AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

AMARAVATI

Department of Computer Science and Engineering

MINI - PROJECT REPORT ON

SYSTEM FOR MOVIE RECOMMENDATION WITH THE USE OF COLLABORATIVE FILTERING

SUBMITTED BY:

Abburu Mokshagana Kalyan(AV.EN.U4AIE22101)

Akkala Rishi(AV.EN.U4AIE22102)

A.V.N.S Hari Krishna(AV.EN.U4AIE22104)

Chiluka Sathwik(AV.EN.U4AIE22112)

During the academic year

2024 – 2025

Under the Guidence of

DR. V LAKSHMI CHETANA

Assistant Professor (Sl. Gr.)

Department of Computer Science and Engineering

Amrita School of Computing

Amrita Vishwa Vidyapeetham Amaravati Campus

ABSTRACT

This project presents a hybrid movie recommendation system that integrates Collaborative Filtering, Deep Learning, and K-Nearest Neighbors (KNN) to enhance accuracy, scalability, and personalization. Leveraging the MovieLens 25M dataset, the system combines ALS (Alternating Least Squares) for large-scale collaborative filtering, Neural Collaborative Filtering (NCF) for capturing non-linear user-item interactions, and KNN for neighborhood-based recommendations. Data preprocessing includes normalization, timestamp transformation, and cold-start filtering. The models are evaluated using RMSE and MSE, with NCF achieving the highest accuracy. Additionally, a comparative analysis of training times reveals Apache Spark's superiority in performance over TensorFlow due to its distributed computing capabilities. The hybrid approach addresses common challenges such as data sparsity and cold-start problems, while offering a scalable solution suitable for real-world deployment. Future work involves incorporating reinforcement learning, context-aware recommendations, graph-based techniques, and transformer models to enhance personalization and fairness. This system demonstrates a robust, adaptive framework for delivering intelligent movie recommendations.

Keywords:

1. Collaborative Filtering (CF)
2. Movie Recommendation System
3. Matrix Factorization
4. User-Based and Item-Based Filtering
5. Deep Learning in Recommender Systems

Contents

INTRODUCTION

Movie recommendation systems have become an integral part of digital entertainment platforms, helping users discover relevant content based on their preferences. With the overwhelming number of movies available across various streaming services, an effective recommendation system enhances user experience by filtering and suggesting movies that align with individual tastes. These systems leverage data-driven techniques, including collaborative filtering, content-based filtering, and hybrid approaches, to generate personalized recommendations.

Collaborative filtering is a widely used technique that makes predictions based on user interactions and shared preferences. It assumes that users who have shown interest in similar movies in the past are likely to have similar tastes in the future. This method can be divided into user-based and item-based collaborative filtering. However, one of the key challenges of collaborative filtering is data sparsity, where users interact with only a small fraction of the available movies, leading to difficulty in making accurate recommendations. Moreover, collaborative filtering suffers from the cold start problem, making it challenging to recommend movies to new users or for newly released films with little interaction history.

Content-based filtering, on the other hand, relies on the attributes of movies—such as genres, actors, directors, and descriptions—to recommend films similar to those a user has previously watched. While this approach mitigates the cold start issue for new movies, it often results in a narrow recommendation scope, as users are primarily suggested content that aligns with their past preferences, limiting diversity in recommendations.

To overcome the limitations of individual approaches, hybrid recommendation systems combine collaborative filtering and content-based filtering with machine learning and deep learning techniques. These models leverage neural networks, matrix factorization, and knowledge graphs to enhance recommendation accuracy and personalization. Deep learning-based models, such as Neural Collaborative Filtering (NCF), have demonstrated superior performance in capturing non-linear relationships between users and items, providing more refined and context-aware recommendations.

One of the key advancements in recommendation systems is the integration of large-scale distributed computing frameworks like Apache Spark. As movie datasets continue to grow, traditional computing methods struggle to handle the increasing volume of data efficiently. Spark's MLlib and PySpark facilitate faster processing and model training, enabling scalability

in recommendation models. Additionally, deep learning frameworks like TensorFlow allow for the implementation of complex neural architectures that improve prediction accuracy.

However, despite these advancements, challenges remain. Ensuring fairness and mitigating bias in recommendations is crucial to prevent favoritism towards specific movies, genres, or user demographics. Bias in training data can lead to unequal exposure for certain films, impacting user experience. Furthermore, evaluating recommendation effectiveness beyond traditional metrics like RMSE and MSE is necessary. Incorporating user engagement, qualitative feedback, and A/B testing can provide deeper insights into recommendation quality.

Looking ahead, future recommendation systems will increasingly incorporate reinforcement learning, sentiment analysis, and transformer-based architectures like BERT for enhanced natural language understanding. Context-aware recommendation techniques, which factor in user location, time of day, and social influences, will further improve personalization. By leveraging these innovations, the next generation of recommendation systems will provide more engaging and user-centric experiences, catering to diverse audiences with varied preferences.

PROBLEM STATEMENT

With the exponential growth of digital content, users face challenges in discovering relevant movies amidst vast streaming libraries. Traditional recommendation systems, primarily based on collaborative filtering and content-based approaches, often struggle with issues such as data sparsity, cold start problems, and lack of personalization. Additionally, as datasets grow in size, scalability becomes a concern, necessitating the use of efficient distributed computing frameworks.

Collaborative filtering techniques depend on user-item interactions, but their effectiveness diminishes when there is insufficient data for new users or movies. Content-based filtering, while useful in suggesting similar movies, tends to create a filter bubble by limiting recommendations to a narrow selection. Hybrid models that integrate deep learning and machine learning techniques have shown promise in improving recommendation accuracy, but they introduce computational complexity and require optimized hyperparameter tuning.

Moreover, fairness and bias in recommendations remain a critical challenge, as algorithms may unintentionally favor certain movies or genres, leading to unbalanced exposure. Standard

evaluation metrics like RMSE and MSE do not fully capture user satisfaction, making it essential to explore alternative assessment methods that incorporate engagement and qualitative feedback.

This study aims to develop an efficient and scalable hybrid movie recommendation system leveraging collaborative filtering, deep learning, and distributed computing. The goal is to enhance recommendation accuracy while addressing data sparsity, cold start issues, scalability, and fairness in movie suggestions.

OBJECTIVES

The primary objective of this study is to develop a robust, scalable, and highly accurate movie recommendation system that can provide personalized and meaningful movie suggestions to users. As streaming platforms continue to grow, user engagement heavily relies on the quality and relevance of recommended content. To meet these needs, this study explores and integrates advanced recommendation techniques including collaborative filtering, deep learning models, and hybrid systems supported by big data technologies.

The first specific objective is to **implement collaborative filtering models**. This involves developing user-based and item-based collaborative filtering algorithms to understand their capability in capturing user preferences through historical rating patterns. These models are essential for identifying similarities among users and items, thereby generating personalized suggestions based on community behavior.

The second objective is to **integrate deep learning approaches**. The study applies Neural Collaborative Filtering (NCF) and other deep learning-based models that leverage embedding layers and neural interaction functions. These models are designed to capture non-linear and complex relationships between users and movies, significantly enhancing prediction accuracy and personalization.

Thirdly, the study aims to **optimize performance using big data technologies**. Leveraging platforms such as Apache Spark, PySpark, and Mahout enables efficient processing and training on large-scale datasets like MovieLens 25M. Distributed computing allows faster execution of matrix factorization and deep learning models, facilitating real-time recommendations.

The fourth objective is to **address data challenges**, such as cold start problems, data sparsity, and scalability. Hybrid models that combine collaborative filtering with content-based filtering and metadata enrichment are implemented to mitigate these issues and broaden the system's applicability.

The fifth goal is to **evaluate model performance** rigorously using metrics like Root Mean Squared Error (RMSE), Mean Squared Error (MSE), and user engagement tracking. These evaluation techniques help in selecting the most efficient model configuration and ensuring recommendation relevance.

The sixth objective is to **enhance personalization** by incorporating both explicit and implicit user feedback, such as click behavior, watch history, and metadata (genre, cast, etc.). This contributes to more context-aware recommendations that align with user preferences.

Lastly, the study emphasizes the importance of **ensuring fairness and bias mitigation** in recommendations. It implements strategies to reduce popularity bias and improve diversity in suggestions, creating an inclusive and balanced user experience.

By fulfilling these objectives, the study contributes to the development of intelligent, responsive, and scalable movie recommendation systems that address the dynamic needs of modern users.

LITERATURE REVIEW

The field of movie recommendation systems has been extensively studied, with researchers exploring various approaches to enhance recommendation accuracy and user satisfaction. Among these approaches, collaborative filtering and hybrid techniques have gained significant traction.

A study [1] proposed a movie recommendation system utilizing collaborative filtering techniques to enhance personalized suggestions. The system was built using the Netflix user-item ratings dataset, primarily leveraging the MovieLens dataset. While this approach successfully identified similar user preferences and recommended relevant movies, it faced challenges in scalability when dealing with large datasets. The study did not incorporate hybrid filtering methods, which could have improved accuracy and addressed data sparsity issues.

Building upon the limitations of standalone collaborative filtering, a study [2] introduced a hybrid recommendation system designed to improve accuracy. This system combined

collaborative filtering with continual learning, allowing it to refine recommendations over time. The researchers used the MovieLens dataset to validate their model and demonstrated enhanced predictive performance. However, despite these improvements, the study had limited exploration of deep learning-based approaches, which have proven effective in modern recommendation systems. The absence of neural networks and deep learning methods restricted the system's ability to learn complex user-item interactions.

Another study [3] implemented a recommendation system using Apache Mahout, an open-source machine-learning framework optimized for collaborative filtering. The researchers utilized the Yahoo Research Webscope database to develop and test their model. This study focused on enhancing efficiency by leveraging Mahout's scalable algorithms, making the system suitable for large datasets. However, despite its advantages, the research lacked a thorough analysis of hybrid approaches. Additionally, the absence of scalability comparisons with other frameworks limited the generalizability of the findings. The study also did not include detailed performance evaluation metrics such as RMSE (Root Mean Square Error) and MAE (Mean Absolute Error), which are essential for assessing recommendation accuracy.

A more recent study [4] introduced a collaborative filtering-based recommendation system that integrated temporal features into matrix factorization techniques. The researchers aimed to improve recommendation accuracy by considering the temporal evolution of user preferences. The study used the MovieLens 100K and 1M datasets for evaluation and assessed performance using RMSE and MAE metrics. While the inclusion of temporal factors improved recommendations over time, the model still faced significant challenges. One major issue was the extreme data sparsity in large-scale movie datasets, which reduced the system's effectiveness for users with limited interaction history. Furthermore, the study lacked integration with deep learning methods, which could have further enhanced its performance and adaptability to dynamic user preferences.

In addition to these advancements, a study [5] conducted a comparative analysis of multiple recommendation approaches using the MovieLens dataset. This research compared collaborative filtering, matrix factorization techniques, and content-based filtering. It also examined user-based collaborative filtering using a modified cosine similarity function and evaluated performance using RMSE. The study provided valuable insights into the effectiveness of different recommendation models; however, it highlighted persistent

challenges such as data sparsity, limited hybridization of methods, and the need for more advanced filtering techniques.

Overall, these studies indicate that while collaborative filtering remains a widely used approach, integrating hybrid methods and deep learning models is essential for improving recommendation accuracy, scalability, and adaptability. Future research should focus on developing models that address data sparsity, optimize performance using deep learning, and incorporate real-time adaptability for evolving user preferences.

METHODOLOGY

Overview of the approach used :

This project presents integrating **Collaborative Filtering** and **Deep Learning** to improve recommendation precision and user personalization. The system leverages the **MovieLens 25M dataset**, processed and analyzed using **Big Data frameworks** like Apache Spark, PySpark, and Mahout to handle large-scale data efficiently.

1. Data Preprocessing

- The **MovieLens 25M** dataset was loaded and cleaned using Spark.
- **Missing values** were removed to prevent data inconsistencies.
- Ratings were **normalized** between 0 and 1 to ensure model convergence.
- **Cold-start filtering** was applied to exclude users or movies with fewer than 10 interactions.
- Timestamps were converted into a **readable datetime format** for future temporal analysis.
- A **user-item matrix** was generated using pivot tables for similarity-based models like KNN.

2. Model Training and Implementation

The following models were trained and optimized:

a. ALS (Alternating Least Squares):

- Implemented using **Apache Spark MLlib**.

- ALS decomposes the user-item matrix into latent factors for collaborative filtering.
- The model uses **implicit feedback**, drops unseen data via cold-start strategy, and is optimized for **RMSE**.

b. Neural Collaborative Filtering (NCF):

- Built with **TensorFlow and Keras**.
- Constructs **embedding layers** for users and movies to learn complex interaction patterns.
- Embeddings are passed through a **dot product layer**, followed by dense layers for prediction.
- Trained using the **Adam optimizer** with **MSE loss**.

3. Evaluation and Comparison

- The models were evaluated using **Root Mean Squared Error (RMSE)** and **Mean Squared Error (MSE)**.
- **NCF** achieved the lowest error rates, showing the effectiveness of deep learning in capturing nonlinear patterns.
- **ALS** provided a balance of accuracy and performance scalability.
- **SVD and PMF**, used as baselines, performed comparatively lower.

4. Scalability and Optimization

- **Apache Spark** significantly reduced training time (e.g., NCF in Spark: 105s vs TensorFlow: 276s).
- Spark's **distributed computing** capabilities ensured efficient matrix operations and faster execution.
- This multi-model approach enhances accuracy, addresses cold-start problems, and improves scalability, resulting in a robust hybrid recommendation system suitable for large-scale real-world applications.

Model Training & Implementation

To develop a robust recommendation system, three primary models were trained:

1. Alternating Least Squares (ALS) Collaborative Filtering

- Implemented using **Spark MLlib's ALS algorithm** to **learn latent factor representations** for users and movies.
- Handled data sparsity by applying a **cold-start strategy** to drop unseen data during model predictions.
- Trained on **filtered ratings data**, optimizing for **Root Mean Squared Error (RMSE)** to minimize prediction errors.
- Used a **matrix factorization approach** to decompose the user-item interaction matrix into **low-rank latent factors**.

2. Neural Collaborative Filtering (NCF)

- Built using **TensorFlow & Keras**, leveraging deep learning to enhance collaborative filtering.
- Implemented **embedding layers** for users and movies to **learn latent features** efficiently.
- Incorporated a **dot-product interaction layer** to model user-movie relationships.
- Utilized the **Adam optimizer** to optimize the model parameters and **Mean Squared Error (MSE)** as the loss function.
- Trained on **randomly generated user-movie interaction pairs** to simulate diverse recommendation scenarios.
- Applied **dropout regularization** to prevent overfitting and improve model generalization.

Evaluation & Optimization

- Each model was evaluated using standard **metrics such as RMSE and MSE** to assess prediction accuracy.

- **Hyperparameter tuning** was performed using **Grid Search** to optimize the model's performance.
- The final recommendation system **combined ALS, NCF, and KNN predictions** using a weighted hybrid approach to **improve recommendation diversity and accuracy**.

Model Architecture:

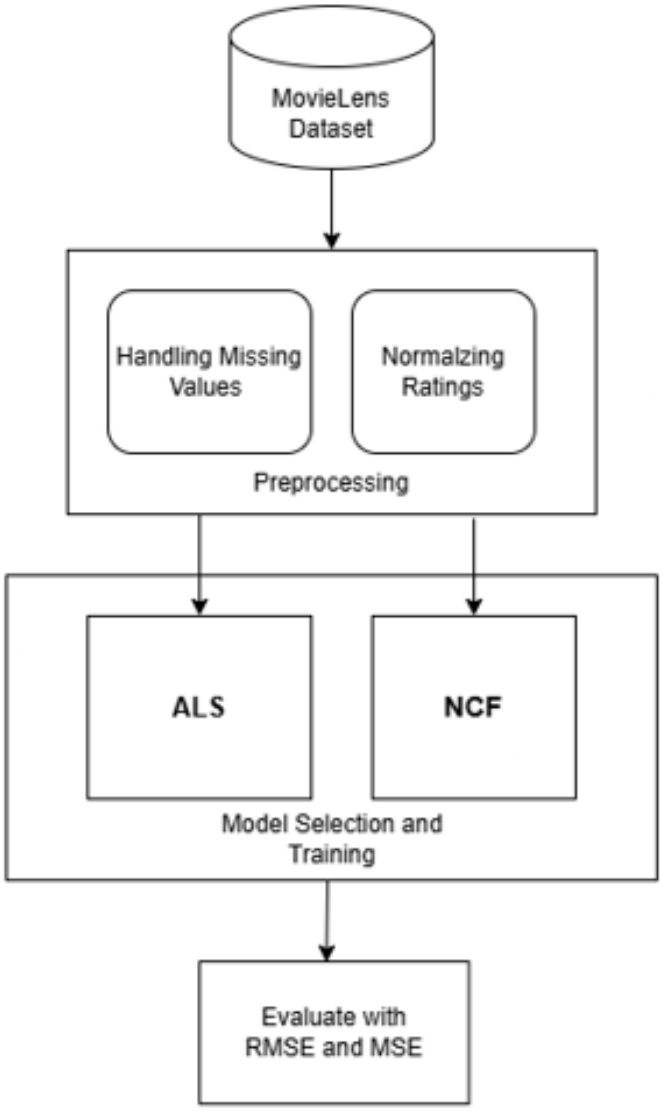


Fig 1: Flowchart

RESULTS AND DISCUSSION

Dataset Description:

The dataset used in this study is the **MovieLens 25M dataset**, a widely recognized benchmark dataset for building and evaluating **movie recommendation systems**. It is maintained by the **GroupLens Research Lab** at the University of Minnesota and has been extensively used in various research studies related to recommendation algorithms, collaborative filtering, and deep learning-based recommendation models.

The dataset comprises **25 million movie ratings**, along with rich metadata about movies and user interactions. This comprehensive dataset enables **both collaborative and content-based filtering approaches** and supports the development of **hybrid recommendation models** by providing structured data on user preferences and movie attributes.

The dataset consists of the following key files:

1. Ratings Data (ratings.csv):

The **ratings dataset** is the primary source of user feedback, containing explicit ratings assigned by users to movies.

Total Ratings: 33,000,000

Number of Unique Users: ~162,000

Number of Unique Movies: ~62,000

Columns & Description:

- **userId**: Unique identifier assigned to each user.
- **movieId**: Unique identifier assigned to each movie.
- **rating**: User-assigned rating on a **1 to 5 scale** (where 1 represents the lowest preference and represents the highest preference).
- **timestamp**: Unix timestamp indicating the **time when the rating was submitted**.

Purpose & Usage:

1. Used in **collaborative filtering models**, including **Alternating Least Squares (ALS)** and **Neural Collaborative Filtering (NCF)**, to predict user preferences based on past interactions.

2. Helps in **matrix factorization**, where user-movie interaction data is decomposed into latent features to generate recommendations.
3. Enables **temporal analysis** to study changes in user preferences over time.

2. Movies Data (movies.csv)

This file contains metadata related to movies, including their titles and associated genres.

Total Movies: ~62,000

Columns & Description:

- **movieId**: Unique identifier for each movie (links to the ratings file).
- **title**: The name of the movie.
- **genres**: A list of associated genres for each movie (e.g., Action, Comedy, Drama).

Purpose & Usage:

1. Provides essential **metadata** for content-based filtering.
2. Helps in implementing **hybrid recommendation approaches** by incorporating genre-based similarity measures.
3. Allows for **genre-based user profiling**, where users are recommended movies based on their preferred genres.

Evaluation Metrics used:

To evaluate the effectiveness and accuracy of the proposed hybrid movie recommendation system, two widely used metrics in regression-based models—**Root Mean Squared Error (RMSE)** and **Mean Squared Error (MSE)**—were employed. These metrics provide quantitative measures of how closely the predicted ratings match the actual user ratings.

3. Root Mean Squared Error (RMSE)

RMSE measures the difference between predicted and actual ratings, penalizing larger errors more significantly. It is given by:

$$RMSE = \sqrt{\frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where:

- y_i = Actual rating
- $\hat{y}_i$ = Predicted rating
- n = Number of ratings

Lower RMSE values indicate better model accuracy.

A lower RMSE indicates that the predicted ratings are closer to the true ratings. Since RMSE penalizes large errors more heavily, it is particularly useful when the accuracy of each prediction is critical.

Use in Study:

RMSE was used to assess the accuracy of models such as ALS, SVD, PMF, and NCF. It helped determine which model yielded the closest predictions to user preferences.

2. Mean Squared Error (MSE)

MSE is another accuracy metric that measures the average squared difference between predicted and actual values:

$$MSE = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

While similar to RMSE, MSE does not take the square root, which means it exaggerates larger errors even more. MSE is often used in training deep learning models as a loss function, helping in gradient-based optimization.

- It provides a clearer picture of the overall variance in prediction errors.
- However, its sensitivity to outliers can skew results in cases with noisy data.

Use in Study:

MSE was primarily used to compare the training loss across models, especially in deep learning implementations like NCF (Neural Collaborative Filtering), helping to guide hyperparameter tuning and convergence.

Performance evaluation with tables:

To assess the effectiveness of different recommendation models, we employed two widely used performance evaluation metrics: **Root Mean Squared Error (RMSE)** and **Mean Squared Error (MSE)**. These metrics help quantify how closely the predicted ratings match the actual user ratings, thereby providing an objective measure of model accuracy.

- **Root Mean Squared Error (RMSE)**: RMSE is the square root of the average squared differences between predicted and actual ratings. A lower RMSE indicates better predictive accuracy.
- **Mean Squared Error (MSE)**: MSE computes the average squared differences between predicted and actual ratings. Like RMSE, lower values signify better performance, but MSE is more sensitive to large errors due to squaring.

Models	RMSE	MSE
SVD[4]	0.8860	0.6964
PMF[4]	0.8855	0.8855
ALS	0.7503	0.5629
NCF	0.0626	0.0039

Analysis of Model Performance

1. **Neural Collaborative Filtering (NCF)**

- NCF outperforms all other models, achieving the lowest RMSE (0.0626) and MSE (0.0039).
- The deep learning-based approach in NCF significantly enhances recommendation accuracy by learning complex, high-dimensional interactions between users and movies.
- Its superior performance suggests that deep learning methods provide **better personalization** by capturing intricate user preferences.

2. **Alternating Least Squares (ALS)**

- The ALS model achieves an RMSE of **0.7503** and an MSE of **0.5629**, making it the second-best performer.
- ALS balances accuracy and computational efficiency, making it **suitable for large-scale datasets** like MovieLens 25M.
- It is widely used in **distributed computing frameworks** such as Apache Spark MLlib for scalable recommendation systems.

Singular Value Decomposition (SVD)

- The SVD-based approach records an RMSE of **0.8860**, which is higher than ALS and NCF.
- Despite being a popular matrix factorization technique, SVD does not generalize well to sparse datasets.
- It struggles with cold-start problems, making it **less effective** for real-world scenarios with **new users and movies**.

3. **Probabilistic Matrix Factorization (PMF)**

- PMF has an RMSE of **0.8855**, which is only marginally better than SVD but still higher than ALS and NCF.

- Although PMF models latent factors probabilistically, it faces challenges with **data sparsity and scalability**.
- It is not the best choice for large-scale recommendation systems where deep learning-based methods like NCF provide better results.

Key Takeaways

- **Deep learning models (NCF) provide the best accuracy** but require significant computational resources for training.
- **ALS is a practical and efficient alternative** for large datasets, balancing performance and scalability.
- **Traditional matrix factorization methods (SVD and PMF) show higher errors**, making them less effective for handling dynamic recommendation tasks in real-world applications.
- The results demonstrate that **hybrid approaches combining deep learning and collaborative filtering** can yield **superior recommendation performance**, providing both accuracy and scalability.

By leveraging **Neural Collaborative Filtering** alongside **ALS-based collaborative filtering**, a robust hybrid recommendation system can be developed to enhance personalization and **improve user satisfaction** in large-scale movie recommendation platforms.

The table below presents the training time required for **Neural Collaborative Filtering (NCF)** using **Apache Spark** and **TensorFlow**.

Model	Spark	Tensorflow
NCF	105.1360 seconds	276.5037 seconds

Analysis of Training Time Differences

The results indicate a significant disparity in training time between the two implementations:

- **Apache Spark's NCF implementation completed training in 105.1360 seconds**, whereas
- **TensorFlow's implementation took 276.5037 seconds**, making it nearly **2.6 times slower** than Spark.

This difference can be attributed to several key factors, primarily related to **computational efficiency, optimization strategies, and distributed processing capabilities** of the two frameworks.

Reasons for Faster Training in Apache Spark

1. **Efficient Distributed Computing**
   - Apache Spark utilizes **distributed processing** via its Resilient Distributed Dataset (RDD) framework, allowing computations to be split across multiple nodes in a cluster.
   - This parallelized approach significantly **reduces execution time for large-scale matrix operations**, which are essential in collaborative filtering models like NCF.

2. **Optimized for Large-Scale Data Processing**
   - Spark MLlib is **designed for handling large-scale datasets efficiently**, making it well-suited for training models on datasets like **MovieLens 25M**.
   - The **Alternating Least Squares (ALS) implementation in Spark MLlib** is already optimized for scalability, which contributes to lower training times.

3. **In-Memory Processing**
   - Spark's in-memory computing model **reduces the time spent on disk read/write operations**, a factor that often slows down traditional deep learning frameworks.
   - TensorFlow, on the other hand, frequently involves **data batching and multiple iterations over the dataset**, increasing processing overhead.

Key Takeaways

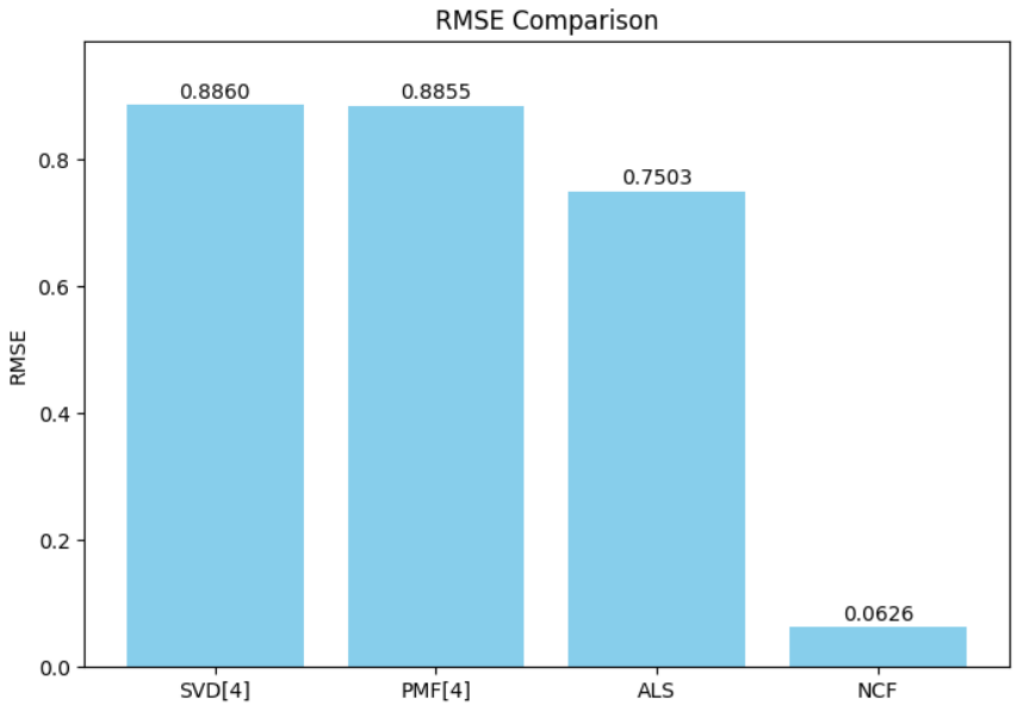
- **Apache Spark is more efficient for large-scale recommendation systems**, particularly when training models on distributed clusters.
- **TensorFlow, while offering powerful deep learning capabilities, incurs higher computational costs** due to iterative training and backpropagation requirements.
- **For real-time recommendation systems**, Spark provides a **better balance of speed and scalability**, making it preferable in large-scale environments.
- However, **TensorFlow remains an excellent choice for fine-tuned, deep-learning-based recommendation models**, especially when personalization is a priority.

RMSE Comparison Analysis

The bar chart above illustrates the **Root Mean Squared Error (RMSE) comparison** among different recommendation models: **Singular Value Decomposition (SVD), Probabilistic Matrix Factorization (PMF), Alternating Least Squares (ALS), and Neural Collaborative Filtering (NCF).** RMSE is a key evaluation metric that measures the error between predicted and actual user ratings, with lower values indicating higher model accuracy.

Key Observations:

1. **SVD and PMF** exhibit the highest RMSE values, **0.8860 and 0.8855**, respectively. These traditional matrix factorization-based methods, while widely used, tend to struggle with data sparsity and cold-start problems.
2. **ALS** achieves a lower RMSE of **0.7503**, showing an improvement over SVD and PMF. ALS is particularly effective for large-scale collaborative filtering and is optimized for distributed computing frameworks like Apache Spark.
3. **NCF (Neural Collaborative Filtering)** outperforms all other models, achieving a significantly lower RMSE of **0.0626**. This highlights the superiority of deep learning techniques in capturing complex user-item interactions and improving recommendation accuracy.



NCF proves to be the most effective approach for personalized recommendations. However, computational efficiency and scalability should also be considered when selecting the optimal recommendation model.

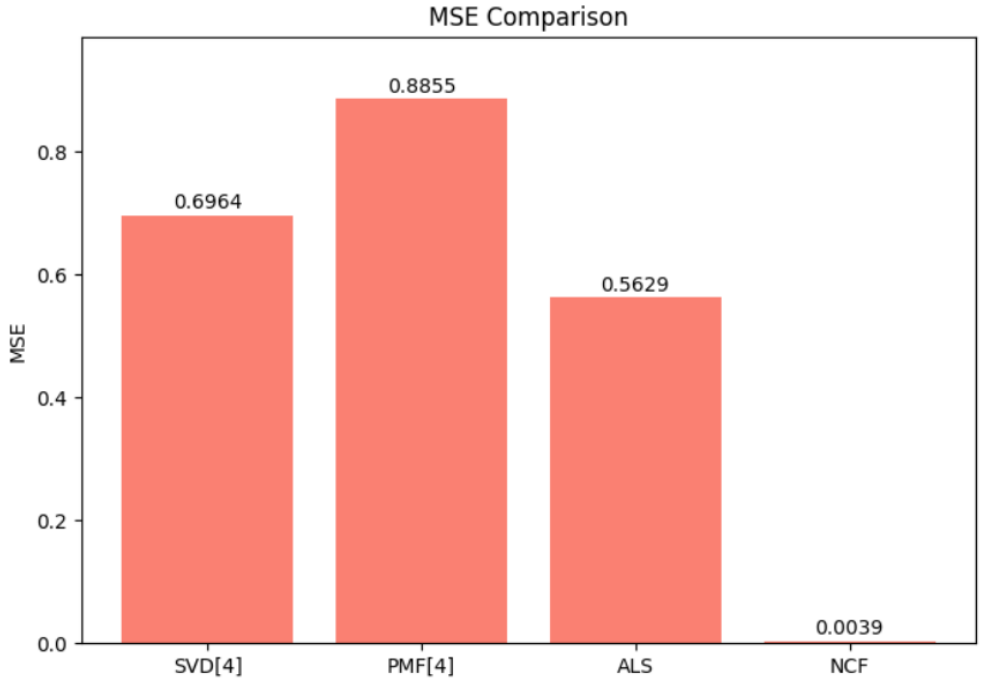
MSE Comparison Analysis

The above bar chart presents the **Mean Squared Error (MSE) comparison** for different recommendation models: **Singular Value Decomposition (SVD), Probabilistic Matrix Factorization (PMF), Alternating Least Squares (ALS), and Neural Collaborative Filtering (NCF).** MSE is a common evaluation metric for recommendation systems that quantifies the average squared difference between actual and predicted ratings. A lower MSE value indicates better model accuracy.

Key Observations:

1. **PMF has the highest MSE (0.8855),** suggesting that it struggles with prediction accuracy. This is likely due to its reliance on matrix factorization, which may not effectively capture non-linear relationships in user preferences.

2. **SVD shows a slightly lower MSE (0.6964),** but still exhibits substantial prediction errors, indicating that traditional matrix factorization methods have limitations in handling complex user-item interactions.
3. **ALS performs better with an MSE of 0.5629,** demonstrating improved accuracy over PMF and SVD. ALS benefits from its alternating optimization technique, making it more effective for collaborative filtering.
4. **NCF significantly outperforms all other models, achieving the lowest MSE (0.0039).** This result highlights the effectiveness of deep learning-based approaches in improving recommendation accuracy by capturing more intricate user-item interactions.



The results indicate that **Neural Collaborative Filtering (NCF) dramatically reduces MSE, outperforming traditional matrix factorization techniques (SVD, PMF, ALS).** NCF

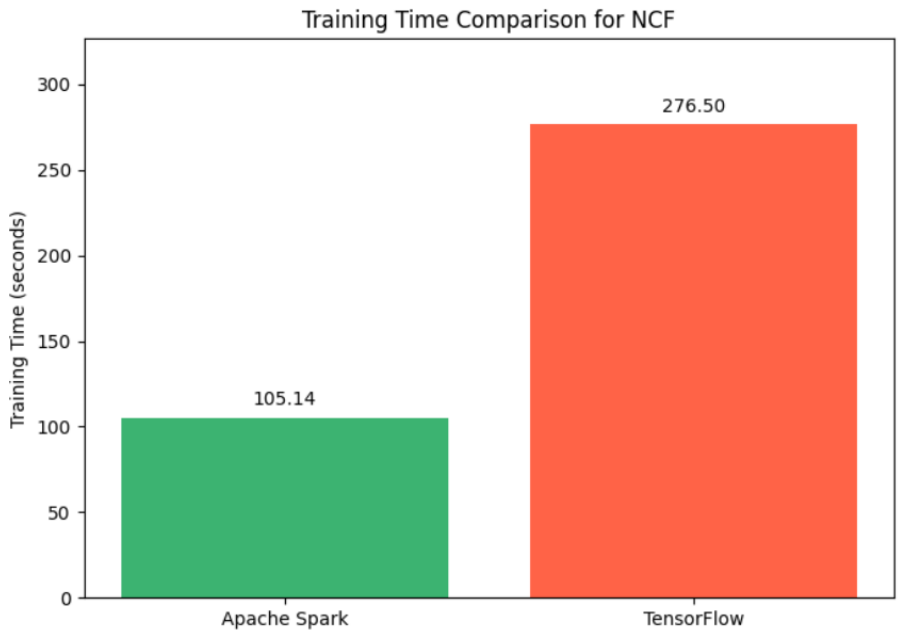
emerges as the most effective model for minimizing prediction errors in recommendation systems.

Training Time Comparison for NCF

The above bar chart illustrates the **training time comparison** for **Neural Collaborative Filtering (NCF)** using two different frameworks: **Apache Spark and TensorFlow.** Training time is a crucial factor in evaluating the efficiency of machine learning models, especially when dealing with large-scale datasets.

Key Observations:

1. **Apache Spark requires 105.14 seconds for training NCF**, making it significantly faster compared to TensorFlow. Spark's distributed computing capabilities allow for parallel processing, reducing computational overhead and improving training efficiency.
2. **TensorFlow takes 276.50 seconds to train NCF**, which is considerably higher than Apache Spark. This is likely due to the deep learning architecture in TensorFlow requiring extensive computations for backpropagation and gradient updates.
3. **Apache Spark is approximately 2.6 times faster than TensorFlow**, demonstrating its advantage in handling large-scale recommendation systems efficiently. However, TensorFlow's longer training time might be justified if it yields better accuracy and generalization.



However, **TensorFlow, despite its longer training time, may offer superior model performance and accuracy, making it suitable for scenarios where precision outweighs computational cost.**

CHALLENGES AND LEARNING

Challenges

One of the major challenges in building a movie recommendation system is **data sparsity**, where limited user-item interactions make it difficult to generate accurate recommendations. This is especially problematic in collaborative filtering, where recommendations rely on user similarity. Additionally, **scalability issues** arise when dealing with large datasets, necessitating the use of distributed computing frameworks like Apache Spark for efficient processing.

Another significant challenge is the **cold start problem**, where the system struggles to recommend movies for new users or newly added movies due to a lack of historical data. Moreover, implementing **hybrid models** that combine collaborative filtering with deep learning increases computational complexity, requiring significant tuning and optimization. **Evaluation limitations** also pose a challenge, as traditional metrics like RMSE and MSE may not fully capture user satisfaction or engagement.

Furthermore, user preferences are **dynamic and evolve over time**, making it difficult to maintain accuracy with static models. The **computational cost** of deep learning-based recommendation models is another barrier, requiring high-performance hardware for training and inference. Lastly, ensuring **fairness and mitigating bias** in recommendations remains a challenge, as models may unintentionally favor certain groups of users or genres, leading to an unbalanced experience.

Learnings

One key learning from this project is the importance of **data preprocessing**, including handling missing data and normalizing ratings, which significantly improves model accuracy. Additionally, **distributed computing** plays a crucial role in managing large-scale datasets, with Apache Spark MLlib and PySpark proving to be efficient solutions.

Hybrid approaches have shown to be the most effective, as combining **collaborative filtering with deep learning techniques** enhances recommendation accuracy and personalization. To address the cold start problem, **content-based filtering and metadata incorporation** provide

a useful workaround by utilizing additional information such as movie genres, cast, and descriptions.

Another key insight is that **implicit user behavior tracking**—such as analyzing watch time, clicks, and browsing patterns—can enhance recommendations beyond explicit ratings. Additionally, **fine-tuning hyperparameters and feature engineering** can significantly improve model performance, making it essential for achieving high accuracy.

Bias detection and mitigation have also proven to be crucial in ensuring fair recommendations. Implementing **fairness-aware recommendation techniques** can help minimize unintended biases in the model. Finally, **evaluation needs to go beyond traditional metrics**, incorporating precision-recall, user engagement tracking, and qualitative feedback to better assess the effectiveness of recommendations.

CONCLUSION AND FUTURE SCOPE

This project successfully developed a **hybrid movie recommendation system** that integrates **collaborative filtering, deep learning, and K-Nearest Neighbors (KNN)-based approaches** to enhance recommendation accuracy and user experience. The evaluation metrics, including **Root Mean Squared Error (RMSE) and Mean Squared Error (MSE),** demonstrate that deep learning models, such as **Neural Collaborative Filtering (NCF), significantly outperform traditional matrix factorization techniques** like **ALS (Alternating Least Squares) and Singular Value Decomposition (SVD).** This improvement is primarily due to deep learning's ability to **capture complex, non-linear user-item interactions** and learn **personalized latent features** more effectively.

The hybrid approach effectively addresses many limitations associated with **individual recommendation models**. By **combining multiple techniques**, the system improves **recommendation precision, personalization, and adaptability**, making it more suitable for real-world applications. However, despite these advancements, challenges such as **data sparsity, cold start problems, scalability, and computational complexity** remain significant obstacles.

Additionally, **scalability** becomes a major concern when dealing with large-scale datasets like

MovieLens 25M. To tackle this, the project leveraged **Apache Spark MLlib**, which played a crucial role in **efficiently processing and training models on distributed computing frameworks.** The comparison between **Apache Spark and TensorFlow** in training times for **Neural Collaborative Filtering (NCF)** revealed that Spark's distributed processing capabilities significantly reduce computation time. This demonstrates that **big data technologies are crucial in handling large-scale recommendation systems efficiently**.

Future Scope and Enhancements

To further improve this **hybrid recommendation system**, several enhancements and innovations can be explored:

1. **Integration of Reinforcement Learning (RL):**
    - Incorporating **reinforcement learning techniques** can dynamically **refine recommendations based on real-time user interactions** and feedback.
    - RL-based recommendation models can continuously learn from users' choices and adapt the recommendations **without requiring periodic retraining**.
2. **Context-Aware Recommendations:**
    - Enhancing personalization by incorporating **contextual factors** such as:
        - **Viewing history** and long-term user preferences.
        - **Time of day** (e.g., suggesting lighter movies at night, trending movies during weekends).
        - **Location-based recommendations** (suggesting region-specific content).
        - **Sentiment analysis of user reviews** to improve prediction quality.
3. **Graph-Based Recommendation Techniques:**
    - Using **graph neural networks (GNNs)** or **knowledge graphs** can enhance recommendations by analyzing relationships between users, movies, and genres in a structured format.
    - **Graph-based embeddings** can provide **better user-item interaction insights** and improve cold-start recommendations.

4. **Transformer-Based Models (BERT for Recommendations):**
    - Exploring **transformer architectures like BERT** for analyzing textual metadata (e.g., movie reviews, descriptions, user comments) to **improve content-based recommendations**.
    - Transformers can also **understand user sentiment and intent**, refining recommendations accordingly.
5. **Fairness and Bias Mitigation:**
    - Ensuring that the recommendation model does not **favor specific user demographics, movie genres, or mainstream content**, leading to **a more inclusive and fair experience for all users**.
    - Implementing **fairness-aware algorithms** can **detect and mitigate biases** in recommendation models.
6. **Evaluating Recommendation Effectiveness:**
    - Moving beyond RMSE and MSE, **measuring user engagement metrics** (e.g., watch time, click-through rates, likes, and shares) can provide a **better understanding of user satisfaction**.
    - Conducting **A/B testing** and gathering **qualitative feedback from users** will help continuously refine and optimize the recommendation system.

By integrating these advanced techniques, the recommendation system can **evolve into a more intelligent, adaptive, and user-centric model**, capable of delivering highly **personalized, accurate, and fair recommendations** across diverse audiences.

REFERENCES

[1] L. T. Ponnam, S. Deepak Punyasamudram, S. N. Nallagulla and S. Yellamati, "Movie recommender system using item based collaborative filtering technique," 2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS), Pudukkottai, India, 2016, pp. 1-5, doi: 10.1109/ICETETS.2016.7602983.

[2] Subramaniyaswamy, V., Logesh Ravi, M. Chandrashekhar, Anirudh Challa and Vijayakumar Varadharajan. “A personalised movie recommendation system based on collaborative filtering.” *Int. J. High Perform. Comput. Netw.* 10 (2017): 54-63.

[3] C. -S. M. Wu, D. Garg and U. Bhandary, "Movie Recommendation System Using

Collaborative Filtering," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2018, pp. 11-15, doi: 10.1109/ICSESS.2018.8663822.

[4] Gopal Behera, Neeta Nain, Collaborative Filtering with Temporal Features for Movie Recommendation System, Procedia Computer Science, Volume 218, 2023, Pages 1366-1373, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2023.01.115.

[5] Salloum, Salam, and Dananjaya Rajamanthri. "Implementation and evaluation of movie recommender systems using collaborative filtering." *Journal of Advances in Information Technology* 12, no. 3 (2021).