E-Commerce Product Recommendation Engine: Enhancing Customer Online Shopping Experience

J. Sri Sai Samhitha[1]
Dept. of Computer Science and Engineering,
Amrita School of Computing, Bengaluru,
Amrita Vishwa Vidyapeetham, India.
bl.en.u4cse21072@bl.students.amrita.edu

K. Adarsh Sagar[2]
Dept. of Computer Science and Engineering,
Amrita School of Computing, Bengaluru,
Amrita Vishwa Vidyapeetham, India.
bl.en.u4cse21075@bl.students.amrita.edu

Mohammed Jaffer Ali[3]
Dept. of Computer Science and Engineering,
Amrita School of Computing, Bengaluru,
Amrita Vishwa Vidyapeetham, India.
bl.en.u4cse21124@bl.students.amrita.edu

Supriya M[4]
Dept. of Computer Science and Engineering,
Amrita School of Computing, Bengaluru,
Amrita Vishwa Vidyapeetham, India.
m_supriya@blr.amrita

***Abstract*—The central culmination of the project is an E-Commerce Product Recommendation System for enhancing customer satisfaction using diverse concrete machine learning algorithms. The recommendation engine uses a dataset containing 7 million rows, with four features: User ID, Product ID, a value which would be ranging from 1 to 5, Rating and Timestamp. One of its main objectives is to suggest to the users some products which could be of interest for them according to their activity history or the services which increase the level of participation and sales. It studies and compares the findings of the following algorithm-based approaches; KNN, KNN-Pearson Coefficient, NMF, SVD, SVD++, and Linear Regression. A quantitative evaluation of these algorithms is performed by implementing these algorithms with and without PySpark to demonstrate the role of distributed computation in further improving the performance of the task. The Accuracy, Precision, Recall, F1 score, and the confusion matrix are the parameters used with the view to evaluating the different models in their ability to predict the user preferences with regards to the items.**

***Index Terms*—E-Commerce Recommendation, PySpark, Machine Learning Algorithms, Model Based Collaborative Filtering, Machine Based Collaborative Filtering.**

I. Introduction

The emergence of E-Commerce has greatly revolutionized how people shop with full product lists and unlimited choices provided to customer. This has raised the call for higher intelligent recommendation techniques to enhance the customer's shopping experience. A recommendation engine has the capability to predict what a particular consumer might want based on his or she past activities, choices, and trends. Here in this proposed project, product recommendation is going to be formed in order to provide better customer engagement, higher conversion rates, and satisfaction level among customers. To this end, we utilize a number of machine learning methods, which provide different input-output mappings as methods for analyzing the interaction of user and items and their desirable characteristics.

The data set Used for this project is greater than 7 million records which records the user id, product id, rating and time stamp of interactions. Virtually, some critical issues emerge with the large dataset, including; Sparsity of the User-Item matrix; Computational complexity concerning the scale of the data. To tackle these issues, we adopt several ordinary collaborating filtering methodologies including K-Nearest Neighbors (KNN), Non – Negative Matrix Factorization (NMF), and Singular Value Decomposition (SVD). They have been adopted in prediction of user preferences and the above mentioned algorithms have advantages and draw backs of prediction accuracy based on aspects of computational complexity and scalability. We shall then compare these methods with the view of determining the most effective technique for enhancing recommendations within an E-Commerce environment.

The project also assesses how PySpark Performance optimization influences the recommendation system algorithms in a distributed computing environment. PySpark can handle large amount of data and is built to work with big data which will make the computation time for frequent training and prediction faster when working with data of millions of rows. To determine the degree of efficiency increase and scalability of the algorithms, we show the results of comparison of the execution time of algorithms using PySpark with algorithms that do not use it. Besides, based on the data gathered, the performances of the algorithms are quantitatively compared employing accuracy, precision, recall, the F1 score, and the confusion matrix. This comparative analysis assists in establishing which of the algorithms and frameworks provide the best solution for improving the user experience for E-Commerce based platforms. To sum up, the development of fixed recommendation system is considered to be vital to

the improvement of the online-shopping system field. This project makes conclusions about strengths and weaknesses of KNN, SVD, NMF, linear regression and similar methods, when they applied with and without PySpark. Besides, the primary objective of this article is not only to reveal the best practices of product recommendation but also to show how the system can be improved to work better with large data sets. By comparing the algorithms in the study, this project shall provide real solutions for firms who wish to make customer interactions unique to their markets, product recommendation correct to the market and indeed make sales in this competitive world.

II. Literature Review

This section give an overall view of the insights obtained by analyzing various papers related to E-commerce recommendation systems using Machine Learning Algorithms and leveraging PySpark environment.

Padmavathi et al. [1]The paper critically reviews the ability of movie recommendation systems, particularly by utilizing hybrid approaches. It combines collaborative filtering and content filtering for an inclusive analysis of the results. Using Netflix dataset, it measures the stability of the Matrix Factorization Technique. The investigation compares models in terms of accuracy, efficiency, and the ability to withstand higher level perturbation . Outcomes affirm that the hybrid models outcompete the conventional models in terms of precision and speed. Furthermore, the research re-establishes the efficacy of the machine learning method known as the XGBoost model which uses 13 feature for better prediction results. Therefore, the research work is useful in providing a bench mark for scholars and practitioners when choosing the best recommendation models to be implemented.

Zhang et al. [2] A new approach to the collaborative filtering recommendations is proposed, using probability matrix factorization to predict the rates by the extra-sbsolute decomposition of the rate matrix. Through normalization, we are able to obtain reasonable probabilistic results, and further, variational inference helps in evaluating the posterior distribution, which allows accurate prediction for those items that have not been rated. The integration of temporal weighting into the user-item-time boosts the potential of the algorithm, and studies have shown an improvement in functionality in datasets of Netflix, movielens, Epinion and so on.This improves accuracy, reduces complexity and integrates temporal aspects. Indicator cosine similarity is replaced by the adjusted cosine similarity while for removing temporal bias, Ebbinghaus forgetting curve is used. To sharpen the accuracy of the predictions, characteristic matrices have been initialized with Gaussian distribution and a method for computing free parameters has been incorporated. It shows remarkable capacity to handle the sparsity of data, especially of the midterm rating, and provides precision and recall of better performance than the other similar systems.

Deng et al. [3] The attitudes of users in relation to the recommendation systems related to the article according to the difference between the mature and the emerging trends which are temporary shifts in the context. The complex deep learning techniques are used to capture the user behavior dynamics across the length of its usage and the incorporation of side information handle the cold start and data sparse issues. Non-synchronous communicative microblogs offer real-time social signals to improve recommendations; latent semantic analysis and its relatives identify indirect user-item associations. The paper surveys graph embedding that enhances the accuracy and the scalability, in addition to occasional collaborative filtering and diffusion based available for predicting the user-item similarity. The explicit and implicit interactions are examined for content-based approaches and matrix factorization. Bayesian methods provide the probability of user's behaviour, while experimental analysis states that conventional models are more efficient and better in cold start problems than graph embeddings. In sum, KGEs are a remarkably promising recommendation system since it identifies new approaches towards improving the precision and capacity of recommendation systems.

Chaudhari et al. [4] This paper reviews travel RSs, covering areas including travel planning; transport; accommodation; weather; and attractions. It also shows how RSs help users to arrange the tours, to select the proper packages, and to make a choice of the hotel, restaurant and other tourists' destinations. The study categorises travel-based RSs based on their characteristics and identifies the weaknesses such as ineffectiveness and non-adaptability of them; this is salient to note that most of the RSs are needed to cater the preferences of multi-tourists. Specific state-of-art approaches elaborated are multicriteria collaborative filtering for tourism services and adaptive neuro-fuzzy inference system (ANFIS) for recommending hotel. The given problem is tried to be optimized by using clustering algorithms such as EM and by reducing the higher dimensionality with the help of PCA algorithm is implemented and item-based collaborative filtering is used for hotel rating prediction. It would also show how frameworks like TourSense achieve macro and micro F1 scores of 0.8549 and 0.7154 in refining recommendation precision, illustrating novel strategies for improving travel RSs.

Paula et al. [5] The paper presents a new dataset on mobile app installations from the Aptoide market comprising more than 4 millions users' installation/uninstallation data. With regards it benchmarks collaborative filtering by naturally exploring user preferences from sparse and noisy explicit ratings. Three snapshots of Aptoide establish user engagement over time providing interest in specific apps and facilitating performance assessment of recommender models. Methods examined include SVD-based, NPF-based, and KNN-based methods, Slope One, and neighborhood-based approach. In the SVD and KNN models, a high value was recorded in the Precision, Recall, and F1-score; however, Slope One was significantly lower than in Baseline only. It has also shown how the used dataset and the computational performance of the investigated models are potentially useful for improving mobile app recommender systems.

Anwar et al. [6] This work discusses the topic of adapting recommender systems (RS) for individual use, with an overview ofRS approaches based on users' preferences and ratings. It comes up with a technique of recommendation by using a hybrid of standalone collaborative filtering and SVD++ techniques especially in recommending movies. The proposed approach is then compared with traditional methods such as, K-NN, SVD, and Co-clustering. The experiment shows that SVD++-based method have better results compared to the other solutions obtaining lower errors RMSE/MAE equal to 0.9201; 0.7219. Compared to other methods, this method performs better especially when dealing with cold-start and data sparsity problems, which are in turn make this method more reliable for generating recommender system. Thus, the presented approach of assisting users in dealing with large amounts of information and presenting them with only relevant products contributes to a rather large increase in the performance of RS. The advantage of the proposed technique has been ascertained by its capability of outcompeting other variants in effectively dealing with the usual problems associated with recommender systems and, therefore, of improving the recommendation quality.

Saini et al. [7] This paper discusses the recommendation system of e-commerce based on the collaborative filtering methods: Comparison is made on the basis of rating prediction standards and execution time. For proper evaluation of the results, the data was divided as training data which was 80% and the testing data was 20%. 11 of various collaborative filtering methods were explored, such as SVD, SVDpp, SlopeOne, NMF, Normal Predictor, several KNN-based methods etc. The used algorithms for a fit on the training set with the default parameter seters. This trained algorithms were then used to assess the performance in the test set. Analyze the time of fitting of the several types of collaborative filtering algorithms and the time of testing several types of recommender systems and compare the RMSE, MAE, and MSE values. Help to advance the knowledge on the algorithm selection, its use and effects in recommendation systems. Recommendations for future research that would extent knowledge of recommender systems. They should also offer suggestions that assist researchers and practitioners in choosing the most suitable algorithm for use in their given applications and for adjusting the parameters of other existing algorithms. According to the results obtained, the Baseline Only algorithm possessed the lowest mean RMSE value, hence showing the highest rating prediction accuracy. The lowest average classification of MAE and MSE means represented by the SVDPP algorithm demonstrated its work on these criteria is accurate. Baseline Only was the computationally least intense of all the algorithms with the least mean test times out of all the algorithms tried.

AlZu'bi et al. [8] This paper also introduces a content-based movie recommendation system that can recommend movies based on metadata, where the TF-IDF and cosine similarity algorithms have been used in recommending the movies to the users, and has also assessed the performance of the various collaborative filtering techniques. To reduce the high dimensionality of the data, the following steps are performed: joining of similar tables, feature selection and removing stop words. TF-IDF and Cosine Similarity was chosen as the primary approaches for developing the recommendation system. This technique would be applied to movie metadata concept such as cast, crews, keywords and genre. The SVD algorithm provided the highest RMSE and MAE values of all the models considered in this study, which were 90% and 69% respectively. Other extensions of the KNN and NMF algorithms also fared well with KNNBasic and NMF embodying the next better performers. research gap– A study of ontology incorporation into hybrid recommendation solutions that incorporate both collaborative and content-based filtering, taking more cognizance on users' behaviour and propensity to do certain things than on simple item attributes. Expanding the space of factors taking into consideration other user's preferences and movie characteristics than included in this study for enhancing the recommendation quality

Addagarla et al [9] This paper presents a concept of a similar image-based recommender system for e-commerce on a similar principle but with different techniques for the purpose of comparison, employing dimensionality reduction with PCA-SVD and an unsupervised clustering approach based on K-Means++, and tests the resultant clusters using a variety of cluster evaluation metrics. The authors suggested a similar ML driven approach for this image-based recommender engine using a combination of PCA-SVD for feature extraction and K-Means++ for clustering. The proposed PCA-SVD transformed K-Means++ was found to enhance the performances of five other unsupervised clustering algorithms in terms of cluster performance. The authors tested on a 40,000-fashion product image dataset and produced Silhouette Coefficient of 0.1414, Choosing the best features that gave CHI score of 669.4 and DBI score of 1.8538 adopting their proposed approach., Applied PCA-SVD in the feature of the image data to reduce the dimensions of the high-dimension. K-Means++ clustering was used in order to cluster similar products Used Manhattan distance to the sample centroids to make list of the N nearest products When compared their approach with other unsupervised clustering algorithms namely Minibatch, K-Mediod, Agglomerative, Brich and Gaussian Mixture Model.

Tran et al. [10] This paper introduces a novel machine learning approach, ML. Recommend that uses both matrix factorization and time factors in e-commerce product recommendations, which is assessed based on a number of measures. The authors put forward a new machine learning model called ML. Recommend that integrates matrix factorization and time factor for product recommendation and logistic regression for customer comments. The ML. Recommend model is as follows: data preprocessing stage, model training stage, model evaluation stage, model dumping and using stage. The ML. Recommend model performance was analyzed with the use of mean absolute error (MAE), mean square error (MSE), root mean square error (RMSE), R-squared, and area under the curve (AUC) with comparison to other recommendation models. To provide an idea of this kind of information, a

matrix factorization and a time factor combination for product recommendations listed based on user ratings could be adopted. As a second step, in order to examine the customer comments on the products, the logistic regression approach is used. A real-world use case followed 9-step closed-loop recommendation modeling process comprising data preprocessing, model training, evaluation, saving/loading and product recommendation. – Research a class architecture where Data, Error, and Predict namespaces are set to meet the needs of the recommendation model - Implemented key methods in the Recommend Engine class that deals with loading string data and building and evaluating the recommendation model as well as performing the final recommendation.

Tyskyi et al. [11] The paper describes a medicinal cocktails recommendation system that the authors propose for developing the therapeutic alcoholic beverages recommendation system for users. We can search for cocktails by available Spirits, Forca Cocktails is an App that enforces healthy living among people, thus expanding the quality of their lives. The system is aimed at dietitians, nutritionists as well as any other individuals who would like to have some wines without actually consuming alcohol.There is a Telegram bot for the fast search of recipes, and the system built in the work uses the system with content-based filtering and the Euclidean distance for the definition of similarity. The approach is composed of two fundamental methods: a combination and fusion method in making prediction and a cascade method in refining ranked recommendations. Furthermore, Analysis of user-product interaction has also been done by matrix factorization. The simplicity and practicality of the system for prescribing individual targeted cocktails based on the user's health condition show that the system has great potential in the field of users' health enhancement. Khadse et al. [12] The work is devoted to the analysis of movie recommendation systems, and presents experimental results of content-based and collaborative filtering approaches using MovieLens 100k dataset. Thus, the objective is to minimize error in predicting users preferences and reduce the problem of information overload brought about by the ever rising use of internet. This research uses algorithms from the GraphLab package as well as City Block Distance, Euclidean Distance, and Pearson Correlation measures, including k-NN and Softmax regression for recommendation.The paper assesses the systems based on Precision and Recall and seeks to achieve a high result for both. The above findings suggest that our proposed Item Similarity Model achieves greater precision than the Popularity Model, in contrast, the Popularity Model has a propensity to over-fit since it utilizes ratings. The results reflect the advantages and drawbacks of each approach, which make the further analysis of recommendation engine's effectiveness more comprehensive. Abdalla et al. [13] This paper is devoted to improving item-based CF models, which is the most used in recommender systems. It deals with such issues as sparsity and the cold-start issue, which are typical for item-based architectures. In an attempt to increase the performance of the algorithms used, five new similarity measures are suggested to better cope with the inherent sparsity of the data. The experiment compares and establishes thirty similarity measures using two datasets, viz; MovieLens 100K and Film Trust, to measure their efficiency.The results indicate that the intended measures, especially the proposed NPSM, outcompete other existing solutions and on the average the measures attain the minimum error and mean squared error. Other suggested indicators such as QTIJ and NHSM also score relatively well on virtually all assessed criteria. The study also shows that the new similarity measures improve the recommendation accuracy by large, particularly in high sparsity data environments, thereby enhancing efficiency of the CF systems. Chen et al. [14] The paper aims to present a music recommendation framework developed for the group context, to develop solutions for the problem of mismatching music preferences among different customers. It utilises listening frequency as an indicator of user interest and uses latent factor models to mix features of track and group level popularity. The framework uses two algorithms: The methods include FM and HPCF, of which the HPCF model is based on Singular Value Decomposition (SVD). Evaluation measures of precision, recall rate, and nDCG is employed and from the data obtained, it is evident that the proposed methods improves accuracy, diversity and novelty of products and services with better recommendation top 50 as compared to the baseline techniques. The framework succeeds at achieving the goals of the approach, including accuracy, diversification, and novelty while avoiding popularity bias and illustrates the utility of the grouping approach with precise group based music recommendation, particularly relevant to the groups with higher similarity. Avini et al [15] The paper therefore discusses recommender systems for online commercial software and the various data mining approaches that may be employed in making the recommendations. It announces c3m – a new clustering algorithm defined by the number of clusters that can change throughout the process. To achieve the objective, this research employs Movielens ml-100k dataset and employs KNN to find similar users and items. The output of the proposed system is therefore a collection of recommended items arranged in a rank order of preferences according to the user's behavior patterns. Moreover, collaborative filtering and clustering are also examined in the paper to develop better clustering approaches and increase recommendation precision; consequently, the proposed method has a lower MAE error rate of 0.43.

III. Materials and Methods

This section summarizes the software and the hardware requirements used to implement various Machine Learning Recommendation models for accurately determining product recommendations to enhance and improve user shopping experience.

A. Software Requirements

1) Python (version 3.8+): It is a language that has been used in the project for handling data, building up models using machine learning and for processing with hadoop through

Pyspark. Overall libraries that include Pandas, NumPy and Scikit-Learn, each of which play enabling roles in maintaining good lexical structure, analyzing the data and modeling.

2) Google Collab Pro: The Google collab pro version offers the possibility to use NVIDIA Tesla T4, V1004 GPUs, and P100 to meet the deep learning requirements easily. These high performing GPUs make the computation of training and testing of models involving large datasets possible. The Google collab pro version offers the possibility to use NVIDIA Tesla T4, V1004 GPUs, and P100 to meet the deep learning requirements easily. These high performing GPUs make the computation of training and testing of models involving large datasets possible.

3) Hadoop (version 3.x.x+): Hadoop acts as a base for storage as well as distributed handling in the project. There is a standard model of data storage and computation similar to the Hadoop Distributed File System and outputs are stored across the machines that were used in computations. This makes it possible to treat the data and makes expansion very easy.

4) PySpark (version 2.x.x+): PySpark is a collection of interfaces in the Python programming language to the large-scale data processing system called Spark. It is used for handling large data in your project and we are able to manipulate the data and perform data analysis and Machine learning with Hadoop n Spark integrate distributed processing system.

5) Python Libraries: Pandas: a zero-cost and compulsory library of data science in Python, NumPy: a fundamental mathematical package also in Python, Scikit-Learn: a data science library in Python for machine learning algorithms. Pandas is majorly used for data analysis on tabular data, and it's better used for data sets with structured data while NumPy is used for numerical computations and finally, Scikit-learn is used to implement the different algorithms on your data.

6) Operating System: Windows 10/11: The project can be developed with the help of Windows 10/11 and certain Linux utilities will be used with the help of WSL2 (Windows Subsystem for Linux) which is built-in in the Win 10/11. This makes it easy in order to incorporate the Linux elements of Hadoop within a Windows built environment.

7) Java Development Kit (JDK 8+): Apache Hadoop and Spark need JDK to be run as both these frameworks are developed in Java. It includes all the requisites for assembling of Java applications and running them and for making certain that Hadoop and Spark run okay in your environment.

B. Hardware Requirements

1) Processor: Intel Core i5 or AMD Ryzen 5 Quad-Core, 2.5Ghz or above, Recommended – 3.00 Ghz or above. At the minimum, the PC needs 4 cores CPU capability to handle basic data processing operations when dealing with more distributed systems such as Hadoop/Spark.

2) RAM : Minimum: 8 GB. Thus, for the small and the middle datasets, the amount of not less than 8 GB RAM will be enough for simple types of works.Recommended: 16 GB or more. For dealing with multiple data sets as well as performing utilization processes, including PySpark jobs and Hadoop activities, 16 GB and above of RAM will notably raise the functionality and cut the time equally.

3) Storage: Minimum: 256 GB SSD. A fast data access technique is crucial for faster data read and write operations, and an SSD is the solution to this problem.Recommended: 512 GB SSD and above with extra 1 TB HDD. The above arrangement enables you store huge data sets especially when dealing with Hadoop HDFS. SSD becomes beneficial for executions of running applications and the use of HDD is recommended for archiving or storing big databases.

4) Network: High-speed internet: Needed to download datasets used in this work, integrate with cloud services, and get updates to the system. A fast internet connection will also help in communication with service such as Google Collab or Hadoop clusters which are run remotely.Ethernet or stable Wi-Fi: For multi-node Hadoop clusters, the available Ethernet connection or Stable Wi-Fi is critical to guarantee the swift and consistent messaging between the different nodes. When working in distributed environment issues like network latency and speed of a system play a very crucial role.

IV. METHODOLOGY

A. Dataset Description

The target data set centers of product review on electronics products where every record corresponds to one review written by a user with an accompanying rating of the product. This dataset is extensive and will be useful in creating and assessing recommendation models that consider users' product interactions in the setting of e-commerce.

1) Attributes: : The dataset contains the following four key attributes for each record:

User ID: A code given to the user to identify him or her uniquely. This attribute makes it easier to track users as well as their specific behavior in the data set.

Product ID: As mentioned earlier it is a number or letters or characters given to each product. Every product has an ID which aids in linking a certain product with a user's feedback.

Rating: A number from 1 to 5 that reflects the perception of the user on the product. This scale of rating offers an opportunity to quantify the level of satisfaction the users have for the product.

TimeStamp: UNIX-time style integer of the date and time when this review was submitted. It also enables temporal analysis of users' activity, their patterns, and their preference shift during different time periods.

2) Data Characteristics: It includes overall 7,824,482 records that describe the electronics category users-product interaction with the ratings given by users to products. Each row represents certain aspects of the rating behavior which forms the basis for developing recommendation systems. .

3) Unique Values: It has about 4,200,000 different user IDs based on information shown in the dataset which includes clients with various characteristics. Also, we have approximately 476 thousand unique product IDs, which reveals the scope of the offered Electronics products to develop effective recommendations for many products.

4) Rating Distribution: The ratings are between 1 and 5 with the highest percentage of the ratings in the 4 and 5 ratings interval. From this distribution it may also be inferred that users of the website are likely to rate products liked hence the perceived satisfaction with the Products in the Electronics category. This gives an skewed distribution which is useful in measuring and adjusting recommendation models.

5) Temporal Aspect: The identification on timestamp data was made whereby data spans for several years and thus, there is the possibility of doing time analysis of users. In knowledge of user preference and rating patterns across time can be used to create temporal models that give an illustration of trends, seasonal changes, and the shift in the Electronics category.

6) Size: Size of the current dataset is 318.77 MB which is also not very large to process with large scale machine learning algorithms. This file size is convenient for data processing and training since it is not heavily loaded with a large number of Mayer spaces when implementing such great algorithms as matrix factorization, and collaborative filtration to create personalized offers of products.

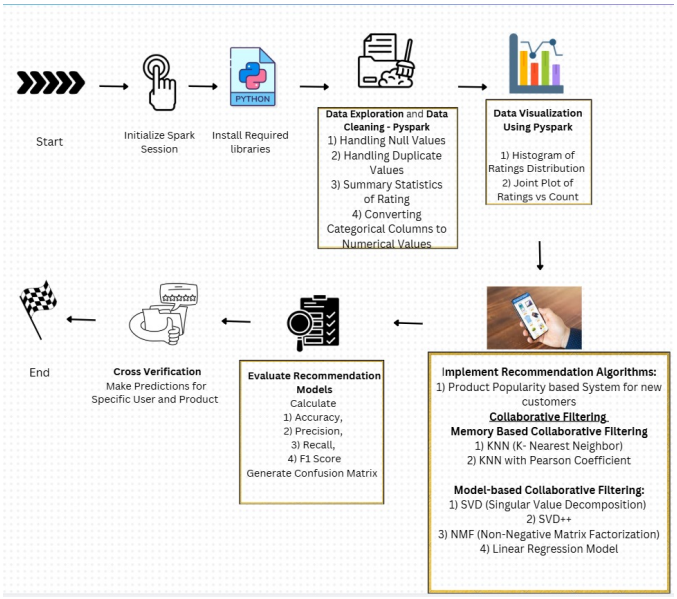

Fig. 1. Architecture of the E- Commerce Product Recommendation

B. Pre-processing Stage

The following data preprocessing steps were performed for the quality of the available data and to get the data ready for modeling . Some of these steps solved issues of data duplication and missing values, aligned all the column headers and converted categorical data into numbers that would facilitate the deployment of recommendation algorithms. During the preprocessing stage the dataset was checked for null values so as to have up to date, complete and quality data. Empty data for any of the columns (user-ID, product-ID, rating or time-stamp) can mean that there is some loss of data and this it may be detrimental to the performance of recommendation system. The rows with null values were either dropped or filled depending on features of the specific column and general relevance of the data.

1) Handling Duplicate Values: Any instance of having more than one row for similar values for a variable were deleted. Issues such that the same user-product interaction could possibly be recorded severally could cause case duplication hence a biased model. This was done to ensure that the data presented was clean with none of the repeats in order to have each line representing a single user product interaction.

2) Renaming Column Names: Therefore the column names have been changed to make them more descriptive and consistent. Some of the column names were changed to user ID, product ID, their rating and time for better understanding. This step makes the dataset more explicitly comprehensible in context and hence during analysis and model duplication, making it simpler for researchers and developers to implement.

3) Categorical Column to Numerical Transformation: There are always discrete variables like user ID, product ID, etc.; one is bound to transform them into numerical form using methods like label encoding or string indexing. This change was very crucial because machine learning models only accept numerical data for their computation. Since all the datasets used in matrix factorization and collaborative filtering require each unique user ID and product ID to be uniquely represented numerically, the new datasets met this requirement. The map of the recommendation system is provided step-by-step in the following Fig. 1: The system starts with the creation of a Spark Session which will allow distributed and optimized computations. The dataset is then subjected to data pre-processing and data cleaning process, which involves dealing with null and duplicate values, description of rating statistics and converting features into numerical form such as user-id and product-id. This makes it easy for the data to be in a format that machine learning models will appreciate. However, another process is carried out through data visualization, in order to uncover patterns of the ratings and user-product interactions using tools such as histograms and joint plots. Subsequently, the system implements two key recommendation approaches: Memory-Based Collaborative Filtering involves nearest neighbor approach and weighted voted based approach such as KNN and KNN with Pearson Coefficient to discover similar Users and Items whereas Model-Based Collaborative Filtering involves training models on the data such as Singular Value Decomposition (SVD), SVD++ ,Non-negative matrix factorization (NMF) and Linear Regression. A Product Popularity Based System is used only for users not having any past communication with the system, particularly, when a user is new. The models are assessed through various usual parameters that include accuracy, precision, recall besides the F1 score and factors obtained from a confusion matrix. Last but not least, the system performs cross validation of predictors concerning specific users and certain products to define the real efficiency and accuracy of the developed prognosis tools. In this pipeline, a systematic training method for constructing an accurate and scalable recommendation model has been outlined.

C. Machine Learning Models for Recommendation

1) Singular Value Decomposition (SVD): Matrix factorization technique is used for designing a recommendation system; the specific technique that has been applied for this purpose is Singular Value Decomposition (SVD). It decomposes the user-item interaction matrix into three components: U (user features), Σ (singular values), and V^t (item features). This decomposition captures the underlying dependencies between the users and products so that predictions can be made from even small numbers. Effective on our work, SVD was used on the users' by-products matrix whereby the inherent pattern that is within a given users preference and product characteristics was utilized. The predicted ratings in turn have been reconstructed using these components for generating the relevant recommendations. The generality of the algorithm in such scenarios and managing data sparsity were among the various benefits provided by the algorithm for improving recommendation results.

2) SVD++: Compared to the regular Singular Value Decomposition, SVD++ takes into account implicit feedback, which means a user's clicks on the products that they did not rate on a scale from one to five, for instance. To integrate explicit and implicit data into our recommendation system, we used SVD++ thathelps to account for latent details in the user's preferences. This algorithm outperformed other methods in modeling personalized users' behavior thanks to accounting for the effect of unrated items which enhanced prediction precision. Based on the experiment, the proposed SVD++ provided better results than the original SVD in terms of corresponding evaluation measurements such as precision and recall. Due to their good performance on sparse data our solution was incorporating it into our system especially since we were dealing with large amounts of sparse data.

3) NMF: NMF is a model for matrix factorization where the interaction matrix of the users and the products is made of two other non-negative matrices; this is very helpful and easy for recommendations. To that end, in our system, NMF was employed to discover underlying factors that reflect the users' and the products themselves. The non-negativity constraint gave the factorized components sensible values, for instance user favourability toward given product classes. Another advantage of NMF is that its performance was even higher on the relatively small number of arrays like in our case when most of the users rate limited number of products. The algorithm given suggested rather precise suggestions, and hence acted as a basis of comparison with the other algorithms.

4) K-Nearest Neighbors (KNN):) KNN is an original memory based approach of collaborative filtering which includes finding that set of users who are most similar to the target product or vice versa based on a defined metric of similarity like Euclidean distance. In recommendation system of our study, KNN for rating prediction searched for neighbors, who are similar to the target user in terms of preferences or behavior. Although KNN was straightforward and relatively easy to execute and implement, its scaling problems were evident just as we tried to apply it to the over 7 million Interactions data set. Still, KNN was used as reference algorithm which gave an understanding of how well memory-based algorithms could perform compared to model-based approach such as matrix factorization.

5) KNN with Pearson Correlation: Pearson correlation was found more suitable than cosine similarity measure because Pearson uses the strength of relations between users or products ratings than the simple cosine measure of KNN. In our recommendation system this method used for finding the similar rating trends for users of products and give a more accurate recommendation. While raw distance measures reflected the distances in the numerical ratings, Pearson correlation coefficients caught the relative shapes of the ratings because they compared the ratios of the differences between ratings to the standard deviations of the ratings, which provided good identification of changes in user preferences when the users provided highly diverse opinions and evaluations.

6) Linear regression: Linear Regression is an easy method of estimating the user ratings from the features including the user number and product number. For example, in our recommendation system, it represented the dependency between input features and people's ratings by a simple linear model, so that it is easy to comprehend and implement. However, this algorithm was highly linear which made it difficult to hold the inherent interactions of user preferences and product attributes. Linear regression may have performed worse than some of the other methods such as SVD++ and NMF, however, in our comparative analysis linear regression was used as a yardstick against which other more complex algorithms could be compared.

V. Results and Discussions

This section describes the various metrics used in this study including accuracy, precision, recall, and the F1 score for the implemented recommendation system and each machine learning model used. The prediction time differences of models performed using PySpark against the same models performed without PySpark are measured. Relative to this, the analysis clearly depicts that PySpark leads to increased efficiency and scalability, particularly when dealing with large systems of data and how it increases the effectiveness of Recommendation System. Table 1 and Table II describes the values of various evaluation parameters without using PySpark modules, whereas Table III and Table IV depicts the performance of models in distributed environment using PySpark.

TABLE I
Model Evaluation Metrics

Model	Accuracy	Precision	Recall	F1 Score
SVD	0.8050	0.8103	0.9914	0.8917
KNN	0.8270	0.8278	0.9988	0.9053
SVD++	0.8020	0.8016	1.0000	0.8899
NMF	0.8150	0.8159	0.9975	0.8976
KNN with Pearson	0.8320	0.8318	1.0000	0.9082
Linear Regression	0.8210	0.8210	1.0000	0.9017

TABLE II
EXECUTION TIME FOR MODELS

Model	Execution Time (seconds)
SVD	0.8786
KNN	3.5849
SVD++	1.0352
NMF	1.0915
KNN with Pearson	0.4564
Linear Regression	0.1833

TABLE III
MODEL EVALUATION METRICS WITH PYSPARK

Model	Accuracy	Precision	Recall	F1 Score
KNN	0.7684	0.8121	0.9284	0.8663
Linear Regression	0.8087	0.8087	1.0000	0.8942
NMF	0.9983	1.0000	0.3814	0.5522
SVD	0.9960	0.3227	0.4187	0.3645
KNN with Pearson	0.7684	0.8121	0.9284	0.8663
SVD++	0.9960	0.3227	0.4187	0.3645

TABLE IV
EXECUTION TIME FOR MODELS WITH PYSPARK

Model	Execution Time (seconds)
KNN	0.6343
Linear Regression	2.9548
NMF	6.4833
SVD	5.5214
KNN with Pearson	0.7861
SVD++	5.2953

Using PySpark the lessons learned from the Results section of this recommendation project, one is able to see a massive advancement from a traditional development environment in developing and testing out recommendation models. Comparing the working time of the models, it is clear that PySpark give significant improvement in terms of speed, scalability and general effectiveness. When there are normal conditions, time taken to execute the models such as NMF and SVD where 1.09 and 1.04 seconds respectively. However, these models executed in PySpark environment took significantly less time; NMF took 6.48 s and SVD took 5.52 s. This is because, as will be discussed in detail later, PySpark is built for distributed computing, which makes it possible to process data in parallel over many nodes; a process that makes computation very fast especially when dealing with very large datasets.

Scalability is often a problem in most environments and with PySpark this issue is solved because PySpark has the ability to distribute signals over a far greater extent allowing for large data sets to be managed without going through memory problems or slowing down. The normal environment models are laminar to machine resources of memory and computational power, which cannot be multiplied with data size. As for the size of data, PySpark does not have limitations because it allows data to be split into multiple clusters across different machines making it more useful in large scale recommendation systems tasks.

Despite the fact that the values for accuracy, precision and recall of PySpark models for example, Linear Regression (0.8087 accuracy); Normalized Multi-Feature (NMF) (0.9983 accuracy) are closer to the normal environment, time taken to execute PySpark models has improved. These speedups also facilitate faster model experimental and tuning phases and therefore fast execution for real world solutions. For example, the model KNN with Pearson coefficient showed the same accuracy in both environments, 0.7684, but the PySpark configurated in the strategy had a shorter computing time, 0.7861 seconds compared to the normal environment 3.58 seconds for KNN. With respect to the PySpark environment, the NMF model outperforms the other models with high percentage accuracy of 99.83% and a high percentage precision of 100% with minimum time consumption of 6.48 seconds. SVD complements closely with a little lower precision and recall, despite scoring high accuracy (99.60%) and time taken to run the algorithm (5.52 sec). KNN with Linear regression have good performances with reasonable metrics but are relatively slower compared to the existing best algorithms. Some of these algorithms such as KNN with Pearson Correlation and SVD++ rank lower primarily because of lower recall rates particularly on large data and long times taken to complete large data set tasks. Therefore, PySpark offers the possibility of processing big data faster, dividing tasks into multiple processors while delivering high accuracy results and is highly recommended for the building of recommendation systems when dealing with large data sets, since the amount of data is constantly increasing. The nature of the findings presented strongly suggests that, in terms of scalability, time to completion, and model performance, PySpark is decidedly superior to older model environments and should therefore be the recommended platform for production-grade recommendation systems.

VI. CONCLUSION

Multiple machine learning algorithm-based E-Commerce product recommendation engine ia proposed and tested where SVD, SVD++, NMF, KNN, KNN with Pearson coefficient, and Linear Regression algorithms are used. Hence in this study comparison between these algorithms and highlighting the gains made in efficiency, scalability and overall performance of these algorithms by incorporated PySpark for distributed computing. Therefore, it is concluded that large scale data set could be efficiently managed by PySpark enhancing the training and prediction time by a faster rate preserving accuracy and producing highly robust models. This research also underlines the use of distributed frameworks in reacting to the performance of recommendation systems while analyzing large datasets in real-world e-business environments.

REFERENCES

[1] Padmavathi, A., Amrutha, G., Sah, R.K., Chapagain, B. and Manasa, A.S.L., 2024, January. Performance Evaluation of Movie-based Recommendation Systems using Hybrid Machine Learning Models. In 2024

5th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI) (pp. 195-201). IEEE.

[2] Zhang, P., Zhang, Z., Tian, T. and Wang, Y., 2019. Collaborative filtering recommendation algorithm integrating time windows and rating predictions. Applied Intelligence, 49(8), pp.3146-3157.

[3] Deng, Y., 2022. Recommender systems based on graph embedding techniques: A review. IEEE Access, 10, pp.51587-51633.

[4] Chaudhari, K. and Thakkar, A., 2020. A comprehensive survey on travel recommender systems. Archives of computational methods in engineering, 27, pp.1545-1571.

[5] Paula, B., Coelho, J., Mano, D., Coutinho, C., Oliveira, J., Ribeiro, R. and Batista, F., 2022, June. Collaborative filtering for mobile application recommendation with implicit feedback. In 2022 IEEE 28th International Conference on Engineering, Technology and Innovation (ICE/ITMC) & 31st International Association For Management of Technology (IAMOT) Joint Conference (pp. 1-9). IEEE.

[6] Anwar, T. and Uma, V., 2021. Comparative study of recommender system approaches and movie recommendation using collaborative filtering. International Journal of System Assurance Engineering and Management, 12, pp.426-436.

[7] Saini, K. and Singh, A., 2024. Original Research Article Comparative analysis of collaborative filtering recommender system algorithms for e-commerce. Journal of Autonomous Intelligence, 7(2).

[8] AlZu'bi, S., Zraiqat, A. and Hendawi, S., 2022. Sustainable Development: A Semantics-aware Trends for Movies Recommendation System using Modern NLP. International Journal of Advances in Soft Computing & Its Applications, 14(3).

[9] Addagarla, S.K. and Amalanathan, A., 2020. Probabilistic unsupervised machine learning approach for a similar image recommender system for E-commerce. Symmetry, 12(11), p.1783.

[10] Tran, D.T. and Huh, J.H., 2023. New machine learning model based on the time factor for e-commerce recommendation systems. The Journal of Supercomputing, 79(6), pp.6756-6801.

[11] Tyskyi, S., Liaskovska, S. and Augousti, A.T., 2023. Design Approaches and Tools for the Implementation of a Medicinal Cocktails Recommendation System. In IDDM (pp. 292-302).

[12] Khadse, V.P., Basha, S.M., Iyengar, N. and Caytiles, R., 2018. Recommendation engine for predicting best rated movies. International Journal of Advanced Science and Technology, 110(2), pp.65-76.

[13] Abdalla, H.I., Amer, A.A., Amer, Y.A., Nguyen, L. and Al-Maqaleh, B., 2023. Boosting the item-based collaborative filtering model with novel similarity measures. International Journal of Computational Intelligence Systems, 16(1), p.123.

[14] Chen, S.H., Sou, S.I. and Hsieh, H.P., 2024. Top-n music recommendation framework for precision and novelty under diversity group size and similarity. Journal of Intelligent Information Systems, 62(1), pp.1-26.

[15] Avini, H., Mirzaei ZavardJani, Z. and AvinI, A., 2022. Improved Recommender Systems Using Data Mining. Transactions on Machine Intelligence, 5(2), pp.97-114.