Indonesian Journal of Electrical Engineering and Computer Science
Vol. 24, No. 3, December 2021, pp. 1628~1637
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v24.i3.pp1628-1637

An enhanced framework for solving cold start problem in movie recommendation systems

Salma Adel Elzeheiry[1], N. E. Mekky[2], A. Atwan[3], Noha A. Hikal[4]
[1,2,4]Faculty of Computers and Information Sciences, Mansoura University, Mansoura, Egypt
[3]Faculty of Computer Sciences and Information Technology, Northern Border University, Arar, Saudi Arabia

Article Info

Article history:

Received Apr 3, 2021
Revised Sep 26, 2021
Accepted Oct 6, 2021

Keywords:

Big data
Cold start
Cosine similarity
Logistic regression
Recommendation system

ABSTRACT

Recommendation systems (RSs) are used to obtain advice regarding decision-making. RSs have the shortcoming that a system cannot draw inferences for users or items regarding which it has not yet gathered sufficient information. This issue is known as the cold start issue. Aiming to alleviate the user's cold start issue, the proposed recommendation algorithm combined tag data and logistic regression classification to predict the probability of the movies for a new user. First using alternating least square to extract product feature, and then diminish the feature vector by combining principal component analysis with logistic regression to predict the probability of genres of the movies. Finally, combining the most relevant tags based on similarity score with probability and find top N movies with high scores to the user. The proposed model is assessed using the root mean square error (RMSE), the mean absolute error (MAE), recall@N and precision@N and it is applied to 1M, 10M and 20M MovieLens datasets, resulting in an accuracy of 0.8806, 0.8791 and 0.8739.

Corresponding Author:

Salma Adel Elzeheiry
Department of Information Technology, Faculty of Computers and Information Sciences
Mansoura University, Egypt
Email: salma.a.elzeheiry@gmail.com

1. INTRODUCTION

The expanding volume of data currently available has created issues that prevent improvement in the performance of recommendation systems (RSs). RSs have Played a major role in accelerating tasks by offering efficient and relevant recommendations [1]. They are used to assist clients in following their preferences amidst enormous amounts of information. To offer a suggestion, an RS collects data from explicit ratings received from users and implicit feedback derived from such activities as browsing and click-through histories [2]. New challenges, such as cold start, sparsity and scalability, have arisen in response to predictive success. Most of the filtering techniques utilized within the recommendation systems uses the information of the client to supply the suitable data, but there is a issue of cold start [3]. Typically, there are two forms of cold start issue: i) When the system is confronted with a new user regarding whom nothing is known (user cold start); and ii) When the system is confronted with a new product regarding which nothing is known (item cold start). User cold start occurs when a user logs in to the system a few times without recording any history, with the result that the system does not create a user profile for coordinating significant items [4]. Item cold start occurs when items are new to the system, there is no evaluation available and there is no information exchange among items [5].

RSs can be investigated in several manners. Here, these methodologies are categorized into collaborative filtering (CF), content-based and hybrid. Collaborative Filtering: is a strategy for recommending items to new users based on their preferences [6]. There are two types of methods commonly used in CF, memory- and model-based [7]. CF depends on the ratings and opinions of other users.

– Content-based RS: a system that attempts to recommend items based on similarities to items that the user liked in the past. It relies on the movie's description or details, although CF depends on the movie's user rating [8].
– Hybrid RS: a system based on a combination of two algorithms for better performance [9]. Hybrid RSs are perceived by many organizations to be the most useful type of RS because of their capacity to eliminate defects that might have occurred when an RS was used and also because their performance is better than that of a two-RS composite [10]. The alternating least squares (ALS) algorithm enables CF between products and users to identify products that users might like based upon their previous ratings.

The remainder of this paper is organized. Section 2 provides a short introduction to Apache Spark. Section 3 covers our methodology resolution of the cold start issue in RSs and illustrates the various components of the suggested framework. Section 4 discusses the experimental MovieLens datasets 1M, 10M and 20M. Section 5 describes the experimental MovieLens datasets 1M, 10M and 20M. Section 5 describes the experimental outcomes and results of an evaluation.

– Introduction to Apache spark

Spark is a common platform for the study of distributed big data processing. It was developed in 2009 at the University of California, Berkeley. Spark is 100 times faster than Hadoop [11]. It is a distributed open-source framework that uses distributed memory to analyze a large amount of data. Spark is a supporter of a resilient distributed dataset (RDD) that enables the quick processing of a large amount of data [12]. Spark is a data processing framework that runs on the java virtual machine (JVM) environment. It uses Hadoop as a storage engine and operates its data processing cluster management. Unlike Hadoop MapReduce, it reduces the memory computing of data.

Spark provides core APIs in several programming languages. It can use structured data such as comma-separated values (CSV) and unstructured data such as JavaScript object notation (JSON). Coding is performed in several programming languages, including Java, Scala, R and Python. Spark also provides other libraries, such as structured query language (SQL), a machine learning library (MLlib) and a graph computing library (GraphX). It works in the cluster managers Standalone, Yarn, and Mesos [13]. Spark has two noteworthy features, resilient distributed datasets (RDDs) and directed acyclic graphs (DAGs).

MLlib is Spark’s largest and most popular distributed machine learning library. It contains quick and scalable instances of common machine learning algorithms and a variety of basic analytical tools. MLlib implements a range of classic machine learning algorithms, including various linear models (support vector machines, logistic regression (LR) and linear regression), naïve Bayes and random forest for classification and regression problems; ALS for CF; k-means for clustering and dimensionality reduction and frequent-pattern (FP) growth for frequent-pattern mining [14]. Spark includes three means of monitoring Spark applications, a web user interface (UI), metrics and external interfaces. The Spark UI shows other facets of the application, which are displayed by default on port 4040 [15].

Using additional information to illuminate the cold start issue such as social tags, user profile and trust [16]. A tag is marked as a type of text information. In the recommendation model, tags can be considered as a link between users and products and reflecting the user’s preference [17]. In the system, there are a lot of tags that have the same semantics. Extending the tags to get a more precise classification result. Tag extension is the process of finding similar tags for each tag or calculating tag similarity. To quantify the similarity between tags using cosine similarity [18].

2. RELATED WORK

Song *et al.* [19] proposed recommendation was based on a weighted bipartite graph and logistic regression method. The user-item bipartite graph is created and user similarities are measured using the bipartite graph’s weights. The similarity is used to calculate user predicted ratings to find top n recommendation with the highest predicted ratings. The value of mean absolute error (MAE) was 0.301 and root mean squared error (RMSE) was 0.812. Reddy and Narasimha [20] presented a statistical prediction dependent on decision tree and logistic regression algorithms for getting the movie rankings. The comparative study found that logistic regression performed 65% better than decision tree and a statistical prediction that are used to rank movies.

To increase recommendation performance, Pan *et al.* [21] proposed a new social tag expansion model to create a user profile. so instead of including the most appropriate tags, the new model presented the completeness of user-profiles by increasing tags and utilizing their relationships. The recall@20 0f the proposed method was 40.4%. He *et al.* [22] tagging features are used in the suggested recommender system to provide relevant recommendations on group discussion the semantic relevance of tags is collected using the WordNet database and then tags are arranged into a hierarchical based on their semantic relevance. The precision of the proposed system was 68.75, the recall was 36.45 and the f-measure was 0.47.

3. METHODOLOGY

3.1. Dataset

Here, assessing three versions of the dataset (1M, 10M and 20M MovieLens) that have been obtained from the real world and sometimes used to evaluate RSs. MovieLens datasets are downloaded from the GroupLens lab website [23].

- MovieLens-20M : it contains 27,278 movies, 138,00 users with 20,000,263 ratings and 465,564 tags [24]. The ratings are on a scale of {0.5,1,1.5,…,5} .
- MovieLens-10M: it contains about 71,567 users, 10,681 movies and 10,000,054 ratings in the range [1-5] and 95,580 tags.
- MovieLens-1M: it contains the information about 6,040 users, 3,900 movies and 1,000,209 ratings in the range [0.5-5]. The range of rating from 0 to5, 0 with being worst and 5 represent the best value [25].

This section presents a comprehensive evaluation of the proposed model on three real-world data 1M MovieLens, 10M and 20M MovieLens. The model is actualized offline to obtain top-K recommendations for a new user. Figure 1 shows the structure of the proposed technique. First, a preprocessing step is performed on the MovieLens dataset. Second, features are extracted using a utility matrix. Then, an ALS algorithm is applied to reduce features, followed by principal component analysis (PCA). Finally, the probability of movies is calculated using LR with relevant tags and sorted in descending order to determine top-N recommendations.

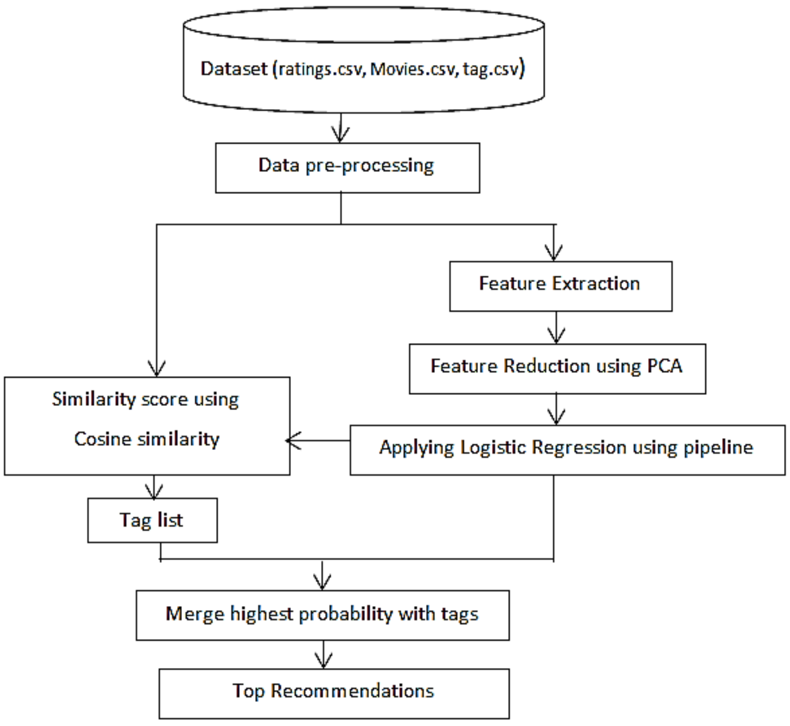
Dataset (ratings.csv, Movies.csv, tag.csv)
Data pre-processing
Feature Extraction
Feature Reduction using PCA
Similarity score using Cosine similarity
Applying Logistic Regression using pipeline
Tag list
Merge highest probability with tags
Top Recommendations

Figure 1. The overall flow of the proposed system

3.2. Data preprocessing

The proposed calculation uses the 1M, 10M, 20M benchmark dataset of MovieLens. The 1M MovieLens comprises 1,000,209 ratings and the rating scale has a range 0.5-5. The 10M MovieLens comprises 10,000,005 ratings and Rating scales in the range 1-5. The dataset of 20M MovieLens comprises 20 million ratings. Users can rate a movie on a range of 1-5 and also can tag to a movie. MovieLens users who have fewer than 30 ratings and timestamp columns are excluded during preprocessing. In movies.csv file removing '|' between genres column. In tags.csv file: remove special character '!','[',']' and stop words as 'was',' because', 'my',' few',' after' and 'had'. The number of users before was 138493 and after preprocessing become 110615. The rating value is labeled as '1' for a rating above 3 of rating value and labeled '0' for a rating below 3 of rating value. Rating CSV file is used and stored on Hadoop distributed file system (HDFS) to extract features from it [26] as shown in Figure 1.

3.3. Feature extraction

The utility matrix (also referred to as the user-item matrix) containing all the user ratings for each movie is generated. The user-item matrix U is the size matrix containing all of the ratings, where U_i is the user, M_i is the movie and R_i is the rating [27], as shown in Table 1.

Table 1. User-item matrix

	M1	M2	M3	M4	M5	M6	M7
UI	4			5	1		
U2	5	5	3				
U3				2	4	5	
U4		3					3

3.4. Feature selection

One of the popular algorithms in collaborative filtering is the alternating least square (ALS) algorithm. In this paper, ALS as appeared in Algorithm (1) is applied on rating files that contain UserId, MovieId and rating. MLlib provides the ALS algorithm with an implementation that relies upon the set of some parameters including rank, λ and numIterations. Mathematically, the main job to find two metrics, A (|M| × P) and B (|N| × P). So the product U ≈ A × B^T= U', where A is the users' latent factors and B is the latent factors of items. The normalized loss function shown in the following formula must be minimized [28].

$$(u_i, v_j) = min_{ui,vj} \sum_{(i,j\in k)} (u_i v_j{}^T - r_{ij})^2 + \lambda(||u_i||^2 + ||v_j||^2) \quad (1)$$

where $r_{i,j}$ refers to rating by a user, $u_i v_j^T$ shows dot product between user u and item v, λ shows regularization parameter, u_i is a user feature vector and v_j depicts item feature vector.

Algorithm 1: ALS Model

```
Input: user Rating File (userId,MovieId,Rating)
Output: productFeature <MovieId,FeatureVector>
userFeature <userId,FeatureVector>
Begin:
1.      Load the rating.csv file data into RDD.
        File → load rating.csv
2.      ParsingRating is a function specified by the user that will be split comma (,)
        ParseRating → map(ParseRating)
3.      Using cache( ) to store data in memory.
4.      Random splitting into 80% for training and 20% for testing.
        Split → randomSplit(Array(0.8,0.2))
5.      Store the two fields of testRDD
        testing → RDD map( MovieId, Rating)
6.      For i =Array ( 1 to n) n different values of λ
7.      For j=Array (1 to m) m different values of lambda
8.      create a recommendation model.
        Model → ALS.train(training, rank, numIteration)
        prediction → model.predict(testingRDD)
9.      Get RMSE using regression metrices.
        regressionMetrices → RegressionMetrics(prediction)
10.     save output <product Feature, user Feature> as a parquet file.
End
```

3.5. Dimensionality reduction

PCA is an unsupervised technique used to decrease the dimensionality of high-dimensional datasets to extract latent useful features and to extract valuable highlights from inactive and inappropriate highlights. This makes handling large amounts of information more effective and improves findings by decreasing the number of dimensions. PCA is a statistical method for decreasing the number of variables while preserving specific details [29].

PCA is a well-known statistical tool for reducing the number of features. It offers a lower error rate than other forms of dimension reduction [30]. PCA is recommended when treating three- or higher-dimensional data to create comprehensible visualizations [31].

PCA is used to obtain more exact results. The output of ALS, (ProductFeature <MovieId,productVector>). ProductVector, is used as the input for the PCA algorithm. Before applying PCA, we use the VectorAssembler() method to collect multiple features into one column called inputCol. There are a few bounds, called pcaFeatures. on setting the PCA model using the setInputCol( ) method as the input column, the setOutoutCol() method as the output column and making the transformation fit the DataFrame as characterized in Algorithm 2. MLlib's PCA estimator is important for creating a new DataFrame that contains the PCA feature vector.

Algorithm 2: Proposed model
```
Input: MovieId, featureVector, genres
Output: Top N Recommendations
1.      load dataset MovieId,featureVector,gen
2.      splitting dataset into 80% as training and 20% as testing.
3.      Using vectorAssembler to combine a given list of columns into a single vector
        column.
        Assember $\rightarrow$ array("id"," Features")
4.      Apply pca to reduce dimensions.
        pca $\rightarrow$ setInputcol("Features")
        setOutputCol("pcaFeatures")
5.      logistic regression requires Features and labels.
        Featurecol $\rightarrow$ PCAFeatures
6.      combine PCA with logistic regression using the pipeline.
        Classifier $\rightarrow$ LogisticRegression() .SetFeatureCol(pcaFeatures).SetLabel(Label)
        pipeline $\rightarrow$ setstages(assembler,pca,classifier)
        model $\rightarrow$ pipeline.fit(training)
        prediction $\rightarrow$ model.transform(testing)
7.      filter label equal 1 as postive values.
8.      for new user
MovieId,probability $\rightarrow$ Top N recommendations.
```

3.6. Logistic regression

LR is a mathematical process on a dataset in which the result is determined by one or more independent variables. The result is calculated using binary variables [32]. LR is the best method for forecasting the probability, where the outcome is yes or no, 0 or 1 [33]. LR has been generally utilized in the field of recommendation systems as a classic classifier. It is a linear model commonly utilized for the estimation of probabilities [34]. LR can take two forms:

- Binomial: the variables can only have 2 values '0' or '1' that can be positive or negative.
- Multinomial: the variables can include 3 or more values.

Binomial logistic regression is used in this study to indicate liking and disliking a movie [35].

LR can be utilized to predict the user's movie preferences. The target function in LR, $h(x)=(W^Tx)$ is converted using logistics function with W a vector of weights equal to x dimension. LR is utilized to predict the positive probability of the Movie and the probability be in the range [0,1] [36]. Utilizing binary classification, so the algorithm outputs that forecasts the new sample of data by applying a logistic function or sigmoid function as (2),

$$g(z) = \frac{1}{1+e^{-z}} \tag{2}$$

where g(z) has the probability of positive values, $z = w^Tx$ with default if $(w^Tx) > 0.5$, the output is positive otherwise negative [37].

The threshold is used to make the output binary (in binary classification cases) and in this case, it assumes the probability that the class will be positive. A value greater than 0.5 equals 1 and the value which less than or equal 0.5 equals 0 [38]. PCA features column is used as input for the logistic regression algorithm. The predictive probabilities can be useful in user rankings according to their likeliness. In the case of binary classification, the probability of a positive class should be taken [39].

The result of the LR model is the probability of the predicted rating. Only labels with values equal to one are used; then, they are arranged in descending order, as indicated by the probabilities in Figure 2. Sorting the probability of movies with genres descending based on movie column.

3.7. Tag similarity

Tags described how about different users feelings about the movie, as a result, the tags grow in number and hence become more descriptive and the recommendation got to be more exact [40]. Each tag was either a sentence or a single word. Each has a specific meaning that is important to the user. There was a list of tags that correspond to each movie. The substance of tag extension was to find similar words for each tag or to determine how similar the tags are. To quantify the similarity between tags by using cosine similarity.

The angle of cosine between two feature vectors is measured by cosine similarity. It was possible to be done by calculating the dot product of the two identities [41]. When numerous words have a similar meaning so the system in some cases failed to understand the difference between two similar words and thus failed to produce the desired result. The similarity between tags was calculated from the formula:

$$CosSim(x,y) = \frac{\sum_i xy}{\sqrt{\sum_i x^2}\sqrt{\sum_i y^2}} \quad (3)$$

In the experiment, using cosine similarity for the relevant tags specially for comedy in MovieLens dataset as shown in Table 2,

The top six relevant tags for comedy "hilarious"," humor"," funny"," spoof"," farce"," dark comedy" merged with probabilities to find the highest probability with tags for each movie. In the case of choosing a comedy movie genre, the result will be the highest probability of the comedy and when applying relevant tags to calculate the similarity of the words or represent the comedy and find out the similarity score of them.

Table 2. Relevant tags with similarity for comedy in MovieLens dataset

Relevant tags	Similarity
Hilarious	0.521
Humor	0.519
Funny	0.499
Spoof	0.474
Farce	0.459
Dark comedy	0.349

4. RESULTS AND DISCUSSION

4.1. Implementation tools

Here, describing the use of the MovieLens dataset to evaluate the proposed approach. All of the experiments were performed on virtual box and involved creating a virtual machine as a single node. The proposed framework was implemented using the scala programming language [11]. The test was conducted on a machine with the descriptions shown in Table 3. A machine with Ubuntu 16.4, Intel® Core™ i5-2400 and 4GB memory was used.

Table 3. Implementation tools used in the proposed approach

Operating system	Ubuntu 16.4
CPU	Intel® Core™ i5-2400 CPU @ 3.10 GHz 4 processor
Environment	spark -2.4.0, Jdk(java 8), Maven version3, Apache Hadoop-2.7.2, scala-2.11.7, SBT-0.13.9. Eclipse 4.7.0(oxygen)

4.2. Evaluation metrics

Five evaluation criteria have been calculated to evaluate the proposed recommendation system: Root mean square error (RMSE), mean absolute error (MAE), precision @N, recall @N and accuracy.

a) Root mean square error (RMSE) is calculated as [42],

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(R_{ui} - R^{\wedge}{}_{ui})^2} \quad (4)$$

where R_{ui} is the predicted rating of an item for a user, $R\hat{}_{ui}$ is the actual rating and N is the total number of ratings in the item set.

b) Mean absolute error (MAE) is used to calculate the error between values of predictions and target values as [43],

$$MAE = \frac{1}{N}\sum_{u,i}|R_{ui} - R^{\wedge}{}_{ui}| \tag{5}$$

c) Precision is the number of items that are rated in the recommendation list. Precision measures the rates of items in the recommendation that the user likes. Precision @N is used to compute for N items instead of all items. K is a size of recommendation list used to find top N recommendations.

$$Pr\,e\,cision = \frac{|C \cap R|}{|R|} \tag{6}$$

where C is the item that has correctly recommended and R is the item that has been incorrectly recommended.

d) Recall is measured out of the total number of products collected by the user in the suggestion list. Recall @k where K refers to the size of the recommendation list.

$$Re\,c\,all = \frac{|C \cap R|}{|C|} \tag{7}$$

e) Accuracy is one for evaluating the classification model. Accuracy is a fraction of the predicted model.

$$Accuracy = \frac{Numberofcorrectprediction}{total\,n\,umberofprediction} \tag{8}$$

After running Algorithm 1, find the assessed ideal estimations of two boundaries of the ALS model, rank and regularization parameter λ. In 20M, rank has coverage of {10, 50, 70, 100} and λ coverage of {0.01, 0.1, 1, 10} on the 80% training and 20% testing. All varieties of parameters were checked and evaluated based on their RMSE. The ideal RMSE found in case when rank equaled to 100 and λ was 10, the value of RMSE reached 0.707. In Algorithm 2, applying the PCA algorithm on the product-feature vector to reduce the feature vector and the result in PCA feature. Then, LR is used to obtain the probability of each movie by using pipeline in the PCA algorithm and LR to create the model.

In Figure 2, the proposed model makes estimations: MAE = 0.501, RMSE = 0.708, Precision@25 = 0.504 and Recall@25 = 0.529 on MovieLens-20M. Table 4 collates the comparison of the proposed model on Movielens-20M with other models [44]. In the Movielens-1M dataset, the proposed model has values of MAE is 0.515, RMSE is 0.7176, Precision@20 is 0.4964 and Recall@20 is 0.5112. Initially, to show the efficacy of the proposed approach, PCA-LR, in Table 5 there are equate to a few traditional approaches. The values of RMSE and MAE of the proposed model are lower than other models as appearing in Figure 3. Both Precision and Recall are to be the highest for the ideal recommendation. The accuracy of the proposed model has been compared to modern recommender system models in Table 6. In Figure 4 the proposed model has an accuracy of 0.8806, 0.8791 and 0.8739 on the 1M, 10M and 20M MovieLens dataset. The accuracy of the proposed model is higher than other approaches as shown in Table 7.

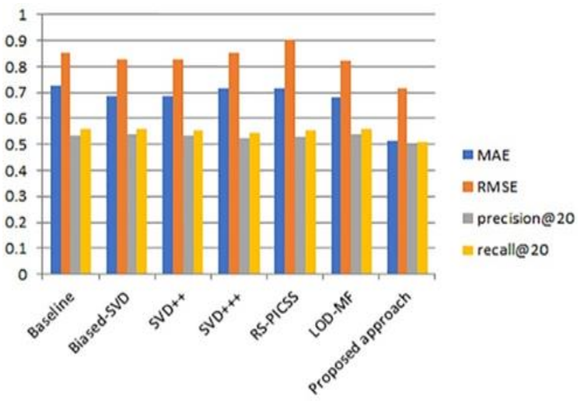

Figure 2. Performance techniques using 20M MovieLens

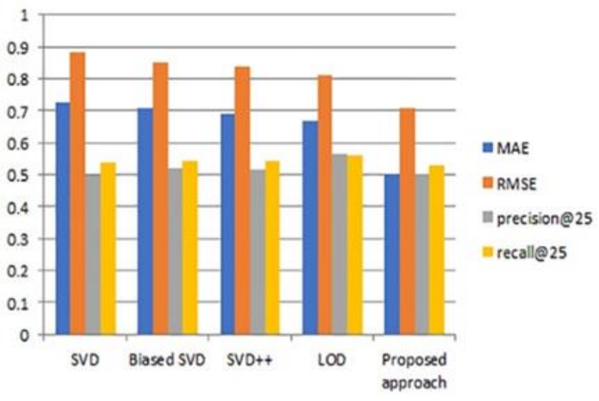

Figure 3. Performance techniques using 1M MovieLens

Table 4. Performance of different techniques using 20M MovieLens

	MAE	RMSE	Precision @25	Recall @25
Baseline (SVD) [44]	0.724	0.885	0.498	0.538
Biased Singular Value Decomposition	0.708	0.851	0.522	0.545
Singular value Decomposition ++	0.689	0.837	0.518	0.542
Open Linked Data based Matrix factorization	0.670	0.811	0.564	0.560
Proposed approach	0.5008	0.7076	0.504	0.529

Table 5. Performance of different techniques using 1M MovieLens

	MAE	RMSE	Precisin @20	Recall @20
Baseline SVD [45]	0.7243	0.8511	0.5361	0.5589
Biased-SVD	0.6874	0.8291	0.5375	0.5605
SVD++	0.6850	0.8276	0.5321	0.5566
SVD+++	0.7153	0.8518	0.5221	0.5454
RS-PICSS	0.7152	0.9017	0.5311	0.556
LOD-MF	0.6799	0.8246	0.5389	0.5605
Proposed approach	0.5150	0.7176	0.4964	0.5112

Table 6. Accuracy comparison between 1M and 10M MovieLens

Model	MovieLens 1M	MovieLens 10M
DKN [46]	0.589	0.658
RippleNet	0.844	0.869
SHINE	0.732	-
convMF	0.846	0.870
convMF+	0.849	0.875
contx-NMF	0.852	0.878
Proposed Model	0.8806	0.8791

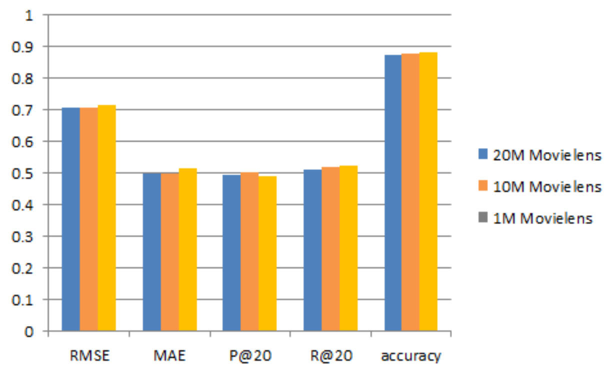

Figure 4. Performance techniques of 1M, 10M and 20M MovieLens

Table 7. Comparison between 1M, 10M and 20M MovieLens

	RMSE	MAE	P@20	R@20	ACC
1M MovieLens	0.7176	0.515	0.4964	0.5112	0.8806
10M MovieLens	0.7059	0.498	0.502	0.521	0.8791
20M Movielens	0.7076	0.5008	0.4911	0.523	0.8739

5. CONCLUSION

In this paper, a new method is proposed in the recommender system based on the logistic regression algorithm and the principal component analysis algorithm. The proposed model provides the probability for each movie in the MovieLens dataset. By applying alternating least squares to find the feature vector of the product, then using the principal component analysis algorithm to reduce these feature vectors. The prediction of the probability of movies depending on the logistic regression algorithm. Tag extension's to find similar words for each tag and determine the similarity score by using cosine similarity. The most appropriate tags are merged with the probabilities to find the highest probability for each movie. This model is trained by applying the logistic regression machine learning classification algorithm, which is giving the accuracy of 0.8806, 0.8791 and 0.8739 on MovieLens 1M, MovieLens 10M and MovieLens 20M dataset. In future work, we will develop the extraction process so that can get more specific features and even making a new classification machine learning classification algorithm on another dataset can improve model accuracy.

REFERENCES

[1] V. Agarwal and A. Vijayalakshmi, “Recommender system for surplus stock clearance,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, pp. 3813-3821, 2019, doi: 10.11591/ijece.v9i5.pp3813-3821.

[2] S. Chen and Y. Peng, “Matrix factorization for recommendation with explicit and implicit feedback,” *Knowledge-Based Syst.*, vol. 158, no. May, pp. 109-117, 2018, doi: 10.1016/j.knosys.2018.05.040.

[3] J. Kim, D. Hwang, and H. Jung, “Product recommendation system based user purchase criteria and product reviews,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 6, pp. 5454-5462, 2019, doi: 10.11591/ijece.v9i6.pp5454-5462.

[4] K. Kaur and V. Bharti, "*A Survey on Big Data-Its Challenges*," Springer Singapore, 2019.

[5] M. Schedl, H. Zamani, C. W. Chen, Y. Deldjoo, and M. Elahi, “Current challenges and visions in music recommender systems research,” *Int. J. Multimed. Inf. Retr.*, vol. 7, no. 2, pp. 95-116, 2018, doi: 10.1007/s13735-018-0154-2.

[6] W. Guanchen, M. Kim, and H. Jung, “Personal customized recommendation system reflecting purchase criteria and product reviews sentiment analysis,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2399-2406, 2021, doi: 10.11591/ijece.v11i3.pp2399-2406.

[7] G. Ramakrishnan, V. Saicharan, K. Chandrasekaran, M. V. Rathnamma, and V. V. Ramana, "*Collaborative Filtering for Book Recommendation System*," vol. 1057. Springer Singapore, 2020.

[8] I. M. Wartana, N. P. Agustini, and S. Sreedharan, “Hybrid model for movie recommendation system using content K-nearest neighbors and restricted boltzmann machine,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 23, no. 1, 2021, doi: 10.11591/ijeecs.v23.i1.ppab-cd.

[9] B. B. Sinha and R. Dhanalakshmi, “Evolution of recommender system over the time,” *Soft Comput.*, vol. 23, no. 23, pp. 12169-12188, 2019, doi: 10.1007/s00500-019-04143-8.

[10] Z. Batmaz, A. Yurekli, and A. Bilge, “A review on deep learning for recommender systems : challenges and remedies,” *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 1-37, 2019, doi: 10.1007/s10462-018-9654-y.

[11] K. Dahdouh, A. Dakkak, L. Oughdir, and A. Ibriz, “Large-scale e-learning recommender system based on Spark and Hadoop,” *J. Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0169-4.

[12] M. A. Rahman, A. Hossen, J. Hossen, C. Venkataseshaiah, T. Bhuvaneswari, and A. Sultana, “Towards machine learning-based self-tuning of hadoop-spark system,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 15, no. 2, pp. 1076-1085, 2019, doi: 10.11591/ijeecs.v15.i2.pp1076-1085.

[13] E. Shaikh, I. Mohiuddin, Y. Alufaisan, and I. Nahvi, “Apache Spark: A Big Data Processing Engine,” *2019 2nd IEEE Middle East North Africa Commun. Conf. MENACOMM 2019*, no. November 2019, doi: 10.1109/MENACOMM46666.2019.8988541.

[14] T. Vu, L. H. Pham, T. K. Huynh, and S. V. H. B, "*Vehicle Classification in Nighttime Using*," vol. 1, no. October. Springer Singapore, 2018, doi: 10.1007/978-981-10-7512-4_66.

[15] H. Lian, Z. Qin, H. Song, and T. H. B, "*E-CAT : Evaluating Crowdsourced*," vol. 1. Springer Singapore, 2018.

[16] Z. Zhang, M. Dong, K. Ota, and Y. Kudo, “Alleviating New User Cold-Start in User-Based Collaborative Filtering via Bipartite Network,” *IEEE Trans. Comput. Soc. Syst.*, vol. 7, no. 3, pp. 672-685, 2020, doi: 10.1109/TCSS.2020.2971942.

[17] H. Chen, W. Yan, H. Sun and M. Cheng, “Tag-Extended Collaborative Filtering Recommendation Algorithm,” *SN Comput. Sci.*, vol. 1, no. 5, pp. 1-13, 2020, doi: 10.1007/s42979-020-00316-7.

[18] D. P. He, Z. L. He, and C. Liu, “Recommendation algorithm combining tag data and naive bayes classification,” *Proc.-2020 3rd Int. Conf. Electron Device Mech. Eng. ICEDME 2020*, 2020, pp. 662-666, doi: 10.1109/ICEDME50972.2020.00156.

[19] W. Song, P. Shao, and P. Liu, "*Hybrid recommendation algorithm based on weighted bipartite graph and logistic regression*," vol. 1001. Springer Singapore, 2019, doi:10.1007/978-981-32-9298-7_13.

[20] D. A. Reddy and G. Narasimha, "A multi-criteria decision approach for movie recommendation using machine learning techniques," *Intelligent System Design*, vol. 1171, pp. 425-433, 2021, doi: 10.1007/978-981-15-5400-1_44.

[21] Y. Pan, Y. Huo, J. Tang, Y. Zeng, and B. Chen, “Exploiting relational tag expansion for dynamic user profile in a tag-aware ranking recommender system,” *Inf. Sci. (Ny).*, vol. 545, pp. 448-464, 2021, doi: 10.1016/j.ins.2020.09.001.

[22] M. He, K. Yao, P. Yang, and Y. Yao, "*Tag2Vec: Tag Embedding for Top-N Recommendation*," vol. 1075. Springer International Publishing, 2020.

[23] H. Koohi and K. Kiani, “Two new collaborative filtering approaches to solve the sparsity problem,” *Cluster Comput.*, vol. 6, 2020, doi: 10.1007/s10586-020-03155-6.

[24] N. Duong-Trung, Q. N. Nguyen, D. N. Le Ha, X. S. Ha, T. T. Phan, and H. X. Huynh, “Genres and actors/actresses as interpolated tags for improving movie recommender systems,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 2, pp. 67-74, 2020, doi: 10.14569/ijacsa.2020.0110210.

[25] P. P. Rokade and D. Aruna Kumari, “Business recommendation based on collaborative filtering and feature engineering - Aproposed approach,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 4, pp. 2614-2619, 2019, doi: 10.11591/ijece.v9i4.pp2614-2619.

[26] S. Panigrahi, R. K. Lenka, and A. Stitipragyan, "A Hybrid Distributed Collaborative Filtering Recommender Engine Using Apache Spark," *Procedia Comput. Sci.*, vol. 83, no. BigD2M, pp. 1000-1006, 2016, doi: 10.1016/j.procs.2016.04.214.

[27] K. Pradeep, I. K. Pradeep, and M. Jaya Bhaskar, "Comparative analysis of recommender systems and its enhancements," *Artic. Int. J. Eng. Technol.*, vol. 7, no. 3, pp. 304-310, 2018, [Online]. Available: https://www.researchgate.net/publication/328231954.

[28] H. Li, Y. Liu, Y. Qian, N. Mamoulis, W. Tu, and D. W. Cheung, "HHMF: hidden hierarchical matrix factorization for recommender systems," *Data Min. Knowl. Discov.*, vol. 33, no. 6, pp. 1548-1582, 2019, doi: 10.1007/s10618-019-00632-4.

[29] F. Lecron and F. Fouss, "An optimization model for collaborative recommendation using a covariance-based regularizer," *Data Min. Knowl. Discov.*, vol. 32, no. 3, pp. 651-674, 2018, doi: 10.1007/s10618-018-0552-3.

[30] M. Jafarzadegan, F. Safi-Esfahani, and Z. Beheshti, "Combining hierarchical clustering approaches using the PCA method," *Expert Syst. Appl.*, vol. 137, pp. 1-10, 2019, doi: 10.1016/j.eswa.2019.06.064.

[31] R. Rahutomo, A. S. Perbangsa, H. Soeparno, and B. Pardamean, "Embedding Model Design for Producing Book Recommendation," *Proc. 2019 Int. Conf. Inf. Manag. Technol. ICIMTech 2019*, no. August, 2019, pp. 537-541, doi: 10.1109/ICIMTech.2019.8843769.

[32] W. Song, P. Shao, and P. Liu, "Hybrid recommendation algorithm based on weighted bipartite graph and logistic regression," *Commun. Comput. Inf. Sci.*, vol. 1001, no. September, pp. 159-170, 2019, doi: 10.1007/978-981-32-9298-7_13.

[33] N. M. Samsudin, C. F. B. Mohd Foozy, N. Alias, P. Shamala, N. F. Othman, and W. I. S. Wan Din, "Youtube spam detection framework using naïve bayes and logistic regression," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 14, no. 3, pp. 1508-1517, 2019, doi: 10.11591/ijeecs.v14.i3.pp1508-1517.

[34] H. Tian, H. Cai, J. Wen, S. Li, and Y. Li, "A Music Recommendation System Based on logistic regression and eXtreme Gradient Boosting," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2019-July, no. July, 2019, pp. 1-6, doi: 10.1109/IJCNN.2019.8852094.

[35] H. S. Priyanka, B. V. Ramya, and K. Ashok, "Classification Model To Determine The Polarity Of Movie Review Using Logistic Regression," *Int. Res. J. Comput. Sci*. vol. 6, no. 6, pp. 87-91, 2019.

[36] Y. Wang, D. Feng, D. Li, X. Chen, Y. Zhao, and X. Niu, "A mobile recommendation system based on logistic regression and Gradient Boosting Decision Trees," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2016-Octob, no. July, 2016, pp. 1896-1902, doi: 10.1109/IJCNN.2016.7727431.

[37] M. Saliha, B. Ali, and S. Rachid, "Towards large-scale face-based race classification on spark framework," *Multimed. Tools Appl.*, vol. 78, no. 18, pp. 26729-26746, 2019, doi: 10.1007/s11042-019-7672-7.

[38] J. Xianya, H. Mo, and L. Haifeng, "Stock Classification Prediction Based on Spark," *Procedia Comput. Sci.*, vol. 162, no. Itqm, pp. 243-250, 2019, doi: 10.1016/j.procs.2019.11.281.

[39] A. Dubey, A. Gupta, N. Raturi, and P. Saxena, "Item-based collaborative filtering using sentiment analysis of user reviews," *International Conference on Application of Computing and Communication Technologies*, vol. 899, pp. 77-97, 2018, doi: 10.1007/978-981-13-2035-4_8.

[40] P. Meel, F. Bano, A. Goswami, and S. Gupta, "*Content-Based and Collaborative*," Springer Singapore.

[41] R. H. Singh, S. Maurya, T. Tripathi, T. Narula, and G. Srivastav, "Movie Recommendation System using Cosine Similarity and KNN," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 5, pp. 556-559, 2020, doi: 10.35940/ijeat.e9666.069520.

[42] F. Anwar, N. Iltaf, H. Afzal, and H. Abbas, "A Deep Learning Framework to Predict Rating for Cold Start Item Using Item Metadata," *Proc.-2019 IEEE 28th Int. Conf. Enabling Technol. Infrastruct. Collab. Enterp. WETICE 2019*, 2019, pp. 313-319, doi: 10.1109/WETICE.2019.00071.

[43] H. P. Tian and E. J. Chen, "Research on Collaborative Filtering Algorithm Based on Spark Platform," *Proc.-2017 Int. Conf. Ind. Informatics - Comput. Technol. Intell. Technol. Ind. Inf. Integr. ICIICII 2017*, vol. 2017-Decem, 2018, pp. 33-36, doi: 10.1109/ICIICII.2017.25.

[44] S. Natarajan, S. Vairavasundaram, S. Natarajan, and A. H. Gandomi, "Resolving data sparsity and cold start problem in collaborative filtering recommender system using Linked Open Data," *Expert Syst. Appl.*, vol. 149, 2020, doi: 10.1016/j.eswa.2020.113248.

[45] R. Wang, H. K. Cheng, Y. Jiang, and J. Lou, "A novel matrix factorization model for recommendation with LOD-based semantic similarity measure," *Expert Syst. Appl.*, vol. 123, pp. 70-81, 2019, doi: 10.1016/j.eswa.2019.01.036.

[46] Z. Khan, N. Iltaf, H. Afzal, and H. Abbas, "Enriching Non-negative Matrix Factorization with Contextual Embeddings for Recommender Systems," *Neurocomputing*, vol. 380, pp. 246-258, 2020, doi: 10.1016/j.neucom.2019.09.080.