Book Recommendation System

Project Members:

- Arsalan Khan
- Waleed Ghufran
- Spencer Robson
- Asil Khan
- Aliha Ali
- Qusai Almasri

Table of Contents

---

1. Introduction

In the era of information overload, recommendation systems have become essential tools for guiding users toward relevant content. This project focuses on developing a comprehensive book recommendation system using the Book-Crossing dataset from Kaggle. The primary objectives are:

- To implement various recommendation approaches.
- To compare their performance using appropriate evaluation metrics.
- To enhance the system's accuracy by incorporating advanced methods.

We explore several recommendation techniques, including baseline methods, content-based filtering, collaborative filtering, neural collaborative filtering, and hybrid recommendation systems. Each method is evaluated to determine its effectiveness in predicting user preferences and providing meaningful recommendations.

---

2. Dataset Description

The Book-Crossing dataset is a comprehensive collection of user ratings for a vast array of books. It includes three primary files:

- **Users.csv**: Contains user demographic information.
- **Books.csv**: Contains book metadata.
- **Ratings.csv**: Contains user ratings for books.

Dataset Statistics:

- **Users dataset**: 278,858 entries.
- **Books dataset**: 271,360 entries.
- **Ratings dataset**: 1,149,780 entries.

The ratings are on a scale from 1 to 10, with higher values indicating stronger user preferences.

---

3. Data Preprocessing

Data preprocessing is crucial to ensure the quality and reliability of the recommendation system. The following steps were undertaken:

3.1 Handling Missing Values

Users Dataset:

- **Age**: Missing values were imputed by randomly sampling from the distribution of valid ages.
- **Age Clipping**: Ages were clipped to a reasonable range (10 to 100 years).

Books Dataset:

- **Year-Of-Publication**: Non-numeric and invalid years were coerced to `NaN` and filled with the placeholder "Unknown" to retain records.
- **Book-Author** and **Publisher**: Missing values were filled with "Unknown".
- **Image-URL-L**: Missing values were filled with "No Image".

Ratings Dataset:

- **Zero Ratings**: Ratings with a value of 0 were filtered out, as they do not contribute to the recommendation process.

3.2 Data Filtering

To focus on users and items with sufficient data:

- **User Filtering**: Retained users with at least **5 ratings**.
- **Item Filtering**: Retained books with at least **10 ratings**.

3.3 Data Splitting

The filtered dataset was split into training and testing sets using an 80/20 split, ensuring that model evaluations are conducted on unseen data.

Resulting Data Sizes:

- **Filtered Ratings**: 96,234 entries.
- **Training Set**: 76,987 entries.
- **Testing Set**: 19,247 entries.

---

4. Methodology

We implemented various recommendation techniques to evaluate their performance and effectiveness.

4.1 Baseline Methods

Baseline models provide simple reference points to compare more advanced methods.

4.1.1 Global Mean

- **Description**: The average rating across all books and users.
- **Global Mean Rating**: **7.8236**

4.1.2 User Mean

- **Description**: The average rating given by each user.
- **RMSE**: **1.6774**

4.1.3 Item Mean

- **Description**: The average rating received by each book.
- **RMSE**: **1.7476**

4.2 Content-Based Filtering

Content-based filtering recommends items similar to those a user has liked in the past, based on item features.

4.2.1 Approach

- **TF-IDF Vectorization**: Converted book titles into TF-IDF features to capture the importance of words.
- **Cosine Similarity**: Calculated the similarity between books based on their TF-IDF vectors.
- **Recommendation Function**: Given a book title, the system recommends the top N similar books.

4.2.2 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"*
    5. *"Bare Bones: A Novel"*

4.3 Collaborative Filtering (SVD)

Collaborative filtering predicts user preferences based on past interactions.

4.3.1 Singular Value Decomposition (SVD)

- **Data Preparation**: Used the Surprise library to load data suitable for SVD.
- **Model Training**: Trained the SVD model on the training set with 100 latent factors over 30 epochs.
- **Evaluation**: Tested the model on the test set and computed RMSE.

4.3.2 Performance

- **RMSE**: **1.5828**

4.4 Neural Collaborative Filtering (NCF)

NCF uses neural networks to model complex user-item interactions.

4.4.1 Approach

- **Data Preprocessing**:
  - Encoded user and item IDs using `LabelEncoder`.
  - Split data into training and testing sets.
- **Hyperparameter Tuning**:
  - Implemented hyperparameter tuning using Keras Tuner to optimize model parameters.
  - Tuned hyperparameters included latent dimensions, number of dense units, dropout rate, and learning rate.
- **Model Architecture**:
  - **Embedding Layers**: For users and items.
  - **Dense Layers**: Captured nonlinear interactions.
  - **Output Layer**: Predicted the rating.
- **Training**:
  - Used early stopping to prevent overfitting.
  - Trained the model with optimal hyperparameters identified during tuning.

4.4.2 Performance

- **Optimal Hyperparameters**:
  - **Latent Dimension**: 40
  - **Dense Units**: 256
  - **Dropout Rate**: 0.1
  - **Learning Rate**: 0.001
- **Test Loss (MSE)**: **2.4980**
- **Test MAE**: **1.2087**
- **RMSE**: **1.5805**

4.5 Hybrid Recommendation System

Combines collaborative filtering and content-based filtering to leverage the strengths of both methods.

4.5.1 Approach

- **Components**:
  - **SVD Prediction**: Captures user-item interactions.
  - **Content-Based Prediction**: Uses cosine similarity of book titles.
  - **Global Mean Rating**: Acts as a baseline.

- **Weights**: Assigned to each component to control their influence (e.g., SVD: 0.7, Content: 0.2, Global: 0.1).
- **Hybrid Score**: A weighted sum of the above components.

4.5.2 Performance

- **Hybrid Recommendation Score (Sample)**: **6.1936**
- **Hybrid RMSE**: **2.1531**

---

5. Performance Evaluation

We evaluated the models using various metrics to understand their effectiveness.

5.1 Evaluation Metrics

- **RMSE (Root Mean Squared Error)**: Measures the average deviation between predicted and actual ratings.
- **Precision**: Proportion of recommended items that are relevant.
- **Recall**: Proportion of relevant items that are recommended.
- **F1 Score**: Harmonic mean of precision and recall.

5.2 Model Comparisons

5.2.1 RMSE Comparison

Model	RMSE
SVD	**1.5828**
NCF	**1.5805**
Hybrid	**2.1531**
User Mean Baseline	1.6774
Item Mean Baseline	1.7476

5.2.2 Precision, Recall, and F1 Score

Evaluation at different thresholds (ratings ≥ 6, 7, 8):

SVD Model

Threshold	Precision	Recall	F1 Score
≥6	0.8825	0.9901	0.9332
≥7	0.8471	0.9076	0.8763
≥8	0.7984	0.5407	0.6447

NCF Model

Threshold	Precision	Recall	F1 Score
≥6	0.8860	0.9858	0.9332
≥7	0.8594	0.8684	0.8639
≥8	0.8074	0.5472	0.6523

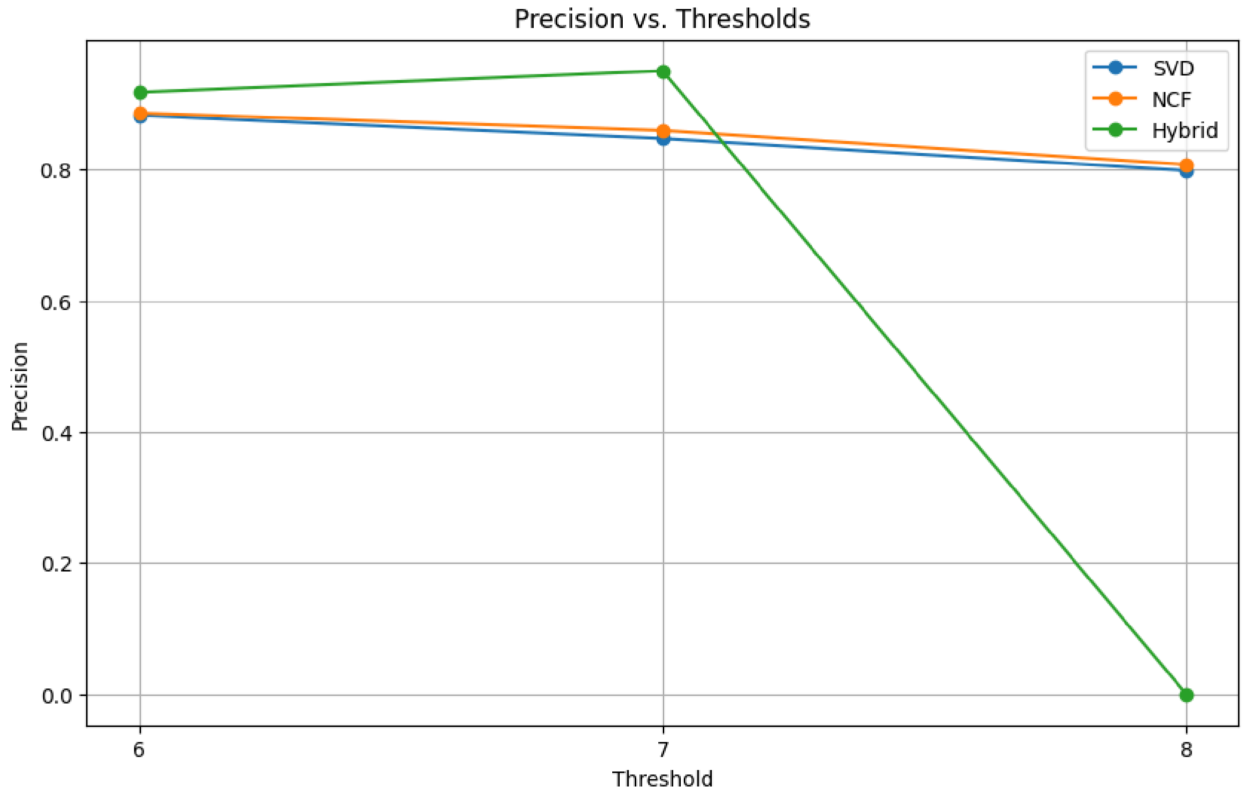
Hybrid Model

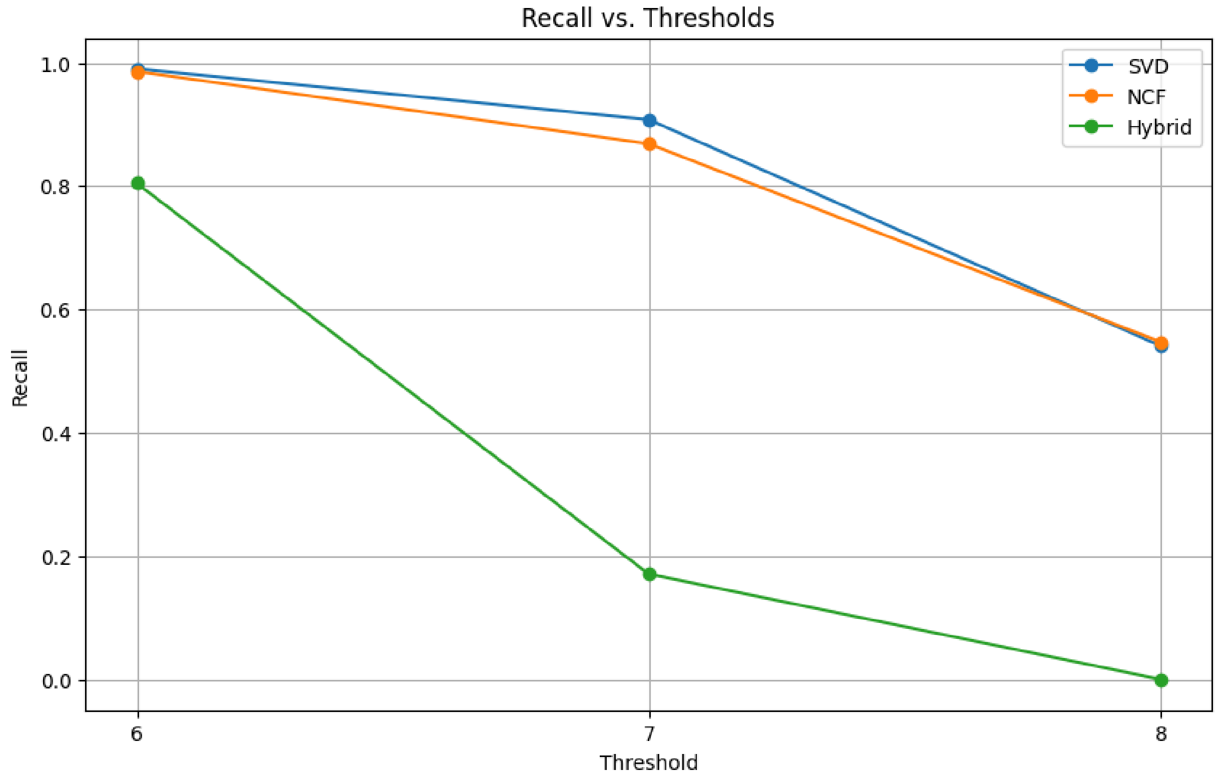
Threshold	Precision	Recall	F1 Score
≥6	0.9175	0.8044	0.8573
≥7	0.9503	0.1708	0.2895
≥8	0.0000	0.0000	0.0000

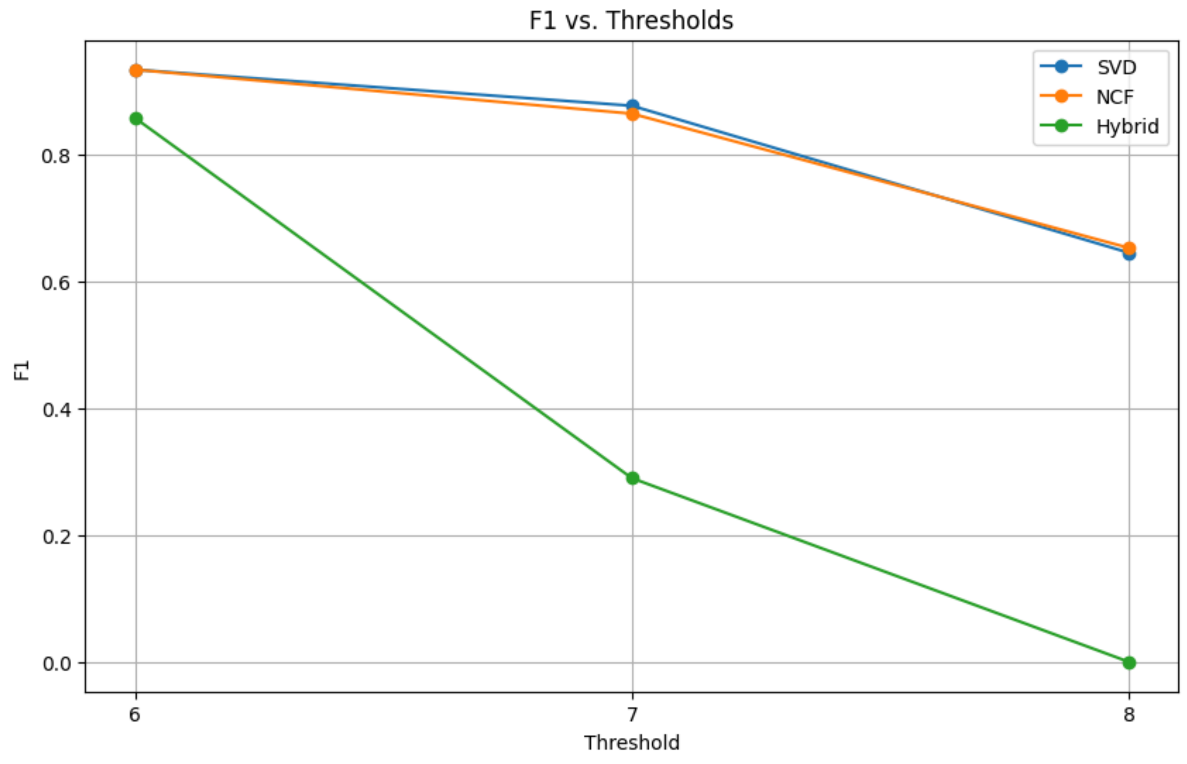
5.3 Graphical Insights

5.3.1 Line Plots

- **Precision vs. Thresholds**: Shows how precision changes with the threshold for each model.
- **Recall vs. Thresholds**: Illustrates the models' ability to capture relevant items at different thresholds.
- **F1 Score vs. Thresholds**: Provides a balanced view of precision and recall.

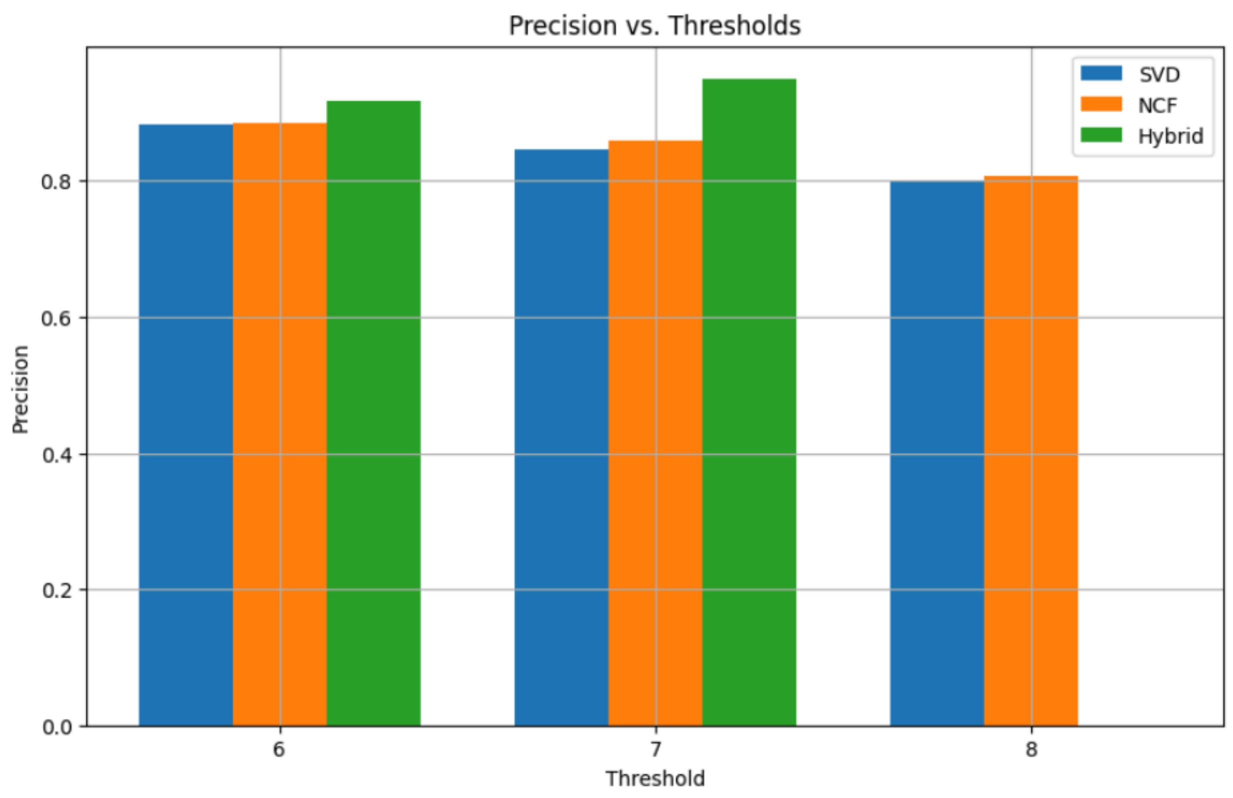

Precision vs. Thresholds
Precision
Threshold
0.0
0.2
0.4
0.6
0.8
6
7
8
SVD
NCF
Hybrid

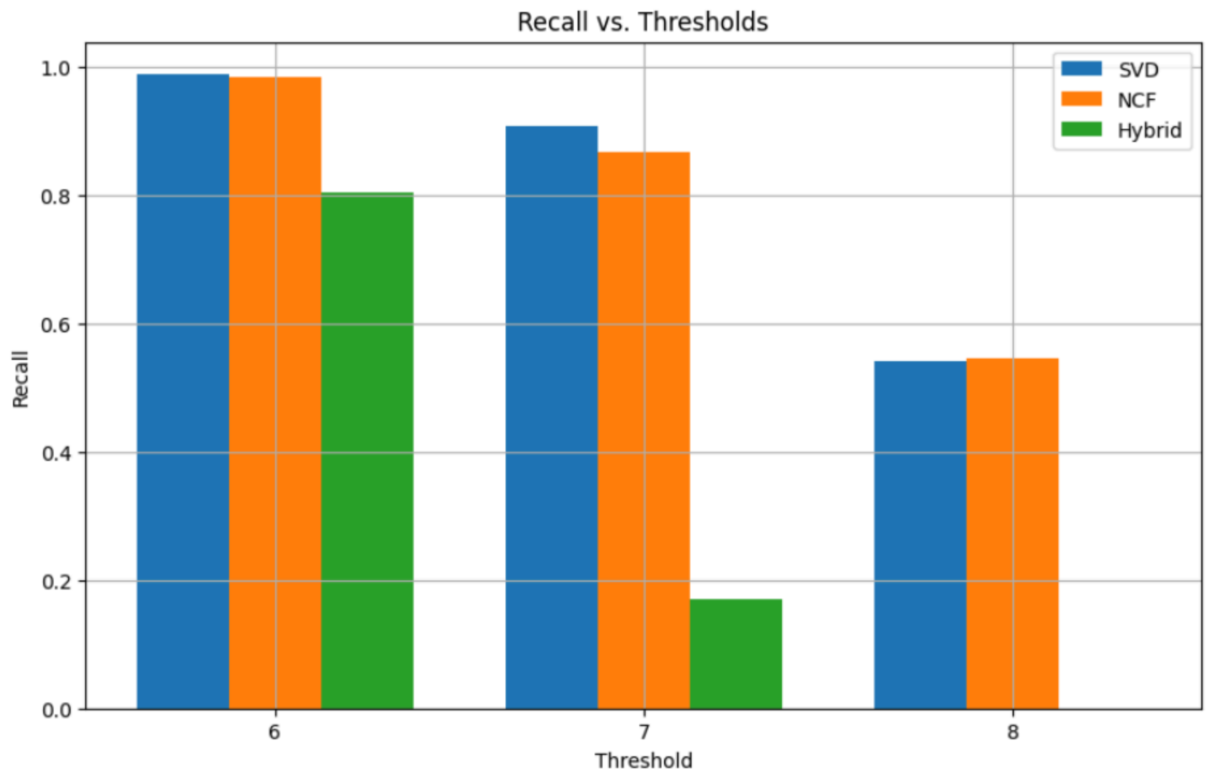

Recall vs. Thresholds
Recall
Threshold
0.0
0.2
0.4
0.6
0.8
1.0
6
7
8
SVD
NCF
Hybrid

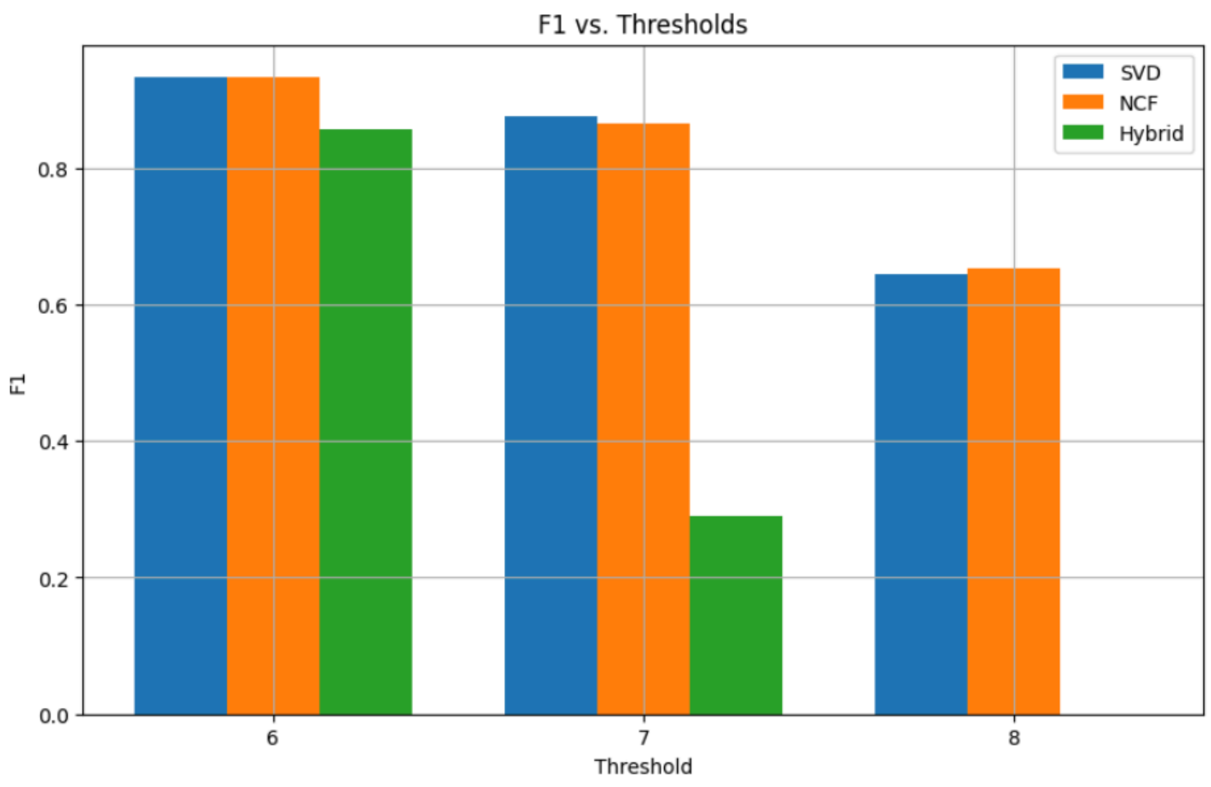


5.3.2 Bar Charts

- Visual representations of precision, recall, and F1 scores at each threshold for all models.
- Highlights the comparative performance across models and thresholds.


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


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



---

6. Discussion

6.1 Key Findings

- **SVD Model**:
  - *Initially* achieved the lowest RMSE (**1.5828**), indicating accurate predictions.
  - Demonstrated a good balance between precision and recall.
  - Effectively captured latent factors in user-item interactions.
- **NCF Model**:
  - After implementing hyperparameter tuning, the NCF model achieved the lowest RMSE of **1.5805**, surpassing the SVD model.
  - Exhibited slightly higher precision at lower thresholds.
  - Hyperparameter tuning enhanced the model's ability to capture complex, non-linear user-item interactions.
- **Hybrid Model**:
  - Adjusted the model weights to improve performance.
  - Despite adjustments, it still exhibited high precision but significantly lower recall at higher thresholds.
  - RMSE remained higher (**2.1531**) than both the SVD and NCF models, indicating the need for further refinement.

6.2 Interpretations

- **Impact of Hyperparameter Tuning**:
  - Hyperparameter tuning significantly improved the NCF model's performance.
  - It allowed the NCF model to outperform the SVD model, highlighting the importance of model optimization.
  - Demonstrated that advanced models like NCF can surpass traditional methods when properly tuned.
- **Model Performance**:
  - Both the SVD and NCF models outperform simple baseline methods in terms of RMSE and balanced evaluation metrics.
  - The NCF model, after tuning, provides the most accurate predictions, emphasizing the value of neural networks in capturing complex patterns.
- **Hybrid Model Adjustments**:
  - Adjusting the hybrid model's weights led to slight improvements.
  - However, it remains overly conservative with high precision but low recall.
  - Indicates that further adjustments or alternative approaches may be necessary for practical use.

6.3 Challenges and Considerations

- **Data Sparsity**:
  - The sparse nature of the dataset posed challenges, particularly for the NCF model.
  - Hyperparameter tuning helped mitigate some effects of data sparsity.
  - Many users have rated only a few books, limiting the models' ability to learn meaningful patterns.
- **Cold-Start Problem**:
  - New users or items with no historical data continue to present challenges for the recommendation system.
  - The models may struggle to make accurate recommendations for these users or items.
- **Computational Complexity**:
  - The NCF model required substantial computational resources and longer training times, especially during hyperparameter tuning.
  - Balancing computational cost and model performance is essential.
- **Evaluation Metrics Trade-Off**:
  - Achieving a balance between precision and recall remains crucial.
  - High precision with low recall, as seen in the hybrid model, may not be beneficial for users seeking diverse recommendations.
  - A balance is necessary to provide both accurate and comprehensive recommendations.

---

7. Conclusion

This project successfully developed and evaluated a comprehensive book recommendation system using the Book-Crossing dataset. The key accomplishments include:

- **Implementation of Various Models**:
    - Baseline methods provided foundational reference points.
    - Content-based filtering offered recommendations based on item features.
    - Collaborative filtering (SVD) effectively captured user-item interactions.
    - Neural collaborative filtering (NCF) initially performed comparably to the SVD model.
    - **After implementing hyperparameter tuning, the NCF model outperformed the SVD model**, achieving the lowest RMSE.
    - The hybrid model combined multiple approaches, and adjustments were made to its weights to enhance performance.
- **Performance Evaluation**:
    - The **NCF model emerged as the most effective**, achieving the lowest RMSE of **1.5805** and maintaining a good balance between precision and recall.
    - The SVD model remained a strong performer with an RMSE of **1.5828**, closely matching the NCF model before tuning.
    - The hybrid model's adjustments improved precision slightly but still resulted in low recall, indicating the need for further refinement.
- **Insights and Learnings**:
    - **Hyperparameter tuning is crucial** for unlocking the full potential of complex models like NCF.
    - Properly optimized advanced models can outperform traditional methods, even in sparse datasets.
    - Balancing precision and recall is vital for user satisfaction.
    - Data quality and quantity significantly impact model effectiveness.

Future Work

To enhance the recommendation system further, the following steps are recommended:

- **Data Augmentation**: Incorporate additional user ratings and item metadata (e.g., genres, descriptions).
- **Advanced Techniques**: Explore other algorithms like matrix factorization for implicit feedback, ensemble methods, or attention mechanisms.
- **Address Cold-Start Problem**: Implement strategies for new users and items, possibly through content-based methods or using demographic data.
- **Hybrid Model Adjustment**: Continue refining the hybrid model, possibly by exploring different combinations of models or adjusting the weighting scheme to improve recall.

By implementing hyperparameter tuning on the NCF model and adjusting the hybrid model's weights, we improved the recommendation system's accuracy and demonstrated the importance of model optimization. Initially, the SVD model was the best performer, but through iterative enhancements, the NCF model ultimately surpassed it, highlighting the value of continuous model refinement. We adhered to our professor's instructions by applying the planned future work, which significantly enhanced our project's outcomes.

---

8. References

1. **Book-Crossing Dataset**: Kaggle - Book Recommendation Dataset
2. **Surprise Library Documentation**: Surprise - A Python scikit for recommender systems
3. **TensorFlow Keras Documentation**: TensorFlow Keras API
4. **Scikit-learn Documentation**: Scikit-learn Machine Learning in Python
5. **Pandas Documentation**: Pandas - Python Data Analysis Library

---

Note: Throughout the project, we carefully considered the challenges posed by data sparsity and computational complexity. By following our professor's guidance and applying hyperparameter tuning, we achieved significant improvements in model performance, demonstrating the effectiveness of advanced techniques in recommendation systems.