Cloud Computing and Big Data Systems
Course Project Report - "Exploring MovieLens 1M Dataset with Apache Spark"

I. Background and Objective

In today's information-rich and content-heavy environment, the volume of entertainment content generated is growing exponentially. This abundance of options makes it increasingly challenging for users to select what to consume. Consequently, there is a need to filter and narrow down the vast pool of content, presenting users with a curated selection that matches their preferences. Recommendation systems have been developed precisely to address this problem by leveraging existing data and user preferences to offer tailored recommendations.

Recommendation systems find extensive use in various domains, such as e-commerce platforms, music applications and social networking. With the enormous amount of content available, recommendation systems help personalize the user experience by suggesting items that align with the user's preferences, interests, and past behavior. This enhances user satisfaction and engagement. Besides, it facilitates the discovery of new and relevant content that users may not have been aware of otherwise. By analyzing user behavior, preferences, and patterns, these systems can recommend items that align with the user's interests but may not be immediately apparent to them.

In this project, we will first focus on doing a brief analysis, understanding how demographic information affected the user interest and then implementing two different recommendation systems by utilizing the movielens dataset from the grouplens site.

II. Cloud Computing Tools

Since the dataset contains one-million records, instead of processing data on our local computing system (e.g. Jupyter Notebook), Apache Spark is the framework we used for our visualizations and model building so that we can accelerate the process of visualization and model training. Also, we used Databricks as the data analytics platform to help us run the Spark workloads.

III. Advantages of Spark

In the era of cloud computing, Apache Spark has become one of the most popular open-source computing frameworks allowing developers to process and analyze more complex and multi-stage data with ease because of the two primary reasons as below[1].

The dynamic in nature yields better analytics
As a general-purpose cluster computing system, it offers numerous well-established libraries for machine learning, graph analytics algorithms and SQL queries with over 80 high-level APIs in Python, Java, and Scala. It also supports iterative, interactive and real-time ad-hoc data analytics beyond "map" and "reduce". The dynamic nature of Spark allows users to develop parallel applications effortlessly.

In-memory caching leads to fast processing
Furthermore, with its low-latency in-memory memory, it stores the data in RAMs to achieve quicker access and accelerate the speed of analytics. Not only can it be 100x faster than Hadoop for big data handling, it can process petabyte-scale clustered data of over 8000 nodes at once without downsampling.

IV. Data Description & Visualizations

The dataset utilized for this project is sourced from the GroupLens website, available at https://grouplens.org/datasets/movielens/1m/. It comprises a comprehensive collection of movie ratings. The

[1] https://www.knowledgehut.com/blog/big-data/apache-spark-advantages-disadvantages

dataset contains 3 files, encompassing 1 million ratings provided by 6000 MovieLens users who joined MovieLens in 2000 associated with approximately 4000 movies.

The "User.dat" file contains demographic information voluntarily provided by users, including UserID, Gender, Age, Occupation, and Zip-code. The "movies.dat" file includes titles that are identical to those provided by the IMDB, as well as genres and MovieID. All rating information, including UserID, MovieID, Rating, and Timestamp, is contained within the "ratings.dat" file.

Exploratory Data Analysis (EDA)

Age Distribution of Users

At the initial exploration of the user data, we analyzed the age distribution of the 6,000 users (Figure 1). Our findings revealed that 40% of the users fell within the age range of 25-34, making it the most prominent group. Approximately 20% of the users were aged between 35-44, representing the second-largest segment. Notably, users aged 18 and below accounted for only 3% of the total user population, which was the smallest proportion among all age groups.

Preferences on Genres

During our analysis of user preferences for movie genres, we observed the occurrence of 18 different genres (Figure 2). Among these genres, Comedy and Drama emerged as the most popular choices among users. They garnered the highest level of interest and engagement.

However, we noticed a significant drop in user interest for genres other than Comedy and Drama. The drop in popularity was particularly noticeable, indicating a substantial decrease in user preference for these genres. It suggests that the majority of users have a strong inclination towards Comedy and Drama, while other genres struggle to capture their attention.

Among the 18 genres, Documentary was found to be the least popular among users. It had the lowest occurrence and least user engagement compared to other genres.

This observation highlights the importance of Comedy and Drama genres in capturing and retaining user interest, while other genres may need to employ strategies to enhance their appeal and engagement among the user base.

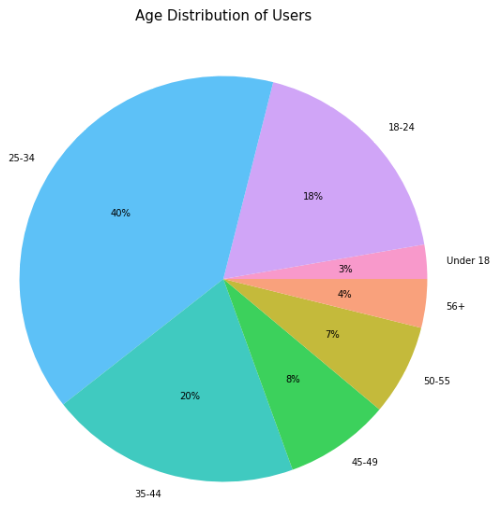


Figure 1

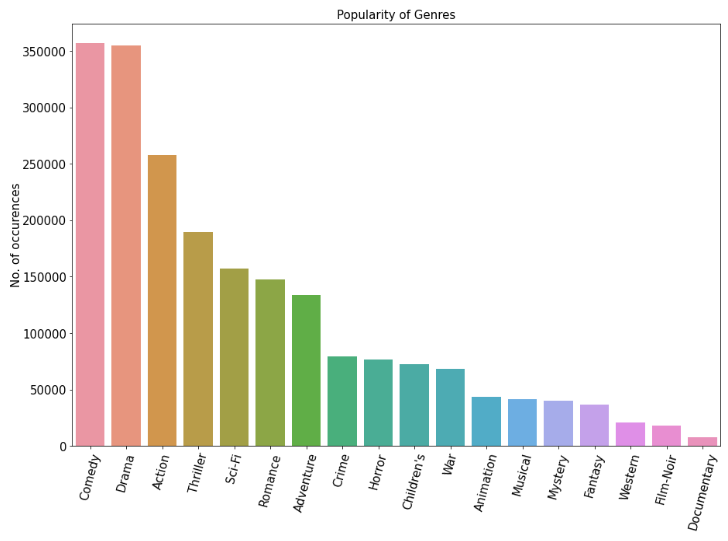


Figure 2

Rating analysis - Age

In our analysis based on age distribution across different genres, we made interesting observations regarding the preferences of different age groups for specific movie genres .

We found that genres such as Animation, Children's, and Fantasy were particularly popular among users aged 18 to 34. This suggests that these genres tend to be more engaging and appealing to younger audiences. On the other hand, the age group of 25 to 49 demonstrated a greater interest and affinity for Western and Film-noir genres.This indicates that these genres have a higher appeal and resonance with adult audiences. While certain genres may be more popular within specific age ranges,the 25-34 age group exhibits a broad interest in various movie genres. This suggests that individuals in this age bracket have a versatile cinematic appetite and are likely to appreciate diverse storytelling styles, themes, and genres.

These findings provide insights into the relationship between age groups and genre preferences. They suggest that certain genres have a stronger association with specific age demographics, with Animation, Children's, and Fantasy being favored by younger audiences and Western, and Film-noir finding more interest among adult viewers.

Rating analysis - Gender

Upon examining the rating distributions of different genres and genders, we discovered that there are notable differences between males and females in terms of their ratings.

In the genres of Animation and Musicals, females tend to provide more positive feedback compared to males. This suggests that females generally enjoy and appreciate these genres more, as reflected in their higher ratings. On the other hand, males tend to express more positive opinions about film noirs, indicating that it resonates more strongly with male audiences. Furthermore, when it comes to the Children's genre, females tend to give higher ratings compared to males. This indicates that females may have a greater affinity for movies targeted at children.

Overall, across various genres, most movie genres receive a rating of 4, indicating a generally positive reception from both males and females.

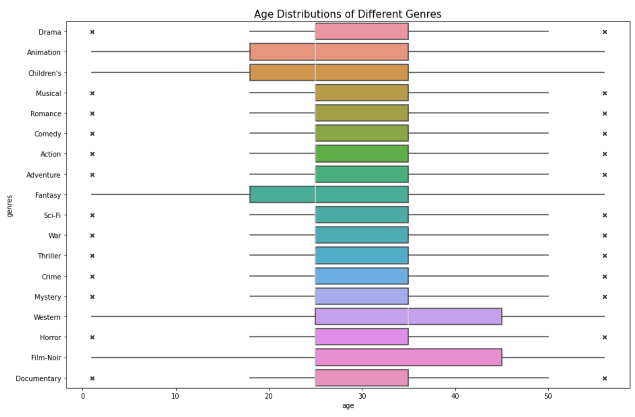


Figure 3

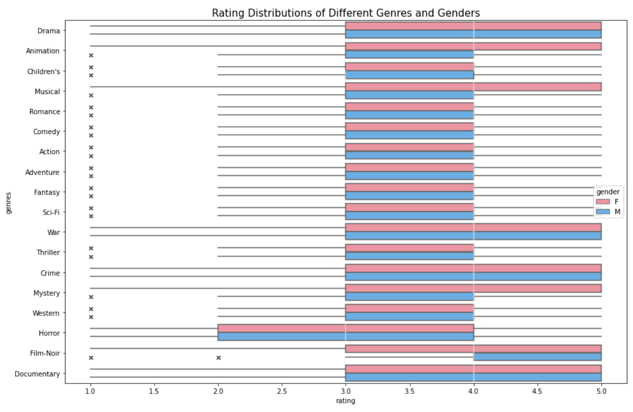


Figure 4

With the insights gained from the analyzed results, we were interested in leveraging the preferences and behaviors of users to make recommendations. The goal of reducing information overload and providing personalized recommendations can be achieved through employing collaborative filtering techniques in implementation of a recommendation system using the "rating.dat" file.

V. Recommendation Approaches & Algorithms

Using the rating dataset, we implemented two recommendation system tasks with Alternating Least Square Recommendation System and Correlation-based Collaborative Filtering.

Alternating Least Square Recommendation System (ALS)

Although the MovieLens dataset contains neither missing values nor extreme values, we leveraged the pyspark.sql module upon fetching the data set in the supervised approaches. We first defined the schema with the field name and data type specified. After that, we randomly split 80% of the dataset into the training set while the rest 20% went into the testing set.

The ALS algorithm aims to train a matrix factorization model with an RDD of ratings by users for a subset of products (i.e., movies) while the hyperparameters are the number of latent factors (i.e., the number of features to use, denoted by rank) and the regularization parameter denoted by regParam. To find the optimal values for them, we performed the grid search while building the model. Each round, we fitted the training set into the ALS model provided by MLlib, a machine learning (ML) library of Spark. The output, the ratings matrix, is close to the product of two lower-rank matrices given the number of features (rank). It is solved by ALS running iteratively with a configurable degree of parallelism. The prediction result was evaluated based on the MSE and RMSE between the training and test outcomes.

Correlation-based Collaborative Filtering

This model is more statistically based which computes the predicted ratings based on the ratings of similar users. The equation we used is as follows,

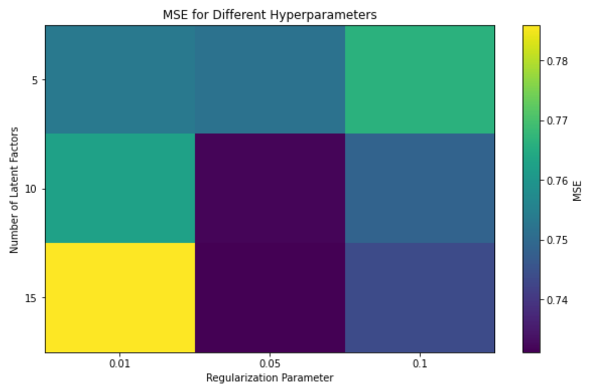
$$\boldsymbol{pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} |sim(a,b)|}}$$

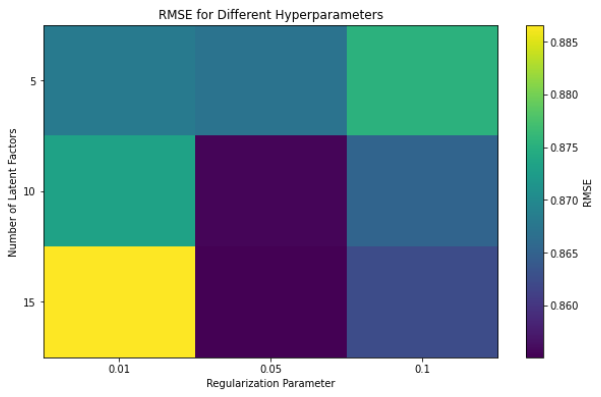
In the equation, we would like to predict the rating of movie *p* given by user *a*. $\overline{r_i}$ indicates the average ratings of user *i* and $r_{i,p}$ means the existing rating of movie *p* given by user *i*. $sim(a,b)$ is the Pearson Correlation Coefficient between user *a* and user *b* in terms of ratings.

Moreover, the predicted ratings computation was experimented with both pandas DataFrame and RDD. While the input RDD data was converted to pandas DataFrame in the former, the latter utilized the "map" and "reduce" approaches. Multiple map transformations were involved to compute the mean rating, Pearson Correlation Coefficient and the mean difference. The actual execution was triggered upon the aggregation of components shown in the equation.

VI. Result

Alternating Least Square Recommendation System (ALS)





The grid search evaluates the performance of the model on the test set for different combinations of hyperparameters. The combination with the lowest evaluation metric (MSE or RMSE) is considered the best. According to the graph above, the best result is achieved with a regularization parameter of 0.05 and 15 latent factors because these hyperparameters result in the lowest mean squared error (MSE) and root mean squared error (RMSE) on the test set.

Correlation-based Collaborative Filtering

pandas Dataframe	RDD with 1 partition	RDD with 2 partitions	RDD with 4 partitions	RDD with 6 partitions
3s	45.8s	26.9s	21.7s	24.3s

All the tasks were run with a 4-core CPU. The table above reveals the runtime needed by pandas and RDD in different partitions to compute the predicted ratings. It is interesting to observe that pandas Dataframe processed the data the most efficiently, which was 7-15 times faster than RDD with different partitions. Meanwhile, RDD with 4 partitions ranked the second-fastest. While increasing the number of partitions from 1 to 4 could significantly speed up, there was an uptrend of processing time from 4 to 6 partitions.

VII. Discussion & Findings

From the ALS Recommendation System

1 - The more the features the longer the runtime

The code is explicitly slow when the number of features is set to be more than 20 (which is not doable for the community edition databrick notebook). This is because the ALS algorithm used in collaborative filtering is computationally intensive and scales with the number of latent factors and the number of users and items in the dataset. Increasing the number of latent factors increases the complexity of the model and the size of the resulting matrices, which can lead to longer training times and increased memory usage.

2 - Iterative optimization needs a good balance between features and resources

Additionally, the ALS algorithm involves iterative optimization, which can require more iterations to converge as the number of latent factors increases. Therefore, it is important to balance the number of latent factors with the computational resources available in order to yield the desired level of accuracy.

From Correlation-based Collaborative Filtering

1 - Pandas works blazingly faster than spark for small datasets

Pandas Dataframe loads all the data into memory on a single machine for rapid computation. Spark, yet, performs parallel and distributed execution on all cores on multiple machines, which is supposedly able to handle complex data processing as data is growing exponentially. When Pandas computes faster than spark, this reflects the dataset we used for the project is relatively small-scale. Parallelization and distributed computing are not required. Thus, Pandas Dataframe can process such microscale data instantly without any performance degradation while Spark struggled due to its complexity.

2 - Partitioning data doesn't always equal faster.

According to the result, we believe using RDD with 4 partitions will be the most optimal configuration.Splitting data into multiple partitions does allow transformations to be executed in parallel and utilize the cores available in the cluster. The jobs can then be completed more rapidly. Nevertheless, having too many partitions may cause task scheduling and shuffling to take much longer than the actual execution time. Also, as we run the distributed tasks on one local machine, when the number of partitions exceeds the optimal, excessive overhead may be caused in managing small tasks.

VIII. Source Code:

- MovieLens 1M Dataset: https://grouplens.org/datasets/movielens/1m/