Final Project

Recommendation Engine for Recommending Books using Big Data Approach

Instructor Erton (Tony) Boci, Ph.D

George Mason University, Fairfax.

ECE 552 – Big Data Technologies

Department of Electrical and Computer Engineering

Team 09:

Prakruti Rajendra Kothari

Shweta Bhosale

Narasing Sarwade

Recommendation Engine for Recommending Books using Big Data Approach

Prakruti Rajendra Kothari
G01345316
pkothari@gmu.edu

Narasing Sarwade
G01290366
nsarwade@gmu.edu

Shweta Bhosale
G01338086
sbhosal@gmu.edu

Abstract: The main goal of this project was to build and evaluate a large scale book recommendation system using the goodbooks-10k dataset and pyspark. In the recent times, there has been a growing interests in the area of recommender systems using machine learning techniques. In this study, we implemented a book recommendation system based on a collaborative filtering approach using the alternating least squared (ALS) model to recommend a user number of books based on their likings and rating. In particular, we performed experimental analysis and successfully obtained minimum root mean squared errors (RMSE) of 0.88 approximately

Keywords: Pyspark, Apache Spark, Jupyter Notebook, ALS, Collaborative Filtering.

I. Introduction

Internet users are growing daily as a result of the Internet's rising accessibility. As a result, there is an increase in the volume of data sent over the Internet. Data overload has resulted from this, where users are overloaded with knowledge and information.The rapid growth in data has led to a new era of information. These data are used to build innovative, more efficient, and effective systems. In this era of big data, the recommender engine is a category among data sieving systems, which aims to predict the ratings a user will assign to objects of interest over the Internet [1].

Recommender systems have become an emerging research area with the growth of e-commerce and web services including advertisements for providing personalized recommendations to a user. For example, Amazon, one of the largest online retailers in the world, has been using robust data-driven analytics to optimize its market growth for over 304 million active customers with 12.2 million products carried. [2]. The recommender engine is a type of data sieving system that aims to anticipate the ratings a user will give to items of interest on the Internet in this age of big data. When clients shop, browse, and leave reviews recommendation system usually offers them product recommendations based on their past search history and likings. For enhancing the outcome of a recommender system, many researchers have been developed recommender algorithms that utilize both explicit and implicit user feedbacks [3].

The two most widely used methods in recommendation systems are content-based algorithms and collaborative filtering memory-based methods. These are a set of methods which use the entire database (user-item database) to make a recommendation to a user. These methods can also be classified into two types: Item-based collaborative filtering: In this particular method, two items are compared and given a similarity score between the two using some sort of metrics like the cosine distance.The primary benefit of a recommender system is that it gives the user a personalized perspective of the data. Machine learning, data mining, and other artificial intelligence techniques are all used in the research on recommendation systems. Our recommendation engine was built using the Collaborative Filtering and the alternating least squared (ALS) method by using Spark machine learning (ML) libraries.

II. Literature Review

Recently, machine learning (ML)-based models [4] [5] [6] have been implemented in various domains, which discussed time limitations, specification, accuracy with efficacy, and intricate issues. Auto analysis, high adaptability, and user data collection from big data environments through recommendation, however, were potential strengths.The experiment's findings demonstrated how Apache Spark and machine learning gather information on viewers' preferences and thought processes, and how this information is then used to produce highly precise recommendations. For user convenience, the recommender system enables users to choose items from the plethora of options on social service applications like Netflix and YouTube [7].

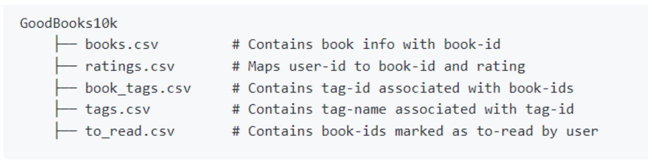
In 2021, the authors implemented stock market [8], Black Friday sale [9] and Amazon food review [10] predictions using Naive Bayes, and other machine learning models on Spark MLlib achieved above 90% accuracy with a good running time. With the use of various data mining algorithms, the author created a system that demonstrated how useful information could be extracted from enormous amounts of raw data. The experimental results showed authentic results with larger datasets in all circumstances if accurate models were obtained [11]. A small enhanced application of the suggestion system is the integration of the analytics of big data, whose base was built upon the Spark framework with several combined CF algorithms, as data is increasing day by day. To analyze these enormous volumes of datasets and for the user's convenience, this application uses the analytics of big data.

Our suggested Recommendation System is based on collaborative filtering through the big data framework Spark, which is derived from user-level and item-level filtering. Despite the size of the dataset, we used ALS to group the ratings into an item- and user-based matrix along with latent features.

III. Dataset

A. *About the Dataset*

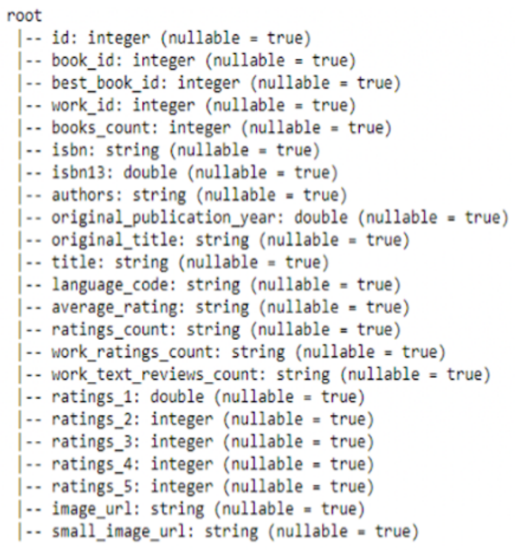
The goodbooks-10k dataset was retrieved from kaggle. The dataset is collected from the goodbooks-10k website [12]. The dataset consists of about 1 million ratings for around 10,000 most popular books. In most cases, there are at least 10 books rated by each user and the rating lies between 0 and 5.These one million ratings will be used to forecast ratings for all the other books that users haven't read yet. Our dataset consists of five csv files.



```
GoodBooks10k
    ├── books.csv          # Contains book info with book-id
    ├── ratings.csv        # Maps user-id to book-id and rating
    ├── book_tags.csv      # Contains tag-id associated with book-ids
    ├── tags.csv           # Contains tag-name associated with tag-id
    ├── to_read.csv        # Contains book-ids marked as to-read by user
```

Fig. 1. Dataset files

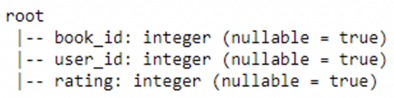
*1) **books.csv** :* The books.csv has metadata for each book (goodreads IDs, authors, title, average rating, etc.). The metadata have been extracted from goodreads XML files, available in books_xml.



```
root
 |-- id: integer (nullable = true)
 |-- book_id: integer (nullable = true)
 |-- best_book_id: integer (nullable = true)
 |-- work_id: integer (nullable = true)
 |-- books_count: integer (nullable = true)
 |-- isbn: string (nullable = true)
 |-- isbn13: double (nullable = true)
 |-- authors: string (nullable = true)
 |-- original_publication_year: double (nullable = true)
 |-- original_title: string (nullable = true)
 |-- title: string (nullable = true)
 |-- language_code: string (nullable = true)
 |-- average_rating: string (nullable = true)
 |-- ratings_count: string (nullable = true)
 |-- work_ratings_count: string (nullable = true)
 |-- work_text_reviews_count: string (nullable = true)
 |-- ratings_1: double (nullable = true)
 |-- ratings_2: integer (nullable = true)
 |-- ratings_3: integer (nullable = true)
 |-- ratings_4: integer (nullable = true)
 |-- ratings_5: integer (nullable = true)
 |-- image_url: string (nullable = true)
 |-- small_image_url: string (nullable = true)
```

Fig. 2. Schema for books.csv

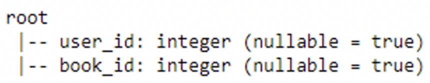
*2) **ratings.csv** :* There are 981756 ratings in the dataset. Each rating consists of user ID, book ID, and a rating score from 1 to 5.Numbers of ratings per book range from 8 to 22,806, and ratings by a user range from 1 to 200.



```
root
 |-- book_id: integer (nullable = true)
 |-- user_id: integer (nullable = true)
 |-- rating: integer (nullable = true)
```

Fig. 3. Schema for ratings.csv

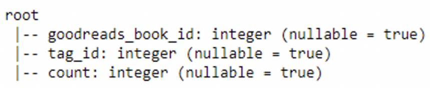
*3) **to_read.csv**:* There are 912,705 book-user pairs in this dataset. Books to read adds a a strong signal to ratings because 99.86% of all books are in someone's to read list and 91.48% of all users have books in their "to read" list.



```
root
 |-- user_id: integer (nullable = true)
 |-- book_id: integer (nullable = true)
```

Fig. 4. Schema for to_read.csv

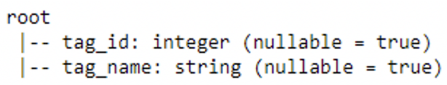
*4) **book_tags.csv** :* The book_tags.csv file contains tag_id, book_id and count information. Tags in this file are represented by their IDs. They are sorted by goodreads_book_id ascending and count descending.



```
root
 |-- goodreads_book_id: integer (nullable = true)
 |-- tag_id: integer (nullable = true)
 |-- count: integer (nullable = true)
```

Fig. 5. Schema for book_tags.csv

*5) **tags.csv** :* The tags.csv file translates tag IDs to their names.

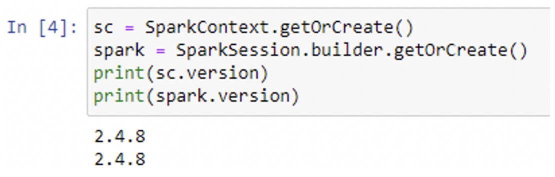


```
root
 |-- tag_id: integer (nullable = true)
 |-- tag_name: string (nullable = true)
```

Fig. 6. Schema for tags.csv

B. *Hardware and Software Requirements*

For the project implementation we used Windows 10 Pro was installed on a virtual machine in Microsoft Azure specifically for this purpose. The fixed disk of the virtual machine contained 128GB of storage space, 6 CPU cores, and 16GB of RAM. In terms of software, Apache Spark version 2.4.8 was used:



```
In [4]: sc = SparkContext.getOrCreate()
        spark = SparkSession.builder.getOrCreate()
        print(sc.version)
        print(spark.version)

        2.4.8
        2.4.8
```

Fig. 7. Spark Version

Furthermore, Apache Parquet was used for data transformation. Jupyter notebook was used for data visualization using various python libraries such as pandas, sns, matplotlib and numpy. It was also used for building the model using the Spark ML.

C. *Data Flow*

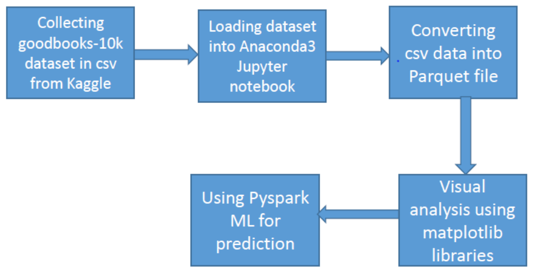

Collecting goodbooks-10k dataset in csv from Kaggle
Loading dataset into Anaconda3 Jupyter notebook
Converting csv data into Parquet file
Visual analysis using matplotlib libraries
Using Pyspark ML for prediction

Fig. 8. Dataflow

D. *Data Architecture*

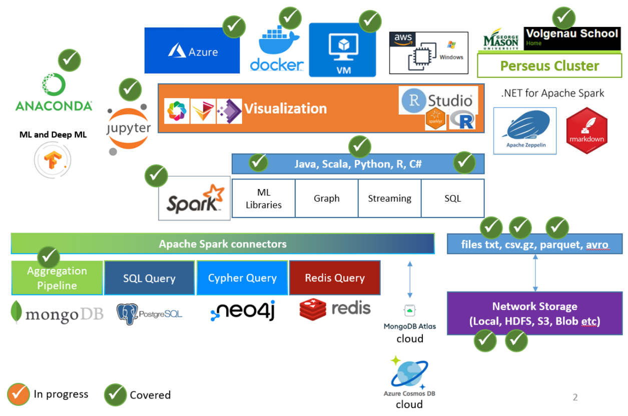
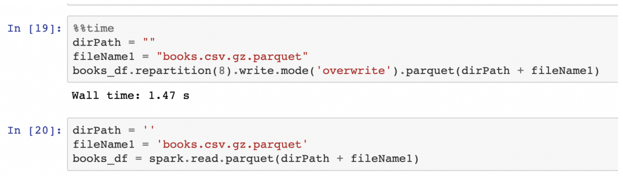


Fig. 9. Data Architecture

IV. Methodology

A. *Loading the Data*

The goodbook-10k dataset was downloaded in csv format from kaggle. We then created a python notebook in the Jupyter notebook. The Spark session was started on the jupyter notebook and then read the csv file using spark.read.csv(). The csv file was further converted into the parquet file using write.parquet() and re-read as parquet using spark.read.parquet(). Once the file was converted to parquet format we count the number of entries.



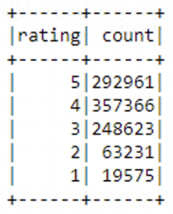
```
In [19]: %%time
         dirPath = ""
         fileName1 = "books.csv.gz.parquet"
         books_df.repartition(8).write.mode('overwrite').parquet(dirPath + fileName1)

         Wall time: 1.47 s

In [20]: dirPath = ''
         fileName1 = 'books.csv.gz.parquet'
         books_df = spark.read.parquet(dirPath + fileName1)
```

Fig. 10. Changing csv file to parquet file.

B. *Exploratory Data Analysis*

To understand our data more clearly we did some Exploratory analysis for our dataset. Some of the analysis that we did were:

*1) **Number of each Ratings**:* The first analysis that we did was to calculate count for each ratings using the groupBy function. And from the analysis we found that rating 4 had the highest entries count and rating 1 had the least number of entries.



```
+------+------+
|rating| count|
+------+------+
|     5|292961|
|     4|357366|
|     3|248623|
|     2| 63231|
|     1| 19575|
+------+------+
```

Fig. 11. Count of each rating

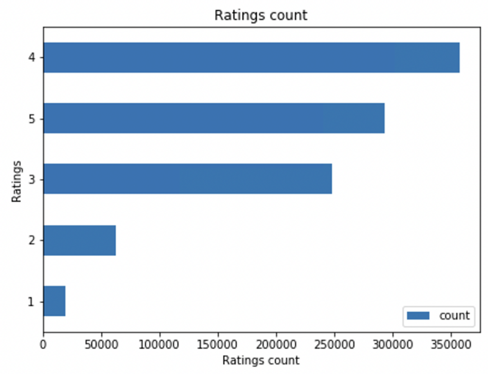


Fig. 12. Count for each rating

*2) **Authors with most number of books**:* Next analysis that we did was to check top 10 authors with the highest number of books and from the analysis we see that ”Stephen King ” author had the highest number of books count i.e 60.

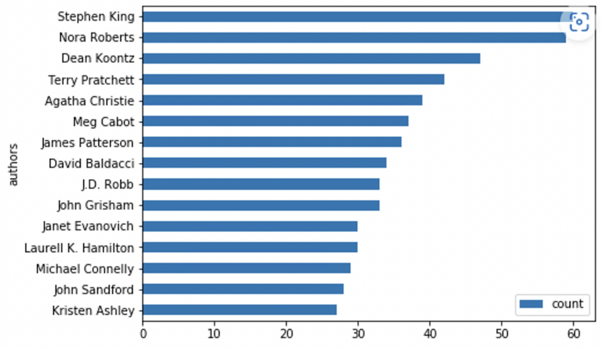
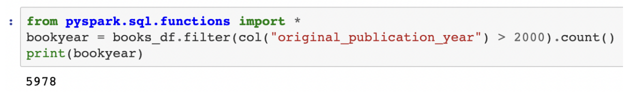


Fig. 13. Authors with most number of books

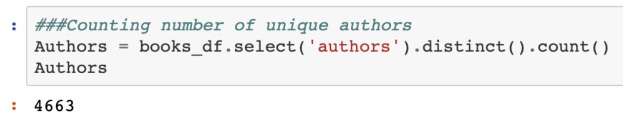
*3) **Number of books published after year 2000**:* Using the spark SQL, the number of books published after the year 2000 were calculated. And it was found that 5978 number of books were published after 2000 year.



```
from pyspark.sql.functions import *
bookyear = books_df.filter(col("original_publication_year") > 2000).count()
print(bookyear)

5978
```

Fig. 14. Books published after 2000 year

*4) **Number of unique authors**:* Using the books meta data file, we then calculated total number of unique authors and we found that there were 4663 total unique authors in our dataset.



```
###Counting number of unique authors
Authors = books_df.select('authors').distinct().count()
Authors

4663
```

Fig. 15. Number of unique authors

*5) **Average ratings for First 10 books**:* The average ratings count was determined using the rating data file. We discovered that the rating with the highest average number of counts was 3.94 with a count of around 175 books.

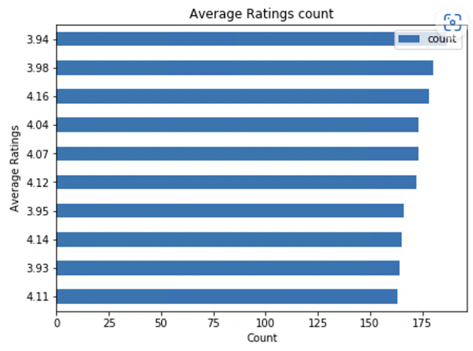
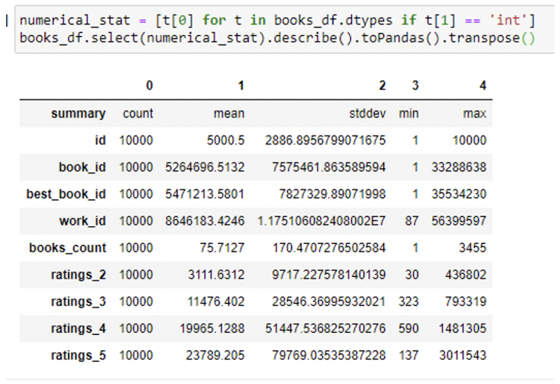


Fig. 16. Average Rating count

*6) **Numerical Column Statistics**:* Next we extracted only the numerical columns from the books data file and the summary statistics like mean, count, standard deviation, minimum and maximum values were generated for these numerical columns.



```
numerical_stat = [t[0] for t in books_df.dtypes if t[1] == 'int']
books_df.select(numerical_stat).describe().toPandas().transpose()
```

	0	1	2	3	4
summary	count	mean	stddev	min	max
id	10000	5000.5	2886.8956799071675	1	10000
book_id	10000	5264696.5132	7575461.863589594	1	33288638
best_book_id	10000	5471213.5801	7827329.89071998	1	35534230
work_id	10000	8646183.4246	1.175106082408002E7	87	56399597
books_count	10000	75.7127	170.4707276502584	1	3455
ratings_2	10000	3111.6312	9717.227578140139	30	436802
ratings_3	10000	11476.402	28546.36995932021	323	793319
ratings_4	10000	19965.1288	51447.536825270276	590	1481305
ratings_5	10000	23789.205	79769.03535387228	137	3011543

Fig. 17. Numerical column Statistics

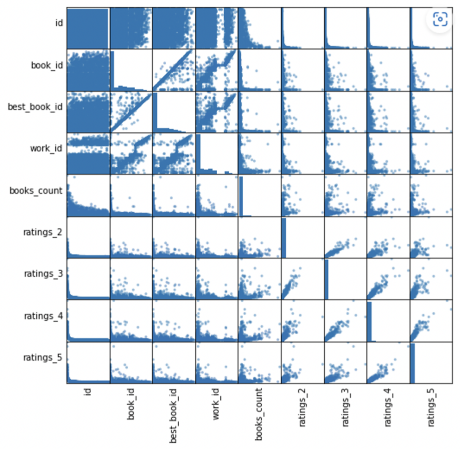
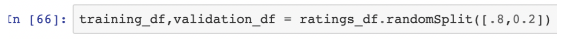


Fig. 18. Numerical column Statistics

C. *Building the Model*

The next step after the data exploration was to implement a machine learning model for the predictions. We used collaborative filtering technique, which is based on ALS, i.e., a matrix factorization algorithm that is used to predict the top books that will be recommended to the users later.

*1) **Data Splitting**:* The ratings dataset was randomly split into training and validation sets to use collaborative filtering using the Alternate Least Square method. The training dataset was split into 80% and validation dataset into 20%.



```
In [66]: training_df,validation_df = ratings_df.randomSplit([.8,0.2])
```

Fig. 19. Data Splitting

*2) **Collaborative Filtering**:* Usually, recommenders system uses collaborative filtering. Matrix factorization is used in the method to forecast the missing entries. According to this study, missing entries can be predicted when users or products are modeled by latent factors and the rating for books is implemented using Spark-ML and ML-libraries, which currently use the model-based collaborative scheme. To learn more about these latent factors, we also used the ALS (alternating least squares) algorithm.

*3) **Alternating least square model**:* This research is based on machine learning and the ALS algorithm, which is a form of regression analysis similar to regression analysis in that regression presents a line of data points that are used to draw through regression. Alternating Least Squares (ALS) matrix factorization decouples a rating matrix R into two lower-ranked matrices, a user_id matrix U and a book matrix B. The objective is to anticipate these "empty cells" in the rating matrix R, which is by nature sparse. In order to identify a square loss minimizer with regard to the other matrix during training, we hold one matrix fixed and repeatedly swap the two matrices.

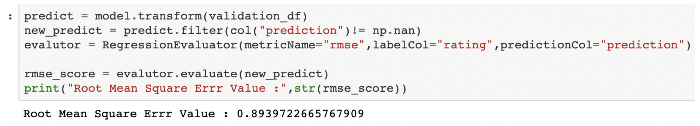
The ALS algorithm has its own unique set of hyper-parameters, just like other machine learning algorithms. We should probably use hold-out validation or cross-validation to adjust its hyper-parameters. Most important hyper-params in Alternating Least Square (ALS):

- maxIter: the maximum number of iterations to run.
- rank: the number of latent factors in the model.
- regParam: the regularization parameter in ALS.

An efficient recommendation engine provides users with personalized predictions quickly and accurately. These systems are becoming more and more valuable to business. Using algorithms for collaborative filtering that locate user commonalities, we can make predictions. ALS is a feature of Apache Spark ML that is implemented for large-scale collaborative filtering tasks. ALS, which is simple and scales well to very large datasets, is used to successfully address the scalability and sparsity of the Ratings data.

V. Results

In this study, the Alternating Least Square (ALS) method is used to train the recommender system. The online book recommendation system was assessed using statistical accuracy metrics called Root Mean Square Error (RMSE). The difference between the predicted value and the actual user rating is measured by RMSE. The purpose of the training was to reduce the value of RMSE, and the ALS model with the lowest RMSE value is saved for future recommendations. The calculated RMSE value for this recommendation system was 0.8939.

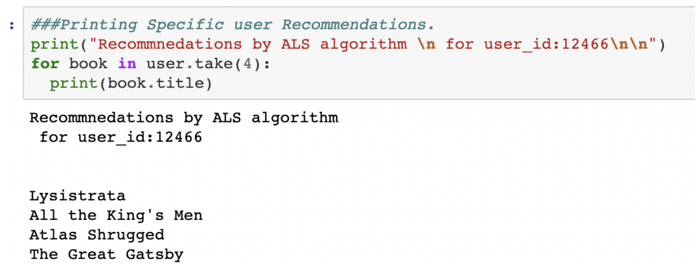


```
predict = model.transform(validation_df)
new_predict = predict.filter(col("prediction")!= np.nan)
evalutor = RegressionEvaluator(metricName="rmse",labelCol="rating",predictionCol="prediction")

rmse_score = evalutor.evaluate(new_predict)
print("Root Mean Square Errr Value :",str(rmse_score))
```

```
Root Mean Square Errr Value : 0.8939722665767909
```

Fig. 20. RMSE Value

Once we implement the ALS recommender system in the jupyter notebook using the Pyspark program, we predicted all the books for all the user_ids. After the prediction was done we can generate and print all the recommended books for a particular user_id using this recommendation system.



```
###Printing Specific user Recommendations.
print("Recommnedations by ALS algorithm \n for user_id:12466\n\n")
for book in user.take(4):
  print(book.title)
```

```
Recommnedations by ALS algorithm
 for user_id:12466


Lysistrata
All the King's Men
Atlas Shrugged
The Great Gatsby
```

Fig. 21. Recommended books for a user

For instance, our ALS algorithm provides top 4 book recommendations for user_id :12466. As shown in the Figure 21. for user_id :12466, the recommended books are: Lysistrata, All the Kings's Men, Atlas Shrugged, The Great Gatsby.

VI. Use cases of Recommendation Systems

Algorithms for recommender systems are used by large corporations to offer a more personalized and interesting customer experience. As a result, recommenders increase customer retention and revenue, assisting businesses in achieving their objectives. Below mentioned are few popular recommendation engine used by some known companies [13]:

- Youtube: YouTube uses a sophisticated two-stage recommender system. Their candidate generation network provides broad personalization via collaborative filtering. A small subset (hundreds) of videos from a large corpus is retrieved [13].
  Given that they are viewed as being extremely relevant to the user, events from the user's YouTube activity history are used as input. The identification of watched videos, search query tokens, demographic information, and other similarity indicators are used to identify and express users [13].
- Facebook : Due to the enormous amount of data being analyzed, Facebook cannot employ the conventional ML approach, therefore sampling is not an option. The Facebook open-source PyTorch and Caffe2 frameworks are combined with a cutting-edge deep learning open source recommendation model (DLRM) [13].
  In order to generate recommendations based on the likes and interactions of people with similar tastes, collaborative filtering and predictive analytics-based methodologies are used. As a result, users find the most pertinent topics through specifically selected content that is recommended [13].

VII. Conclusion

Apache Spark is ideal for applications that require fast data querying, transformation, and analytics outputs. As a result, the recommendation system created in this study is built on Apache Spark. A recommendation system for book recommendation is constructed in this study using the alternating least squares (ALS) matrix factorization approach on Apache Spark MLlib. The research reveals that applying the ALS matrix factorization method greatly reduces the RMSE value, which is 0.893. As a result, it is demonstrated that the ALS algorithm is appropriate for training explicit feedback data sets in which users submit ratings for books. The proposed recommendation system architecture is based on the lambda architecture, which allows users to see recommendations in real time.

The results led to the conclusion that the alternating least squares parameter selection could have an impact on Recommendation System performance. As a result, a system needs a backup to remember user preferences in order to remember and collect accurate and thorough data. In the future, sieves can be lengthened for significantly better outcomes.

References

[1] M. J. Awan, R. A. Khan, H. Nobanee, A. Yasin, S. M. Anwar, U. Naseem, and V. P. Singh, "A recommendation engine for predicting movie ratings using a big data approach," *Electronics*, vol. 10, no. 10, p. 1215, 2021.

[2] L. Chen, R. Li, Y. Liu, R. Zhang, and D. M.-k. Woodbridge, "Machine learning-based product recommendation using apache spark," pp. 1–6, 2017.

[3] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, pp. 109–132, 2013.

[4] Y. Ali, A. Farooq, T. M. Alam, M. S. Farooq, M. J. Awan, and T. I. Baig, "Detection of schistosomiasis factors using association rule mining," *IEEE Access*, vol. 7, pp. 186 108–186 114, 2019.

[5] M. Gupta, R. Jain, S. Arora, A. Gupta, M. Javed Awan, G. Chaudhary, and H. Nobanee, "Ai-enabled covid-19 outbreak analysis and prediction: Indian states vs. union territories," *Gupta, M., Jain, R., Arora, S., Gupta, A., Awan, MJ, Chaudhary, G., & Nobanee, H.(2021). AI-Enabled COVID-19 Outbreak Analysis and Prediction: Indian States vs. Union Territories. Cmc-Computers Materials & Continua*, vol. 67, no. 1, pp. 933–950, 2021.

[6] M. Anam, M. Hussain, M. W. Nadeem, M. Javed Awan, H. G. Goh, S. Qadeer *et al.*, "Osteoporosis prediction for trabecular bone using machine learning: a review," *Computers, Materials & Continua (CMC)*, vol. 67, no. 1, 2021.

[7] S. Qin, R. Menezes, and M. Silaghi, "A recommender system for youtube based on its network of reviewers," pp. 323–328, 2010.

[8] M. Javed Awan, M. S. Mohd Rahim, H. Nobanee, A. Munawar, A. Yasin, and A. M. Zain, "Social media and stock market prediction: a big data approach," *MJ Awan, M. Shafry, H. Nobanee, A. Munawar, A. Yasin et al.," Social media and stock market prediction: a big data approach," Computers, Materials & Continua*, vol. 67, no. 2, pp. 2569–2583, 2021.

[9] M. Javed Awan, M. S. Mohd Rahim, H. Nobanee, A. Yasin, and O. I. Khalaf, "A big data approach to black friday sales," *MJ Awan, M. Shafry, H. Nobanee, A. Yasin, OI Khalaf et al.," A big data approach to black friday sales," Intelligent Automation & Soft Computing*, vol. 27, no. 3, pp. 785–797, 2021.

[10] H. M. Ahmed, M. Javed Awan, N. S. Khan, A. Yasin, and H. M. Faisal Shehzad, "Sentiment analysis of online food reviews using big data analytics," *Hafiz Muhammad Ahmed, Mazhar Javed Awan, Nabeel Sabir Khan, Awais Yasin, Hafiz Muhammad Faisal Shehzad (2021) Sentiment Analysis of Online Food Reviews using Big Data Analytics. Elementary Education Online*, vol. 20, no. 2, pp. 827–836, 2021.

[11] E. Hernández-Nieves, J. Parra-Domínguez, P. Chamoso, S. Rodríguez-González, and J. M. Corchado, "A data mining and analysis platform for investment recommendations," *Electronics*, vol. 10, no. 7, p. 859, 2021.

[12] "goodbooks-10k," https://www.kaggle.com/datasets/zygmunt/goodbooks-10k.

[13] "goodbooks-10k," https://datarootlabs.com/blog/recommender-systems-use-cases-choosing-one-for-your-business#what-is-a-ml-recommender-system-and-how-does-it-work.