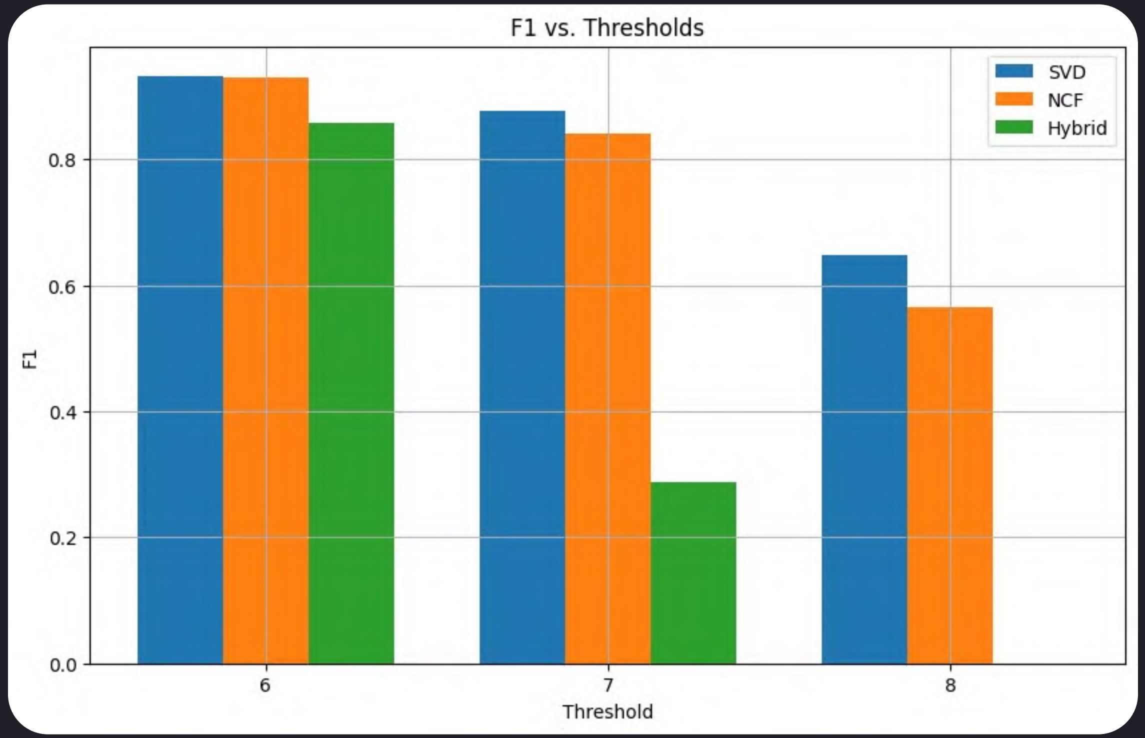

Precision vs. Thresholds
Precision
0.0
0.2
0.4
0.6
0.8
6
7
8
Threshold
SVD
NCF
Hybrid


Recall vs. Thresholds
Recall
0.0
0.2
0.4
0.6
0.8
1.0
6
7
8
Threshold
SVD
NCF
Hybrid


F1 vs. Thresholds
F1
0.0
0.2
0.4
0.6
0.8
6
7
8
Threshold
SVD
NCF
Hybrid

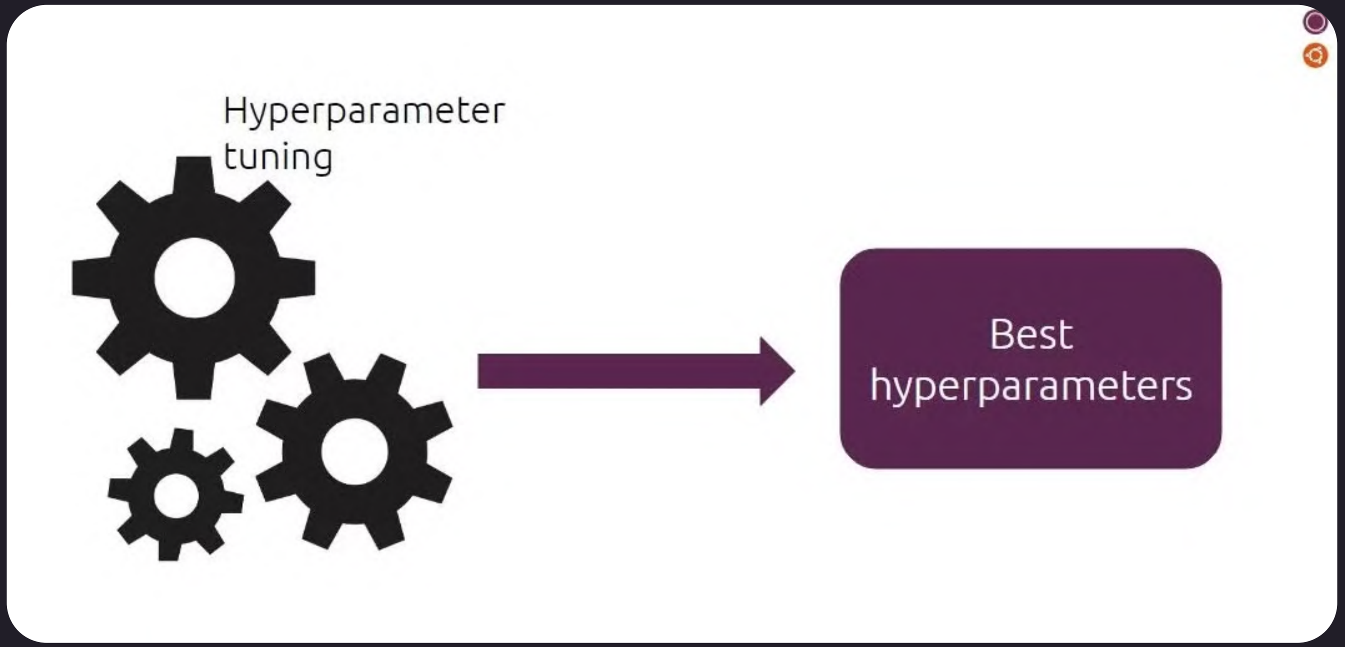

Precision vs. Thresholds
Precision
0.0
0.2
0.4
0.6
0.8
6
7
8
Threshold
SVD
NCF
Hybrid

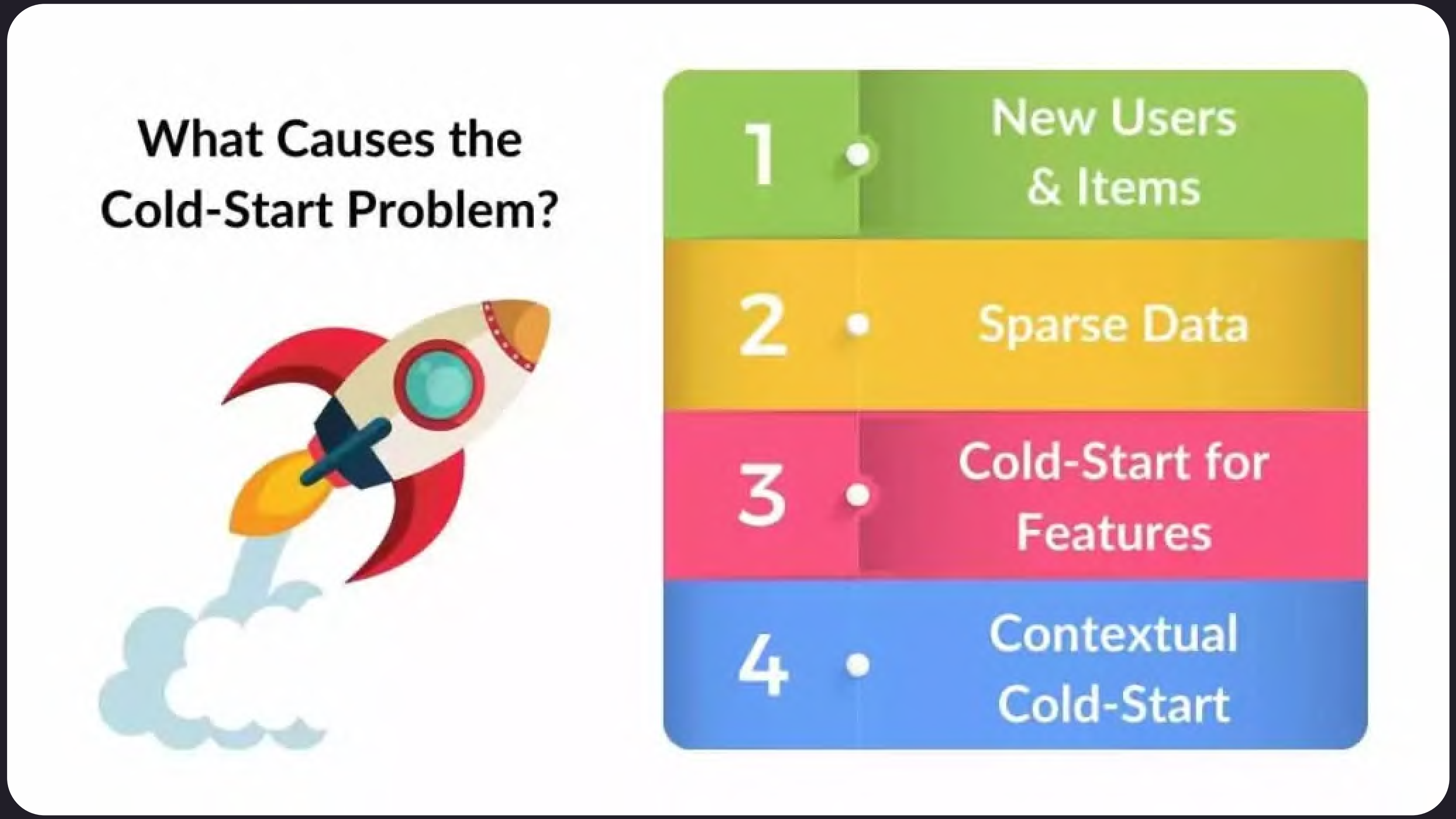

Recall vs. Thresholds
Recall
0.0
0.2
0.4
0.6
0.8
1.0
6
7
8
Threshold
SVD
NCF
Hybrid


F1 vs. Thresholds
F1
0.0
0.2
0.4
0.6
0.8
6
7
8
Threshold
SVD
NCF
Hybrid

Challenges

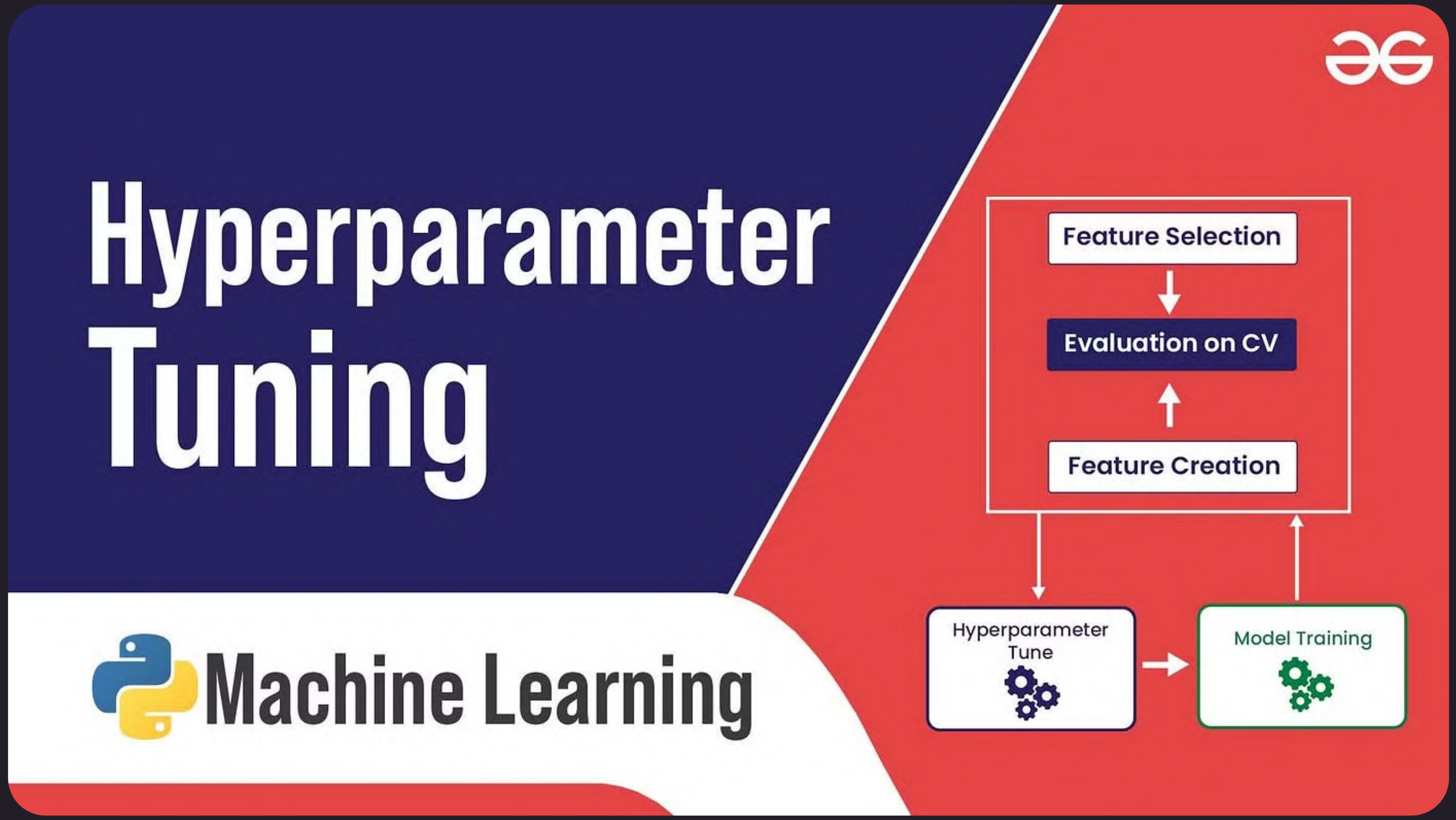
- □ **Data Sparsity**:
  - Limited ratings per user/book.
  - Affects model learning.
- □ **Cold-Start Problem**:
  - New users/items lack historical data.
- □ **Computational Complexity**:
  - NCF requires more resources.
- □ **Evaluation Trade-Offs**:
  - Balancing precision and recall is crucial.

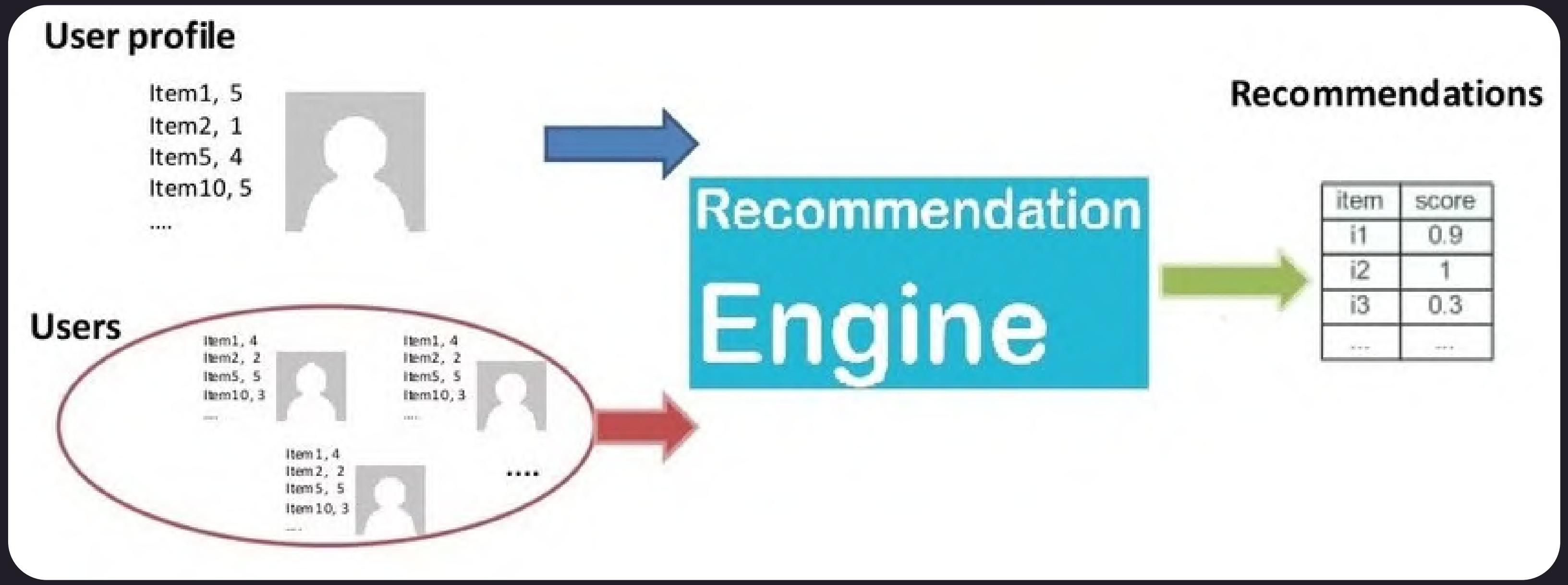



What Causes the Cold-Start Problem?
1 New Users & Items
2 Sparse Data
3 Cold-Start for Features
4 Contextual Cold-Start

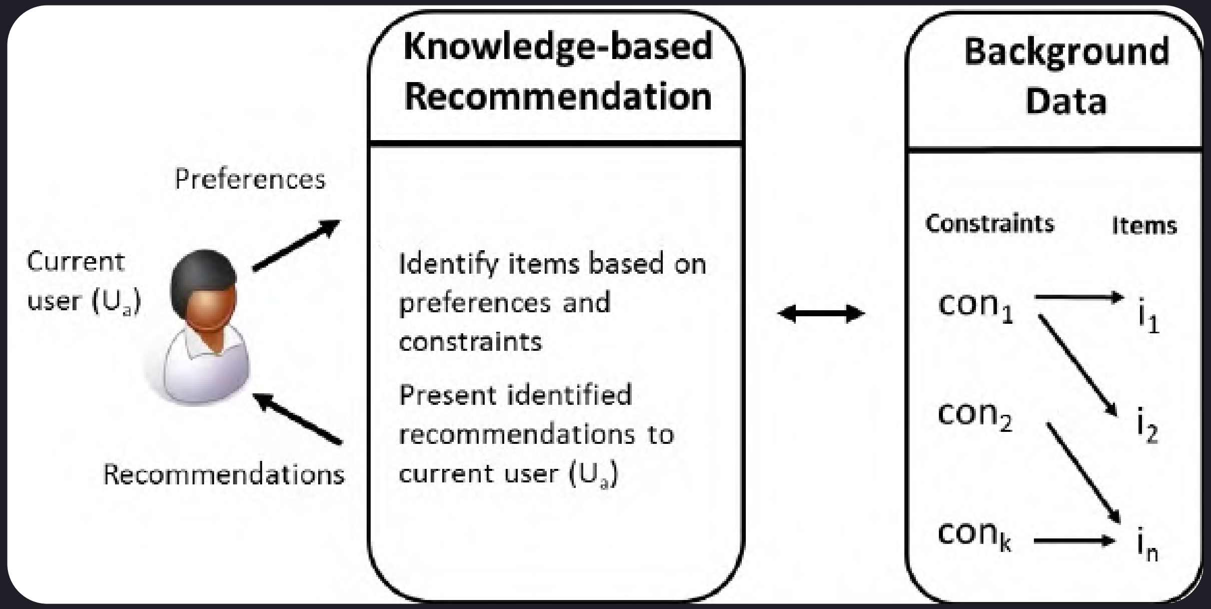
Future Work

- **Hyperparameter Tuning**:
  - Optimize NCF and hybrid models.
- **Hybrid Model Adjustment**:
  - Rebalance weights to improve recall.
- **Advanced Techniques**:
  - Explore other algorithms like KB recommendations.





item	score
i1	0.9
i2	1
i3	0.3
...	...



Identify items based on preferences and constraints

Present identified recommendations to current user (U_a)

Conclusion

Successfully implemented and evaluated various models.

SVD model was most effective.

NCF model showed promise.

Hybrid model needs adjustment.

Thank You For Listening!

Now any questions?

References

- □ Book-Crossing Dataset: Kaggle
- □ Surprise Library Documentation: Surprise
- □ TensorFlow Keras Documentation: TensorFlow
- □ Scikit-learn Documentation: Scikit-learn
- □ Pandas Documentation: Pandas