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

PROJECT

Submitted by

BL.EN.U4SCE21072	J. Sri Sai Samhitha
BL.EN.U4SCE21072	K. Adarsh Sagar
BL.EN.U4SCE21124	Mohammed Jaffer Ali

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA SCHOOL OF COMPUTING

AMRITA VISHWA VIDYAPEETHAM

BENGALURU 560 035

MAY - 2024

Annexure 1

E-COMMERCE PRODUCT RECOMMENDATION ENGINE: ENHANCING CUSTOMER ONLINE SHOPPING EXPERIENCE

A PROJECT REPORT

Submitted by

BL.EN.U4CSE21072   J SRI SAI SAMHITHA

BL.EN.U4CSE21072   K ADARSH SAGAR

BL.EN.U4CSE21124   MOHAMMED JAFFER ALI

in partial fulfillment for the award of the degree

of

COMPUTER SCIENCE AND ENGINEERING

AMRITA SCHOOL OF COMPUTING,

AMRITA VISHWA VIDYAPEETHAM

BENGALURU 560 035

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING, BENGALURU

BONAFIDE CERTIFICATE

This is to certify that the Project Report (19CSE353 – Mining of Massive Datasets) entitled “**E-COMMERCE PRODUCT RECOMMENDATION ENGINE: ENHANCING CUTOMER ONLINE SHOPPING EXPERIENCE”** submitted by

J SRI SAI SAMHITHA	BL.EN.U4CSE21072
K ADARSH SAGAR	BL.EN.U4CSE21075
MOHAMMED JAFFER ALI	BL.EN.U4CSE21124

in partial fulfillment of the requirements for the award of the **Degree Bachelor of Technology** in “**COMPUTER SCIENCE AND ENGINEERING**“ is a bonafide record of the work carried out under my guidance and supervision at Amrita School of Computing, Bengaluru.

SIGNATURE

Dr. Supriya M.

Dept. of CSE, ASE, Bengaluru

This project report was evaluated by us on ………………

ACKNOWLEDGEMENTS

The satisfaction that accompanies successful completion of any task would be incomplete without mention of people who made it possible, and whose constant encouragement and guidance have been source of inspiration throughout the course of this project work.

We offer our sincere pranams at the lotus feet of **”AMMA”, MATA AMRITANANDAMAYI DEVI** who showered her blessings upon us throughout the course of this project work.

We owe our gratitude to **Prof. Manoj P.**, Director, Amrita School of Engineering, Bengaluru.

We thank **Dr. E. A. Gopalakrishnan,** Principal and Chairperson-CSE, Amrita School of Computing, Bengaluru for his constant support and inspiration.

It is a great pleasure to express our gratitude and indebtedness to our course faculty **Dr. Supriya M.** Department of Computer Science and Engineering, Amrita School of Computing, Bengaluru for her valuable guidance, encouragement, moral support, and affection throughout the project work.

We would like to thank express our gratitude to project panel members for their suggestions, encouragement, and moral support during the process of project work and all faculty members for their academic support. Finally, we are forever grateful to our parents, who have loved, supported, and encouraged us in all our endeavors.

J Sri Sai Samhitha – BL.EN.U4CSE21072

K Adarsh Sagar – BL.EN.U4CSE21075

Mohammad Jaffer Ali – BL.EN.U4CSE21124

ABSTRACT:

This project deals with the development of an E-Commerce Product Recommendation System that can guarantee increased customer satisfaction due to the provision of product recommendations from their order history. Built over a database of around seven-million rows, the system identifies and employs key attributes such as User ID, Product ID, Rating, and Timestamp to quantify and qualify user behavior. The main objective is to suggest a product with a focus on the individual user's preferences, which will increase his activity and sales. In achieving this, the project employs and assesses the feasibility of a further set of machine learning algorithms, of which are KNN, KNN with Pearson Coefficient, NMF, SVD, SVD++, and Linear Regression. These algorithms are thoroughly checked and compared for the forecasting accuracy by the following parameters, namely the Accuracy Rate, Precision, the Recall Rate, the F1 Score, and the Confusion Matrix. One of the project strengths is the use of PySpark, a distributed computing framework, to deal with computations for this scale of a dataset. In particular, by availing itself of the element ability to perform multiple operations simultaneously as well as distribute data storage, PySpark drastically improves the scalability of the recommendation system. Here we propose a comparison of the models built with and without the PySpark that shows that while a distributed computing protocol does decrease the execution time of an algorithm, it also does not worsen, though may even enhance, the predictive ability of the algorithms if resources are scarce, such as in the case of SVD and NMF. This full comparison emphasizes on two opposing aspects of an algorithm: accuracy and computational complexity, and demonstrates advantages of distributed systems when it comes to handling big data. In conclusion, one obtains a powerful, flexible, and high-performing recommendation system adaptable to contemporary requirements of existing e-commerce sites.

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1.1 Background

Recommendation systems are seen as key components of most contemporary e-business environments since their primary objective is to help customers increase their satisfaction by presenting recommendations that can suit their tastes and needs most effectively. These systems solve the problem of information overload, a problem that faces most users of the internet, that is, how to choose from the overwhelming options available online. The effectiveness of recommendation systems is useful for customers in that it gives customers recommendations on various products by studying the user's behavior and their interaction history and preferences during shopping.

Recommendation engines are again classified based on the approach used as content-based filter, collaborative filter and hybrid filter systems. The type of methods refers also to the content of products according to the items with which the user has previously interacted. The second type of filtering known as collaborative filtering goes through the users' behavior patterns to arrive at a product that similar users have found appealing. The second technique takes the best features from both systems and improves the recommendation precision.

The increase of e-commerce platforms, however, poses new problems in the engineering and implementation of recommendation systems. Large data sets containing millions of user interactions are unlikely to be easily manageable and thus call for complex and efficient algorithms. Such quantities of data can hardly be handled by traditional machine learning models which explains why distributed computing frameworks are used. This project is centered around building the ability to use such frameworks to create a recommender system that is scalable, fast, and produces the correct recommendations.

1.2 Problem Statement

The current issue that has been raised is that of creating an optimal product recommendation system for an e-Commerce platform in order to ascertain the likelihood of buyer preferences as well as recommending the right products for the buyer. Having a large number of records for 7 million ‘user-product’ interactions, it is implemented as a complex task to help develop a system that should not only consider the characteristics of sparse matrices but also cope with the dynamics and changes in ‘users’ behavior over time. The idea is to use the KNN, KNN with Pearson, SVD, SVD ++, NMF and Linear regression in order to get insights from the dataset and suggest relevant products. Moreover, the project will conduct a comparison of the accuracy of these models with and without PySpark computation to unravel how the scalability and parallelization features of PySpark, compared to the sequential computation, enhance the possibility of the recommendation system in terms of accuracy, precision, recall, and F1 score. The eventual goal is to improve the satisfaction of users concerning product recommendations hence making customers to engage more with the platform and ultimately cause sales.

1.3 Objective

A) The major aim of this project is to create a growing efficient e-commerce recommendation system. The idea of the system is to employ machine learning algorithms that will help to provide relevant products that would increase the interest and satisfaction of users. The project also emphasizes distributed systems, showing that libraries such as PySpark allow coping with great sets of data and are still effective.

However, more specific tasks include data preprocessing for a dataset comprising 7 million records, as well as the comparison of multiple collaborative filtering algorithms. Categorized into memory-based algorithms such as KNN, KNN with Pearson Co efficient and model-based algorithm like SVD, SVD ++, NMF and Linear Regression. The efficiency of the developed system is measured by indicators that are generally well-known, including Accuracy, Precision, Recall, and F1 Score.

This work is most directly applicable to the e-commerce platforms where the processing of large amounts of data is required in real-time mode. Data of customer's purchasing habits are not only for the purpose of selling more and making more money but also for coupling the customers so that they can enjoy their shopping experiences. The other goal that enhances scalability is the implementation of distributed computing, and since the platforms discussed here handle millions of users on a daily basis.

1.4 Dataset Description

The dataset used in this project consists of 7 million rows, capturing user-product interactions with the following key features: User ID, Product ID, Rating and Date. The User ID and Product ID help in matching the interactions between the respective IDs of users and products as in Tables 1 and 2. The Rating feature is a figure between one and five, reflecting the user's attitude to a certain product. This enables the system also to determine the level of satisfaction that is likely to be attached to these items. The Timestamp shows at what date and time the user was active and thus can give insights on temporal analysis of the usage of the product which may include for example analysis of the festive seasons or promotion crazes. This dataset is common with typical usage data in e-business where customers rate and/or review products which they have bought or used, and they will form the framework for developing a recommendation model.

This dataset holds some common inherent features of large-scale user-product interaction data. It is extremely sparse, with a lot of potential interaction missed or unreported, so only a fraction of all possible user-item pairs has feedback. This skews are typical for recommendation systems and creates a problem for algorithm design as amount of data to classify users and their relations decline. Besides that, it should also be noted that the distribution of ratings is rather binomial, and there could be more of one or two ratings, which will eventually dominate the score. This creates a problem of skewness and thus the use of data normalization or transformation techniques to prevent the models from favoring the prominent rates and favor a more balanced doing of rates. The feature columns are as included below:

User ID: The User ID is an identification number of each user across the given dataset. It is used so as to identify one user from the other and it could be a combination of numbers and letters. This feature refers to the code that are involved with the process of rating a certain product. While discussing the MapReduce task, the User ID is used to track and distinguish the interactions, while the feature of aggregation of ratings is performed based on Product ID.

Product ID: Product ID is an identification number of the product that is created within the system and assigned to each listed product. It makes one product different from others in the dataset In essence, features define each product in the dataset. Every Product ID has a rating provided by the user and Product ID is used as ID for grouping ratings in MapReduce shuffle and sort phase. It sums up the ratings to arrive at the mean rating for each product identified by a Product ID.

Rating : The Rating column is the user's opinion or his/her general assessment or rating of a particular product. The values in this column are numerical, most often presenting a range between 1 and 5 where the exit—that is, the lowest expression of satisfaction with the program—is categorized as 1, and the entry—that is, the highest expression of satisfaction with the program—is categorized as 5. This is the key for the MapReduce task in this case because the ultimate objective is to determine an average rating for products by collating all the end-user feedback. The

Rating helps compare how popular products are and determines the level of satisfaction users have with the products.

Timestamp : The Timestamp column contains the days and time the user used to rate certain product and hence, contain temporal aspect of the rating. It appears to be useful for analyzing trends over time, including when the system is most active or preferred times within the day, week, month, year or after promotional campaigns. While it is not used in the computation of average ratings in this particular task, it got be useful for additional sophisticated computations in the future work in this context.

In order to analyze and use this dataset for machine learning, it was needing some preprocessing steps to be done. For the cases where values in one of the fields are missing, the approach used was that if the missing values were few and the type was unknown, then the value was imputed with either the mean or median value depending on which was more appropriate for the distribution of the values in the field or the row was deleted if the amount of missing data was significant. This makes it possible to obtain completeness of the database and the possibility of using it to develop stable models. To avoid cases where similar topics are repeated resulting in bias towards the overall findings, duplicates were removed. Since the two features, User ID and Product ID are categorical, they were transformed to numerical form by use of indexing. This transformation helps to make these categorical variables more amenable for input into these machine learning algorithm type of structures.

1.5 Research Methodology

1.5.1 Preprocessing and Data Cleaning

a) Handling Missing Data

In real scenarios data can be incomplete with missing values and this is not favorable in model development. I/First as a preliminary step in data analysis, there is always a consideration on the degree and spread of missing data in a given data set. If the missing values are few, we impute the missing data, where used numerical data imputation by mean, median or mode, while in categorical imputation by mode of frequency. In case the missing values are many or middle of the spreadsheets, then we will choose to remove the rows or columns they belong to avoid skew. Thus, we reduce the impact of missing data and effectively assure the model of correct further work when dealing with such data.

b) Removing Duplicates

This is due to the fact that duplicate entries make the process redundant and the model into making biased predictions. For instance, if a user chooses the same product several times, it may create complications to the recommendation system. In this project, the entries which have similar User ID values and Product ID values are considered and eliminated. This makes it less likely to get instances of a user with multiple product interactions counted as separate interactions, and all interactions aggregated in the correct way. This is important in avoiding over representation of the data because some interactions may have multiple records in the data set.

Categorical variables are often the object of encoding: Dividing the data into categories Explaining the cause of certain attributes Providing relationships between or attributes of, various parameters.

In machine learning models, algorithm needs numeric data but most of the real-world big data sets contain categorical data such text data, IDs etc. The data in these variables must also often be transformed into a form that is suitable for use in machine learning. In this project, we use indexing to turn categorical variables such as User ID and Product ID into numerical features. This is done with the help of integer values attributed to categories in these columns. Such encoding makes it easier for the model to handle these variables because ME models require numerical inputs for effective computation and accuracy.

c) Encoding Categorical Variables

Categorical variables are the ones describing defined types or classes of data and do not have a numeric value, which is the reason for encoding which transforms them into numerical format. In supervised machine learning, the models need quantitative information while in most datasets, there are qualitative factors such as unique customer and product identifiers. To be useful to machine learning algorithms, these variables have to be transformed into a numeric data type. Similar to the previous project, in this work, we employ indexing to transform our category data such as the User ID and Product ID into an index. This is done in the following manner, where each category in these columns has been given a unique integer. In fact, such encoding enables the model to make sense of these variables in a manner that improves the model because most machine learning algorithms work with numbers exclusively for computation reasons.

1.5.2 Machine Learning Algorithms

It has been disposed that machine learning models are essential in developing accurate recommendation systems that will involve analyzing patterns of users' item interactions. Six algorithms were applied in this project, which applied different methodologies to improve accuracy and reliability of the prediction models. These models cover memory-based and model-based methods in collaborative filtering to give the users' relevant suggestions.

a) KNN (K-Nearest Neighbors)

KNN works out that other users/items like a target user/item by estimating distance metrics such as cosine similarity. Suggestions are given based on the idea of nearest neighbors' preferences in order to make the best use of collaborative filtering.

b) KNN with Pearson Correlation

This version of KNN uses Pearson correlation as a measure of proximity and ranking users/items to account for this bias and get improved results for similarity-based recommendations.

c) SVD (Singular Value Decomposition)

SVD gets the user-item interaction matrix and then breaks the elements of the matrix into latent factors which in turn are used to predict the missing values given by the users and items. It is efficient when the number of data observations is small in contrast to the number of features.

d) SVD++

An improvement over SVD, SVD++ also incorporates implicit feedback (for instance, clicks or views) into its algorithm giving a better prediction of the ratings with a few stated ratings.

e) NMF (Non-Negative Matrix Factorization)

NMF utilizes matrix factorization with constraints that the elements of the two matrices are must be non-negative and the latent features are therefore interpretable.

It is done with the purpose of making primary user and item profiles for purposeful recommendation.

f) Linear Regression

Linear regression predicts the number of ‘stars’ that users assigned to items based on the features that best describe the users and items. The basic one though effective is used as a reference for assessment of the recommendation performance.

1.5.3 Evaluation Metrics

The evaluation metrics have a very important function to determine the effectiveness of various recommendation models carried out with and without PySpark. They assure the best prediction of a user and improvement of the recommendation quality of the system.

a) Accuracy: Stands for the percentage correct shows the average accuracy of the model. For example, higher accuracy means that the system fits well with the users’ feedback.
b) Precision: Concerns itself with the relevancy of every recommended item among all recommended. In our project, precision helps the user to discover products they might be interested in and contributes to the project relevance.
c) Recall: Used to assess the extent with which the system is capable of identifying all form of recommendations that may be beneficial to it. Higher recall always portends an understanding of the preferences of the users since even unnecessary items are captured.
d) F1 Score: Translates the cases of true positives and true negatives into a same scale without regard to the basic calculations of precision and recall. This is especially helpful in our project to decide where to measure models for which both precision and recall matter.

e) Execution Time: Stresses models’ computational complexity, particularly, in cases when PySpark is used as a distributed computing environment. Less time that is taken when using PySpark show that it is designed for the large volume of data.

These metrics enable us to evaluate KNN, SVD, and NMF algorithms in standard and distributed environments and determine the most appropriate algorithm for practical implementation.

CHAPTER 2

LITERATURE SURVEY

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 benchmark 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 analysed 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 modelling 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.

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 behaviour 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.

CHAPTER 3

HADOOP TASK

3.1 Introduction

Hadoop has become critical to improve customer experience by offering a flexible and effective environment for processing a hundred million pieces of data, including customer reviews and rating. Thus, due to the ability to separate the programming model into a functional MapReduce approach, Hadoop is perfect for large distributed computations across numerous devices. In this case, it is needed to measure the mean value or average of the given product rating according to the users' opinions. The Mapper phase is tasked with parsing the dataset and reform the data to include instances where it is key-value data; in which the key is the product ID and the value as the rating. These pairs are then shuffled and sorted in order to be ready for the Reducer stage when the average of the ratings given to particular product is calculated. This allows the business to gain understanding about their customer's preferences and hence reveal some of the popular products in the market and issues concerning low rated products. With the help of Hadoop, the procedure of an extensive analysis of big data can be carried out in parallel, which means that the companies will be able to develop relevant strategies for applications, modify products that do not meet the customer's needs, make other valuable decisions, therefore raising the customer satisfaction level.

3.2 Dataset Description

The dataset used for this Hadoop-based analysis comprises 7 million rows, detailing user-product interactions through four key features: UserID, ProductID, Rating, TimeStamp. The User ID and Product ID are categorical names that connect the users and products together and important for rating aggregation during the

MapReduce process. The Rating feature is another that a user fills from 1 to 5, and it is essential for calculating the average number of ratings per product. Despite the fact that the Timestamp field is not used in the solution of this task, it can contain temporal data that can be useful in further analysis, such as seasonal or promotional characteristic of users' activity. The features included are as described below:

User ID: The User ID is an identification number of each user across the given dataset. It is used so as to identify one user from the other and it could be a combination of numbers and letters. This feature refers to the code that are involved with the process of rating a certain product. While discussing the MapReduce task, the User ID is used to track and distinguish the interactions, while the feature of aggregation of ratings is performed based on Product ID.

Product ID: Product ID is an identification number of the product that is created within the system and assigned to each listed product. It makes one product different from others in the dataset In essence, features define each product in the dataset. Every Product ID has a rating provided by the user and Product ID is used as ID for grouping ratings in MapReduce shuffle and sort phase. It sums up the ratings to arrive at the mean rating for each product identified by a Product ID.

Rating : The Rating column is the user's opinion or his/her general assessment or rating of a particular product. The values in this column are numerical, most often presenting a range between 1 and 5 where the exit—that is, the lowest expression of satisfaction with the program—is categorized as 1, and the entry—that is, the highest expression of satisfaction with the program—is categorized as 5. This is the key for the MapReduce task in this case because the ultimate objective is to determine an average rating for products by collating all the end-user feedback. The Rating helps compare how popular products are and determines the level of satisfaction users have with the products.

Timestamp : The Timestamp column contains the days and time the user used to rate certain product and hence, contain temporal aspect of the rating. It appears to be useful for analysing trends over time, including when the system is most active or preferred times within the day, week, month, year or after promotional campaigns. While it is not used in the computation of average ratings in this particular task, it got be useful for additional sophisticated computations in the future work in this context.

Being characteristic of large-scale e-commerce data sets, these features imply several inherent difficulties associated with account for sparsity, rating distribution skewness and missing values to name a few, which may call for preprocessing prior to incorporation in to the Hadoop infrastructure. Preprocessing steps helps in eliminating what needs to be eliminated in the dataset and making the data ready for distributed processing. The presence of missing values was handled through imputation and or deletion depending on the situation as the presence of duplicate entries were also handled. The nominal features, User ID and Product ID, were then converted to numerical for usage in potential subsequent machine learning tasks. In the MapReduce process of Hadoop, the mapper step retrieves all distinct (Product ID, Rating) combinations, and all the shuffle and sort process allows all ratings for each product to be grouped, so that average ratings can easily be calculated in the reducer step. This makes good usage of Hadoop's scalability to handle sparsity and data skewness typical in number of interactions and computes the results accurately and in parallel over millions of interactions.

3.3 Phases of Hadoop Mapper Reducer Task

3.3.1 Mapper Phase

This phase involves coming up with an output mapper which works through the raw dataset line by line extracting some pertinent data which could be useful for aggregation in the next phases. Each line of input corresponds to a record of the datasets and the data fields include userid, productid, rating, time stamp. The mapper splits each line and takes the productid and rating. These two values are emitted in the form of key and value where the 'key' value is productid and the 'value' being a rating. For instance, given an input line of user1,product1,5,1234567890 the mapper will output (product1, 5). This process is done separately for portions of the dataset by different mappers present in different nodes, to make the process parallel. Therefore, it creates such additional key-value pairs as a mapper product1, count 5 and product2, count 4 to process all the products of the dataset.

3.3.2 Shuffle and Sort Phase

This phase of Shuffle and Sorting comes up to take intermediate Key- Value pairs produced by mappers and then forms groups by key (productid). The values for each related key are grouped into a list for productid. For example, if two mappers emitted fields product1 value 5 and product1 value 3, the phase of shuffle and sort will combine them into (product1, [5, 3]). This phase is crucial helping on making sure each reducer collects all ratings for a certain product irrespective of the mapper that handled them. The keys are also arranged during this phase as well for easy processing by reducers when it comes to keys. This categorization and filtering take place at all nodes in the cluster with a view of distributing the load of reducers evenly.

3.3.3 Reducer Phase

Reducer phase briefs involves the working on the grouped key-value pairs that results from the shuffle and sort phase. The reducer gets a list of ratings as the value for each productid which is the key. The reducer determines the average rating on the product which will be obtained by summing on the rating list and dividing it by the number on ratings list. For example, if the input to the reducer is (product1, [5, 3, 4]), the reducer calculates:

Sum = 5 + 3 + 4 = 12

Count = 3

Average = 12 / 3 = 4.0

It also comes with an additional pair of key-value; the key is the productid while the value is the obtained average rating. For example, it can deliver (product1, 4.0). The same is done for all products leading to the output dataset with product ID and their average ratings.

3.3.4 Output Phase

At the end of the reducers the output is transferred to a distributed file system like the Hadoop distributed file system (HDFS). The output format is a dictionary where each productid is followed by the average rating for that productid. For example:

```
product1, 4.0
product2, 4.5
product3, 4.2
```

It can then be aggregated and sought for further analysis or represented in the form of a visualization.

The MapReduce model manages the dataset with 7 million records effectively because the model is based on a distributed approach. Mappers operate with certain pieces of the data at once maintaining parallelism, and reducers combine results separately for each productid. The shuffle and sort phase entails an effective transfer of all mapper output to reducers in a way that brings all ratings for a single product at the reducer side. The exact division of work across nodes also makes the process highly scalable and fit for purpose for large datasets.

3.4 Result

The outcome of MapReduce activity is directed towards aggregation data set with average rating for each product in the input data set comprising millions of records. Under each product ID (productid), the customers' overall average rating is calculated to provide useful step-by-step summaries. For instance, if there were several ratings for the product such as 5, 3 and 4 in the input data, the value for the product such as product1 would be 4. The last outcome is saved in a distributed storage space in order to be retrieved later for additional computations or for general use. The analysis produced in this study has its usefulness in helping organisations understand the trends in the success of products, customer satisfaction, and opportunities for improvement that contribute to more effective decision-making.

3.5 Conclusion

The-launched MapReduce framework is natural for housing, processing, and analyzing large-scale datasets as shown by the computation of averages of product ratings generated from a dataset of millions of records. Mapper, shuffle and sort, and reducer forms the three major steps that are required to complete the given computation in a structured as well as parallelised approach. One of the attractive features of this method is that it operates record by record In the mapper phase a record gets transformed to key-value pairs where each record contains one productid and one rating. This phase happens in parallel in many of these nodes and thus saves a lot of time when handling big data. The shuffle and sort phase which is done automatically is important one because it gathers all the ratings related to each product ID and aligns them to be processed in the next phase. This ensures that all data that might be relevant to a certain product is collected in one location so that data collection can be simplified. The reducer phase then takes these grouped data points and computes average rating for a specific product by cumulatively adding all the ratings and then dividing the total by count of such ratings. This helps computation to be accurate and the final output is in terms of productid and average rating. This distribution of the framework can mean that reducers can run without needed to interact with one another, improve the processing speed and capability. After each reducer finishes its work, the output is stored in a distributed storage which will help with data accessibility and reliability for future processing. The MapReduce approach not only deals with issues of parallel computation, data reorganization, and data merging but also reduces the problem of bottleneck by its orderly stages. Due to the load that is shared equally across the multiple nodes, and because of the optimized flow of data, the framework is able to handle millions of records. As such, it is a very important instrument for complex data processing, as it offers both qualitative and quantitative results.

CHAPTER 4

SPARK TASK EXECUTION

4.1 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.

4.2 Methodology

4.2.1 **Data Preprocessing**

In our project the key to a good and efficient data preprocessing is preparing the dataset for implementing the machine learning algorithms. One of the major decision-making steps is the treatment of missing data, which can significantly affect a prediction or negatively impact model performance. Handling of the missing values was done whereby one had to analyze it based on the ratings and time in form of a timestamp. Over half of the missing values were handled using statistical portrayals like mode, median or mean and if certain rows or columns contained high missing values, they were deleted. This makes sure that we work with all the information needed in constructing models to be accurate. The data preprocessing techniques implemented are as below:

A) Handling Missing Data: In any raw data set, missing data pose a huge problem in that they impact on the quality of information and general accuracy of machine learning algorithms. For this project, we noticed that some of the features we would select are incomplete and have missing values, such as ratings and timestamp features. If the missing data was few, we did some imputations like replacing missing numerical values with mean or median of the column. In case of categorical data, the mode value has been used. Where the missingness ratios were high such that using imputation may give us a skewed result we could decide to drop the complete rows or columns. This made sure that the dataset that was used was good for training good models for the credit scoring models.

B) Removing Duplicate Entries: Often records matching similar user-product interactions in the dataset may artificially inflate specific pairs and thus negatively affect the model. For instance, it appears that if a single user continuously rates multiple products with the same User ID and Product ID, then the recommendations can be influenced. To handle this issue, we did the duplicates check and removed all the rows which are duplicate from the dataset. What I did to achieve this was that I ensured that there were no repeat of a particular user-product combination which helped me to minimize redundancy in the data and therefore giving out a fairer results from the models.

C) Encoding Categorical variables; The problems related with machine learning are that it takes inputs in numerical format, so features like User IDs and Product IDs are numerical which should be transformed into categorical one. For this project, indexing was employed to provide numerical tags to each of the User ID and Product ID. Besides, this conversion was making these identifiers compatible with the machine learning algorithms while at the same time preserving the relationships and the uniqueness of these features in order to correctly translate them into the model.

4.2.2 **Spark Modules**

PySpark modules are used for manipulating large datasets in optimized manner. Pyspark.sql module is used for data manipulation, and transformation to create DataFrames for pre-processing. In data preprocessing, we use pyspark.ml.feature which for instance transforms categorical data into numerical form. For machine learning tasks, there are modules pyspark.ml.recommendation.ALS to construct collaborative filtering models for recommendations. Last but not the least, PySpark.ML uses pyspark.ml.Pipeline which provides a smooth pathway to data preprocessing, feature engineering and model building.

A) pyspark.sql: This module is crucial when working with the data and performing some calculation and transformation on it. This enables the manipulation of greater structures of organized data by making use of data frames, filter, group as well as aggregate large data set. For data operations such as reading, querying and manipulating big data we use pyspark.sql by making the data ready for machine learning and analysis.

B) pyspark.ml.feature: This particular module is used the feature engineering step which is an important stage in getting ready data for machine learning. We use string index functions such as StringIndexer and OneHotEncoder to transform qualitative features into quantitative ones. Also, other feature transformers like

VectorAssembler aid in conversion of several feature columns into one vector so it’s acceptable by machine learning algorithms in PySpark.

C) pyspark.ml.recommendation.ALS: For collaborative filtering technique the Alternating Least Squares (ALS) algorithm from this module has been used. ALS is a matrix factorization procedure normally applied to recommender systems. We use it to make recommendations regarding items and users, to estimate what items a user might prefer based on the patterns that we are able to detect.

D) pyspark.ml.Pipeline: The Pipeline module is designed to help simplify the machine learning process. It let us join sequentially one or another stage of data preprocessing, feature engineering and model training phases. This makes the workflow to be optimized for reusability to make such process in the transformation of data, the training of models and the generation of the actual predictions to be repeatable, scalable and efficient.

E) pyspark.sql.functions: This module offers multiple functions for alteration of DataFrame columns that is helpful when selecting, modifying or modifying DataFrames at scale. We do this in this module to increase our data processing capacity, and to confirm that the data is in the right format for analysis or modeling.

F) StringIndexer: Converting the categorical data from a qualitative form is important to most machine learning models, and that is the main purpose of this module. In this case, it is applied on the userId and productId that is userIndex and productIndex numerical columns respectively. Collaborative filtering algorithms that are used in machine learning models require read numerical data hence StringIndexer helps ensure that features with string data like user and product IDs can be particularly applied to the model for training. It gives an index to every category thus makes the process an essential part in natural language processing.

G) Pipeline: The Pipeline module is an enhanced subtype of the chain of operations applied on the data; each step is logically enclosed and easily adjustable. This remains helpful in different aspects of a project because it encompasses several processes of data preprocessing and model development in a unified framework. In the code, Pipeline is used to transform the sequence of the operation of StringIndexer steps of both – userId and product Id. Pipeline makes sure that all these transformations are done at a single place so that it is easy to monitor the reproducibility and higher efficiency when working with big data in distributed environment such as Spark.

4.2.3 Machine Learning Models

A) 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),   (singular values), and (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.

B) 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++ that helps 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.

C) 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 favorability 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.

D) 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.

E) 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.

F) 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.

4.3 Results

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: Model Evaluation Metrics : Without PySpark

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	1.000	0.8899
NMF	0.8150	0.8159	0.9975	0.8976
KNN with Pearson Coefficient	0.8320	0.8318	1.000	0.9082
Linear Regression	0.8210	0.8210	1.000	0.9017

Table 2: Model Execution Time : Without PySpark

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

Table 1 and Table 2 depict the values of evaluation metrics the models of the Machine Learning Recommendation models (Without PySpark) and provide us valuable insights:

a) SVD: The last used model of SVD has an accuracy of 0.8050 and its value of recall is 0.9914 which demonstrates the capability of this model impressively but this model is slightly lower in terms of preciseness and F1 score. Sample applications that have made use of the model include recommending ratings which are likely to be liked by users, however matrix decomposition makes the model slightly slower especially on large data sets

b) KNN: It also has high accuracy with a score of 0.8270, a high and satisfactory recall of 0.9988, thus makes KNN a favourable algorithm for situations where high recall is important and accurate discovery of the positive instances is important. It also proves very high accuracy 0.8278 and recall 0.9053. However, the execution time consuming by KNN is quite high (3.5849 seconds), thus, KNN cannot be suitable for large-scale data unless it is being optimized.

c) SVD++: Slightly superior to the basic SVD, SVD++'s recall, though being 1.0000, is inferior in accuracy to the basic SVD (0.8020). Although it has perfect recall, it uses implicit feedback, which was maybe an informing factor; it also brings more unfamiliar elements that can slow down calculation time compared to KNN with Pearson yet simple models.

d) NMF: NMF demonstrates reasonable levels of accuracy that are equal to 0.8150, precision being 0.8159, and recall of 0.9975. According to the degrees of rating it seems to be good when applied in situations where lack sufficient data to train on. It is a tiny bit longer by comparison to SVD with the execution time standing at 1.0915 seconds which is, however, quite acceptable for most users.

e) KNN with Pearson: Compared to other classification techniques, KNN with Pearson having high accuracy score of 0.8320 and F1 score of 0.9082. It also attains a recall of 1.0000 confirming the case in question ability to locate the instances in question. This model has been made superior by using the Pearson correlation in the calculation of similarity of users and a significantly lesser execution time of 0.4564 seconds than standard KNN.

f) Linear Regression: Similarly, Linear Regression passes our tests – it achieves exactly 0.8210 accuracy, 1.0000 recall, and 0.9017 f1-score. Despite the simplicity of the model, and the relatively short time required to execute predictions (0.1833 sec), it is predisposed to miss complex non-linear dependencies between the features. It is a concrete option when time and computational overheads must be minimum.

In general, KNN with Pearson appears to be the most accurate, with high recall and relatively fast execution time, so it is the most stable model for the set. But it can and should be much faster: for example, models such as Linear Regression and SVD offer a favourable balance of time and accuracy. Others like SVD++, NMF add other factors or constraints than SVD but they have slightly higher cost in terms of time execution. There is a decision that depends on requirements for accuracy, speed, and considerate computational complexity in the forecast model.

Table 3 : Model Evaluation Metrics : Using PySpark

Model	Accuracy	Precision	Recall	F1-Score
SVD	0.7684	0.8121	0.9284	0.8663
KNN	0.8087	0.8087	1.000	0.8942
SVD++	0.9983	1.000	0.3814	0.5522
NMF	0.9960	0.3227	0.4187	0.3645
KNN with Pearson Coefficient	0.7684	0.8121	0.9284	0.8663
Linear Regression	0.9960	0.3227	0.4187	0.3645

Table 4 : Model Execution Time : Using PySpark

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

The Table 3 and Table 4 depict the performance indicators and the execution time of the models when coded with PySpark, as well as their improvement and disparities compared to their counterparts not coded using PySpark. Model Evaluation Metrics with PySpark:

1. **Accuracy:** Linear Regression was slightly higher as seen when compared to the non-PySpark result which was 0.8210 while for PySpark was 0.8087. In this case, the reliance on the PySpark-based implementation might have allowed benefiting

from parallelization and distribution but still exhibit small sensitivity to variations in distributions.

KNN and KNN with Pearson maintained the accuracy of 0.7684, thus upon using different frameworks, the performance of these models is unchanged. SVD and SVD++ also slightly decreased their accuracy with PySpark (The former now was 0.9960). It might be because of the complexity of the given models, or because the data was sparse, and that caused impairing of the performance.

When compared with original NMF, even though it receives near zero sparsity (1.0000), its accuracy (0.9983) is rather low; it could be correlated with the decomposition method utilized in PySpark, which may make the model less sensitive to each user-item rating.

2. **Precision:**

With PySpark, NMF received a perfect precision score, equal to 1.0000, which indicates that it accurately classified all positive samples, although, simultaneously, the recall was not that high to provide both indicators a high F1 number.

There was not much of a difference between KNN and KNN with Pearson at 0.8121, hence the PySpark models were accurate in identifying relevant items.

Linear Regression also obtained the accuracy of 0.8087 showing its efficiency to locate more significant instances in distributed manner way.

SVD and SVD++ achieved lower precision values 0.3227 partly due to PySpark environment impact on them. It could be attributed to a situation where the factorization process requires more time when operating in a distributed system than when the two processes of factorization and model building occur together, thus there is a less competent model for distinguishing between significant and insignificant predictions.

3. **Recall:**

   Linear Regression was superb with 1.0000 Recall which means the identification of true positive prediction did not falter from its non-PySpark version.

   KNN and KNN with Pearson as well kept high recall values even at 0.9284 which reflects high recall characteristic in capturing items to recommend.

   SVD and SVD++ had the highest decrease of recall (0.4187) with PySpark. This paints an even bleaker picture about their ability to identify and rank relevant items because, even if they made the predictions, they only did so halfway.

   In regards to recall at 0.3814 NMF yielded a low recollect rate, which was expected due to the inherent limitations of the algorithm used in adapting and recognizing all the necessary items in the given data set while still maintaining high precision rates.

4. **F1 Score:**

   KNN and KNN Pearson gave an F1 of 0.8663 indicating balanced Precision and Recall with PySpark according to their earlier F1 result without PySpark.

   Linear Regression, although giving lower accuracy in PySpark scored slightly higher F1 score of 0.8942; this depicts the balance between precision and recall and ability to rotate accurately.

   NMF, while yielding high precision, performed worst in F1 score (0.5522) due to very low recall ratio that indicates imbalance.

   SVD and SVD++ had even lower F1 rating (0.3645) which means the models made plenty of predictions but they were not efficient when it came to precision-recall metrics, especially at a large scale.

Execution Time for Models with PySpark:

When using Pearson, KNN had the least execution time relative to other techniques at 0.7861 compared to the 0.4564 found when implementing KNN without PySpark. This means that, despite the fact that PySpark is built to be considerably more efficient in parallelizing tasks, it is not necessarily optimized for simpler models such as KNN.

PySpark consumed more time Linear Regression and took 2.9548 seconds whereas original program consumed 0.1833“. The additional time was most probably consumed on overhead related to parallelism and distributed computations within the larger framework.

SVD and SVD++ took more time in execution compared to its non- PySpark times, respectively 5.5214 and 5.2953 seconds. This is further suggestive of the time-consuming nature of matrix factorization and the extra time it takes in distributed environments across multiple nodes.

NMF is computationally expensive; therefore, it is apparent that the NMF had the highest execution time of 6.4833 seconds while in the distributed execution environment.

The proposed PySpark improvements have resulted in higher model performance in execution time for simple models like KNN and KNN with Pearson since the library works efficiently with large datasets. For the less intensive models such as K-Means and PCA, the execution time has significantly reduced as expected because they can be run in parallel, while for the more complex models like Linear Regression, SVD and SVD++, the time has slightly advanced, probably due to the overhead of distributed processing.

4.4 Conclusion

Multiple machine learning algorithm-based E-Commerce product recommendation engine is 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.