I.J. Intelligent Systems and Applications, **2020, 2, 11-20**
Published Online April 2020 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2020.02.02

The Empirical Comparison of the Supervised Classifiers Performances in Implementing a Recommender System using Various Computational Platforms

Ali Mohammad Mohammadi
School of computer science, institute for research in fundamental science (IPM), P.o.Box 19395-5746, Tehran, Iran.
E-mail: mohammadi81ali@gmail.com

Mahmood Fathy
School of computer science, institute for research in fundamental science (IPM), P.o.Box 19395-5746, Tehran, Iran.
E-mail: mahfathy@ipm.ir

Received: 02 August 2019; Revised: 29 August 2019; Accepted: 12 September 2019; Published: 08 April 2020

***Abstract*—**Recommender Systems (RS) help users in making appropriate decisions. In the area of RS research, many researchers focused on improving the performances of the existing methods, but most of them have not considered the potential of their employed methods in reaching the ultimate solution. In our view, the Machine Learning supervised approach as one of the existing techniques to create an RS can reach higher degrees of success in this field. Thus, we implemented a Collaborative Filtering recommender system using various Machine Learning supervised classifiers to study their performances. These classifiers implemented not only on a traditional platform but also on the Apache Spark platforms. The Caret package is used to implement the algorithms in the classical computational platform, and the H2O and Sparklyr are used to run the algorithms on the Spark Machine. Accordingly, we compared the performance of our algorithms with each other and with other algorithms from recent literature. Our experiments indicate the Caret-based algorithms are significantly slower than the Sparklyr and H2O based algorithms. Also, in the Spark platform, the runtime of the Sparklyr-based algorithm decreases with increasing the cluster size. However, the H2O-based algorithms run slower with increasing the cluster size. Moreover, the comparison of the results of our implemented algorithms with each other and with other algorithms from recent literature shows the Bayesian network is the fastest classifier between our implemented classifiers, and the Gradient Boost Model is the most accurate algorithm in our research. Therefore, the supervised approach is better than the other methods to create a collaborative filtering recommender system.

Index Terms—Distributed Machine learning, Supervised classifiers comparison, Recommender System, Apache Spark, Deep Multilayer Perceptron.

I. Introduction

In the recent decade, the interest in Recommender Systems (RS) has significantly increased [1]. In this area, Collaborative Filtering is one of the most effective techniques [2, 3]. To implement the CF-based Recommender Systems (CFRS) the Matrix Factorization (MF), Singular Value Decomposition (SVD), and neighborhood method have achieved significant improvements [4, 5]. In the MF approach, the users' previous interests to the items encoded into a rating matrix and this matrix indicates the similarities between the users and items [6, 7]. However, these algorithms suffer from Sparsity and Cold-start problems [8], and these problems reduce the accuracy of the algorithms [6]. From the Machine Learning (ML) perspective, the Cold-start and Sparsity problems [9] have a common root in the scarcity of the labeled data to train recommender algorithms [10, 11]. In this regard, we want to compare the ability of the ML supervised algorithms in creating a CF-based RS when the related dataset is not big and find the most appropriate supervised algorithm for this task.

Our work consists of two major parts. First, we train and test many supervised algorithms on the classical platform. Next, we do the same experiments on the parallel processing platform using Apache Spark [12] technology to study the capabilities of each algorithm in creating a CFRS using Movielens 100-k dataset [13]. The employed algorithms are the Decision Tree (DT) [14], Random Forest (RF) [15], Support Vector Machine [16], Bayesian Network (SVM) [17], Generalized Linear Model (GLM) [18], Gradient Boost Model (GBM) [19], and the Multilayer Perceptron Neural Network (MPNN) [20]. And, we implement a Deep multilayer Perceptron Neural Network (DMPNN) [21] on parallel processing platform to study its capability to classify our structured dataset.

Accordingly, we compare the performances of our supervised algorithms with each other, also with the other similar works from the literature. The experiments and studies indicate that when the dataset has a structured format and the number of its features is not numerous thus the ML supervised solution and the manual feature selection is still the best approach to make a CFRS. Also, when we used Apache Spark platform to increase the processing speed, on the single node Apache machine, our supervised algorithms ran significantly faster than the classical approach, and when we added more computers to the cluster, the speed of the Sparklyr [22] based supervised classifiers increased; however, with adding more nodes to the cluster the training time of the H2O [23] based algorithms significantly increased.

Amongst all of the employed algorithms in this paper, our most accurate classifier is the Gradient Boost Model from the H2O package. Also, most of our algorithms, specifically our DMPNN, perform better than most of the algorithms presented in recent literature.

The rest of the article is organized as follows: Section 2 represents the related works. Section 3 introduces the research questions. Section 4 presents all of the details about the algorithms and methods. Section 5 describes the performances of the algorithms. Finally, section 6 discusses the conclusion and future research ideas.

II. Related Work

In this literature review, the focus is on the RS algorithms which they trained using the Movielens dataset.

Yuan et al. [24] proposed a recommendation system called the imputation-based Singular Value Decomposition to solve the data Sparsity problem. They used MovieLens 100k in their work. The Root Mean Square Error (RMSE) [24, 25] for their method is 0.8821 in the best case. Costa et al. [25] claimed that due to the MovieLens sparsity, single recommender system cannot perform as well as multiple algorithms using Ensemble method on this dataset. This experiment revealed that the joint results of the multiple recommender algorithms are highly reliable than the single algorithm. Singh et al. [26] claimed that the user's interest in the specific item can change over time, and this is one of the essential factors that affect the accuracies of the algorithms. Thus, to address this problem, they used the k-means algorithm to analyze the MovieLens dataset to find the popularities of the items for the users at a different period of the time. Al-Bakri et al. [27] proposed a recommendation system using the K-means algorithm. They preprocess the Dataset to improve classification accuracy.

Luo et al. [28] developed a method called co-SVD that reduces the Sparsity problem more than other methods. This Sparsity reduction increased the accuracy of their algorithm. Hazrati et al. [29] proposed a model-based Pair-wise CF approach that uses the Restricted Boltzmann Machine to find the feature map. Zhang et al. [30] used a deep neural network to predict the rating scores on MovieLens dataset to provide a collaborative filtering recommender system. First, their model uses a quadric polynomial regression method as a feature representation. Then, these features are considered as the input data of their algorithm to predict the rating scores. The RMSE result for this algorithm is %0.9874. Liu et al. [31] introduced a method called Deep Retentive learning to predict demographic information using MovieLens dataset. They used an automatic feature selection technique instead of a handcrafted method. Also, they use deep neural networks to predict the rating data. The RMSE result for this algorithm is 0.845 %. Lee et al. [32] proposed a collaborative filtering algorithm based on Deep Neural Network algorithm. They use normalized user-rating and item-rating vector as the inputs to the algorithm, and use batch normalization technique to tackle the over-fitting problem. They claim that their methods achieve RMSE of 0.907% on MovieLens-100k, and 0.848% on MovieLens-1M. Yan et al. [33] used a Deep auto-encoder and Convolutional text network to create a recommender system. Their method combines all of the features from User and Item tables to address the Sparsity and cold start recommendation problem. They claim that their algorithm is superior to the many algorithms introduced in the literature. The RMSE of their proposed model on the MovieLens-100K dataset is 0.914%, and on the MovieLens-1M is 0.859%. Barbieri et al. [34] proposed a method to transform a Collaborative Filtering problem into Supervised Learning problem. Their Auto-encoder based Collaborative Filtering System (A-COFILS) achieved a Mean Absolute Error (MAE) of 0.697% in MovieLens-100k dataset and MAE of 0.661% for MovieLens-1M. Also, they compared the RMSE score of the different CF techniques. Accordingly, the algorithms and their results are as follows. The RMSE score for A-COFILS is 0.885% and for the Baseline-COFILS is 0.892%, and the Kernel-PCA-COFILS obtained the RMSE of 0.898%. Wu et al. [35] introduced an RS method based on the user-rating centrality. Their proposed method is trained with Movielens-100k and they gained the RMSE of 0.8983%. Yao et al. [36] proposed a user-user neighborhood approach. Their method addressed the Sparsity problem. They gained better accuracy by removing the sparse data. The best RMSE for their algorithm is 0.9440% with a learning rate of 0.015%. Ma and Gan [37] proposed a method which predicts the rating data of the films using Random Forest Regression algorithm and the MAE of their algorithm is 0.749%. Berg et al. [38] proposed an Auto-encoder based method that uses Matrix completion approach to find the best proposal for the users. The RMSE for their algorithm is 0.909%. Fu et al. [39] introduced a method using deep learning to provide recommendations with analyzing the connections between items and users. They used a feed-forward network and pre-trained model for prediction. The RMSE of their algorithm (Multi-views Neural Network) on MovieLens-1M is 0.830% with the training time of 8 minutes, and RMSE for MovieLens-10M is 0.776% with 2 hours of training time. Gedikli et al. [40] proposed an RS that called FR-Rec. This algorithm generates the

predictions simply by looking for frequency of rating in the usual user-item rating matrix. The RMSE score for this algorithm on the MovieLens-100k dataset is ~1.06%, and for the parameterized version of their algorithm is ~0.93%. Richter et al. [41] performed the comparison of four main open source tools that are used to run the distributed machine learning algorithms; these tools are Mahout, MLlib, H2O, and SAMOA and they analyzed three characteristics of the tools namely availability of the methods, scalability, and speed. They claimed the H2O is the best package.

According to the literature, in the last decade, most of the CFRS algorithms were based on the Matrix factorization and Singular Value Decomposition, and still researchers working on these methods to improve the performances. Two of the most well-known problems in this approach are the Sparsity and the Cold start problems. These problems mostly raise due to the nature of the CF dataset properties. Currently, to solve these problems and increase the accuracy of the algorithms, many researchers started to use machine learning classifier. However, there are the scarce of empirical and comparative research in the area of the CFRS which widely analyze the advantages of the supervised classifiers over the traditional methods.

III. Research Objectives

Our main objectives are as follows: (1) the empirical comparison of the various ML classifiers in classical and distributed ML platforms (Apache Spark), and comparing the three different frameworks (Caret, Sparklyr, and the H2O) to analyze their usefulness to implement these algorithms. (2) Studying the capability of the DMPNN algorithm to classify small datasets such as Movielens 100-K. (3) Comparison of the performance of machine learning supervised classifiers with other algorithms such as SVD (Matrix Factorization approach) for creating a CF-based RS.

IV. Material and Methods

A. Supervised Algorithms

All of the algorithms that we used to classify the dataset are as follows:

Decision Tree and Random Forest [15]: a Decision Tree algorithm in the learning process splits the input dataset into the subsets and repeats the process on each subset using a recursive partitioning technique, and ends this recursion when splitting no longer adds value to the predictions. The Random Forest algorithm is an ensemble classifier that uses many decision trees, and the outputs are the results of a union of the individual trees.

Support Vector Machine [16]: this algorithm is a discriminative classifier that performs classification by finding the hyperplane which tries to maximize the margin between two classes. Basically, it is a linear classifier and uses kernel functions to classify the data.

Bayesian Network [17]: this algorithm predicts the classes based on the probability estimation. This algorithm uses the Bayesian Inference theory.

Generalized Linear Model [18]: this algorithm is a flexible generalization of an ordinary linear regression that allows the response variables that have error distribution models other than a normal distribution.

Gradient Boost Model [19]: this model is a technique to solve the regressions and classifications problems. It builds the ensemble model in a stage-wise fashion using weak prediction models and produces a better-generalized model.

Artificial Neural Network [20]: this network is a computing system inspired by the biological neural networks, and are composed of a large number of highly interconnected processing units called neurons that these neurons are working in unison to solve specific problems.

Deep Artificial Neural Network [21]: this algorithm is a complex artificial neural network that uses sophisticated mathematical approaches to solve complex problems. One of the dependencies of a Deep Algorithm to achieve better performance is the size of the dataset. The predictive ability of this algorithm increases when the training dataset gets bigger.

B. Packages and Libraries

In this research, we used the Caret [42], Sparklyr, and H2O which are the function libraries to implement ML algorithms.

The Caret is a functions library for training and plotting classification and regression models, and it has several functions to streamline the model building and evaluation process.

The Sparklyr is a general engine for big data processing and run R codes on Parallel Processing approach, and provides a dplyr [43] compatible back-end [44], and supports connecting to Apache Spark clusters.

The H2O is the scalable platform to implement the parallel algorithms. Table 1, illustrates the packages and related algorithms that we used in this research.

C. Preprocessing

In Machine Learning projects to obtain better results from the applied algorithms, the dataset has to be in a proper format. Thus, we removed Null values and performed the numeric transformation (e.g., log scale), and data normalization. Table 2 indicates a summary of the User and Rating table before and after normalization.

D. Feature Selection

Feature selection is an important technique used in data preprocessing. The goal of the feature selection is to find the most relevant features for the task. We used all of the relevant features to create a collaborative filter by predicting the rating. These features are listed below.

rating ~ age + gender + occupation + Action + Adventure + Animation + Children + Comedy + Crime + Documentary + Drama + Fantasy + FilmNoir + Horror + Musical + Mystery + Romance + SciFi + Thriller + War + Western

Table 1. Supervised algorithms have been employed in this research

Algorithms	Caret	Sparklyr	H2O
Random forest (RF)	Yes	Yes	Yes
Decision Tree (DT)	Yes	Yes	Yes
Support Vector Machine (SVM)	Yes	-	-
Bayesian Network (BN)	Yes	Yes	Yes
Generalized Linear Model (GLM)	Yes	-	Yes
Gradient Boosting Model (GBM)	-	-	Yes
Artificial Neural Network (ANN)	Yes	Yes	Yes
Deep Neural Network (DNN)	-	-	Yes

Table 2. User and Rating table summary of the MovieLens-100k dataset

User Summary		Ratings Summary	
Users	Normalized users	Ratings	Normalization ratings
Min.: 1.0	Min.: 0.0000	Min.: 1.00	Min.: 0.0000
1st Qu.: 254.0	1st Qu.: 0.2576	1st Qu.: 3.00	1st Qu.: 0.5000
Median: 447.0	Median: 0.3485	Median: 4.00	Median: 0.7500
Mean: 462.5	Mean: 0.3935	Mean: 3.53	Mean: 0.6325
3rd Qu.: 682.0	3rd Qu.: 0.5000	3rd Qu.: 4.00	3rd Qu.: 0.7500
Max.: 943.0	Max.: 1.0000	Max.: 5.00	Max.: 1.0000

E. Optimization

We trained all of the algorithms with default optimization parameters except the GBM and DMPNN. We choose the GBM due to its prior competence in classifying our data using default optimization parameters and the DMPNN to find the best fine-tuning approach to optimize this neural network to perform the task.

1. The GBM optimization

For the optimized version of the GBM, the optimization parameters initialized as follows:

The number of trees is 20. The learning rate is 0.2. The Max depth of each tree is 10. The stopping round is 2. Stopping tolerance is 0.01. And the model type is "gbm_covType2".

2. The DMPNN optimization

The fine-tuning parameters for the DMPNN initialized as follows:

rate=0.001, rate_annealing=2e-6, momentum_start=0.2, momentum_stable=0.4, momentum_ramp=1e7, l1=1e-5,
l2=1e-5, max_w2=20, score_validation_samples=10000,
max_categorical_features = 2147483647,
reproducible = TRUE,
export_weights_and_biases = TRUE,
mini_batch_size = 1, stopping_rounds=2,
input_dropout_ratio= 0.1,
stopping_metric="misclassification",
stopping_tolerance=0.01

For the DMPNN algorithm, not only we apply the optimization, but we also create many versions of this network to explore and find the most appropriate structure for it. These versions are different only in the size of their hidden layers, the number of the neurons in each hidden layer, and the type of their activation functions. Table 9 illustrates the details of each different version, and Fig. 1 indicates our best model of the DMPNN.

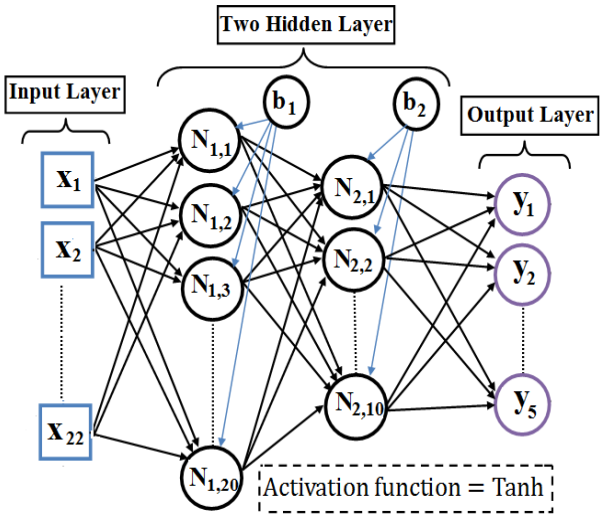

Fig.1. The structure of our most accurate Deep Multi-Layer Perceptron Neural Network with only two hidden layers and Tanh activation function

V. Experiment Analysis

A. Dataset

We use the MovieLens-100k dataset to train all algorithms. This dataset includes 100,000 ratings from 943 users with 1682 movies [13].

B. Hardware platform

The experiments carried out using DELL INSPIRON Core-i3 as a server (Master) and all of the algorithms executed from this computer.

C. Metrics

We evaluated the performance of the algorithms using Accuracy and Root Mean Square Error (RMSE) methods.

The accuracy is the measure of the degree of closeness of calculated value to its actual value [26]. Accuracy is defined as:

$$\text{Accuracy} = \frac{(\#\text{TP}) + (\#\text{TN})}{(\#\text{TP}) + (\#\text{TN}) + (\#\text{FP}) + (\#\text{FN})} \qquad (1)$$

The RMSE is the standard deviation of the prediction errors [26]. RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} \left(P_i - \hat{q}_i \right)^2}{N}} \qquad (2)$$

Here, P_i is predicted and $\hat{q}_i$ is actual rating of item i. and N shows the total number of predicted item. In our work the ratings more than 3 are considered as a high rating (recommended items), and lower than 3 as a low rating (not recommended items).

D. Execution times of the algorithms

We analyzed *the* run times of the algorithms, and the following section indicates the results.

1. The Caret-based algorithms

All of the algorithms in this section implemented with Caret package, and according to the experiments, the Bayesian network with the training time of 1.2280 seconds is the fastest algorithm in convergence, and the Decision tree with 0.6360 seconds is the fastest in prediction time, also the Random Forest with the training time of 12 hours is the slowest algorithm. Table 3 indicates the training and test time for each algorithm.

2. The Sparklyr-based algorithms

In this section of the experiments, all of the algorithms are implemented using the Sparklyr package. Also, the algorithms executed with two different computational powers. First, we used a single computer to configure it as a master node of the Spark machine and use it to run the algorithms. Next, build a cluster of two computers and trained all of the algorithms again. Accordingly, by distributing the algorithms on two computers, all of the algorithms converged faster. Also, between all of the employed algorithms, the Bayesian network is the fastest in training time. Moreover, the MLP algorithm is more expensive in training stage, but its inference time is much faster than other algorithms. Table 4 indicates more details about the algorithms in this experiment.

Table 3. The runtime of the algorithms implemented with the Caret package

Hardware	**Time**	**Algorithms – Caret**					
		Bayesian	Tree	Random Forest	MLP	GLM	SVM (Radial)
Master node (4 core) Total Memory: 0.84 GB	Train	1.22807 seconds	2.546679 minutes	12.80152 hours	1.37019 hours	44.93274 minutes	1.784533 hours
	Test	16.96497 seconds	0.6360362 seconds	4.846277 seconds	1.838106 seconds	0.3470201 seconds	4.165922 minutes

Table 4. The runtime of the algorithms implemented with the Sparklyr package

	Hardware	**Time**	**Algorithms – Sparlyr**			
			Bayesian	Tree	Random forest	MLP
1	Master, (4 core) Total memory: 0.84 GB	Train	7.646437 Seconds	25.08443 Seconds	56.68924 Seconds	1.760384 Minutes
		Test	0.518029 Seconds	0.4910278 Seconds	0.4690268 Seconds	0.419024 Seconds
2	Master + Workers (5 core) Total memory: 1.64 GB	Train	3.922224 Seconds	13.21176 Seconds	49.0358 Seconds	40.04729 Seconds
		Test	0.35902 Seconds	0.463026 Seconds	0.4080229 Seconds	0.210012 Seconds

3. The H2O-based algorithms

In this section of the experiments, all of the algorithms have been implemented using the H2O package, and the algorithms have been executed using three various sizes of the Spark cluster.

First, we configured an Apache Spark server using a single computer and run all of the algorithms in the same condition.

Next, we built a Spark cluster by adding a few computers to our Spark server and trained the algorithms in this cluster.

Again, we extended the cluster size by adding more computers to the last cluster and trained our algorithms once more.

Therefore, each algorithm runs three times using three different cluster sizes. Accordingly, the results of the algorithms indicate that with increasing the cluster size, the algorithms run slower. For instance, in the single node Spark machine, the run time of the Random Forest is 39 seconds, and with increasing the cluster size, the run time of the algorithm significantly increased. Table 5 indicates the training and test time of each algorithm and the details about each cluster.

Table 5. The runtime of the algorithms implemented with the H2O version: 3.20.0.1

Hardware		Time	Algorithms – H2O					
			Random Forest	Bayesian	Multi GLM	GBM	Optimized GBM	Deep Learning c(20,10) tanh
1	Master, (4 core) Total memory: 0.84 GB	Train	39.58926 Seconds	2.586148 Seconds	3.003172 Seconds	21.0362 Seconds	15.82291 Seconds	11.59766 Seconds
		Test	0.1830111 Seconds	0.1090062 Seconds	0.1440079 Seconds	0.1060061 Seconds	0.025002 Seconds	0.140008 Seconds
2	Master+ Workers, (5 core) Total memory: 1.64 GB	Train	5.992343 Minutes	4.965284 Seconds	3.609206 Seconds	1.525754 Minutes	43.29448 Seconds	16.39994 Seconds
		Test	0.1700099 Seconds	0.1580091 Seconds	0.2260129 Seconds	0.1340082 Seconds	0.01800084 Seconds	0.155009 Seconds
3	Master+ Workers, (14 core) Total memory: 3.03 GB	Train	15.14852 Minutes	3.72882 Seconds	3.811221 Seconds	3.243541 Minutes	1.240903 Minutes	22.283844 Seconds
		Test	0.3616018 Seconds	0.3774011 Seconds	0.3260012 Seconds	0.3462 Seconds	0.02200103 Seconds	0.1570021 Seconds

E. Accuracies of the algorithms

1. The accuracy comparison of the Caret-based algorithms

According to the table 6, between the algorithms that we implemented using Caret package, the Random forest with 36.28% is the most accurate algorithm.

2. The accuracy comparison of the Sparklyr-based algorithms

In this section of our experiments, the Logistic Regression with accuracy of 35.29% is the most accurate algorithm, and the Decision Tree with accuracy of 35.11% is in the second place. Table 7 indicates the accuracy of the algorithms.

3. The accuracy comparison of the H2O-based algorithms

Table 8 illustrates the results of the algorithms based on H2O. According to the results, the accuracy of the algorithms is close to each other, and the GBM is the most accurate algorithm between them.

Table 6. The accuracy of the algorithms implemented with the Caret package

Algorithm	Accuracy
Random Forest	0.3612%
Bayesian	0.3075%
Tree	0.3559%
MLP C(20,10)	0.2674%
GLM	0.3550%
SVM	0.3602%

Table 7. The accuracy of the algorithms implemented with the Sparklyr package

Algorithm	Accuracy
Random Forest	0.3494%
Bayesian	0.3492%
Tree	0.3511%
MLP C(20,10)	0.3527%

Table 8. The accuracy and RMSE of the algorithms implemented with the H2O package

Performances		Algorithms					
		Random forest	Bayesian	Multi GLM	GBM	Optimized GBM	Deep Learning c(2,10)
Accuracy		0.3617%	0.3047%	0.3422%	0.3694%	0.3804%	0.3401%
MSE		0.5298143	0.588088	0.5586138	0.5404639	0.5278379	0.5534342
RMSE		0.7278834	0.766869	0.7474047	0.7351625	0.7265245	0.7439316
Log loss		1.422561	2.064796	1.448472	1.401826	1.387695	1.449757
Mean per-class Error		0.7171554	0.7636593	0.7903271	0.7348674	0.7067766	0.7786473
Top-5 Hit Ratios	1th	0.361668	0.304777	0.342218	0.369456	0.380451	0.340136
	2th	0.650535	0.576194	0.624214	0.659115	0.664695	0.629503
	3th	0.845363	0.792637	0.827121	0.848986	0.858940	0.832493
	4^{th}	0.948649	0.920037	0.940652	0.949440	0.952105	0.938736
	5^{th}	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Table 9. Details of each version of our Deep Multilayer Perceptron and related RMSE values

N	Model Configuration		Evaluation	
	Hidden Layer	Activation Function	MSE	RMSE
1	c(200, 100, 50, 5)	Rectifier	0.5659594	0.7523027
2	c(200, 100, 50, 5)	Rectifier Dropout	0.5659634	0.7523054
3	c(200, 100, 50, 5)	Tanh	0.5584915	0.7473229
4	c(200, 100, 50, 5)	Tanh Dropout	0.5662589	0.7525017
5	c(200, 100, 50, 5)	Rectifier	0.5659594	0.7523027
6	c(200, 100, 50, 5)	Rectifier Dropout	0.5659634	0.7523054
7	c(200, 100, 50, 5)	Tanh	0.5584915	0.7473229
8	c(200, 100, 50, 5)	Tanh Dropout	0.5662589	0.7525017
9	c(500,200, 100, 50, 5)	Rectifier	0.5705829	0.7553694
10	c(500,200, 100, 50, 5)	Rectifier Dropout	0.5659594	0.7523027
11	c(500,200, 100, 50, 5)	Tanh	0.5596489	0.7480968
12	c(500,200, 100, 50, 5)	Tanh Dropout	0.574533	0.7579795
13	C(200, 100, 50)	Tanh	0.556305	0.7458585
14	C(8,2)	Tanh	0.5555929	0.7453811
15	C(100,50)	Tanh	0.5602152	0.7484752
16	**C(20,10)**	**Tanh**	**0.5534342**	**0.7439316**
17	C(20,10)	Rectifier	0.5575525	0.7466944

4. Comparing the different DMPNN models

We build seventeen different versions of the DMPNN using the H2O package. The differences of these versions are in the size of the hidden layers and the activation functions. The 16th row of Table 9 indicates the network and its related configuration details that gained the best classification accuracy. This best model trained using two hidden layers and the Tanh activation function.

5. The optimization results of the GBM and DMPNN

The optimized version of the GBM gained better accuracy than the entire algorithm in this research and its RMSE is 0.7265245. Also, the optimized version of the DMPNN gained the RMSE of 0.7439316. All of the important details about the optimized version of our DMPNN are in the 16th row of Table 9, and Fig. 2 indicates its convergence to the optimum point.

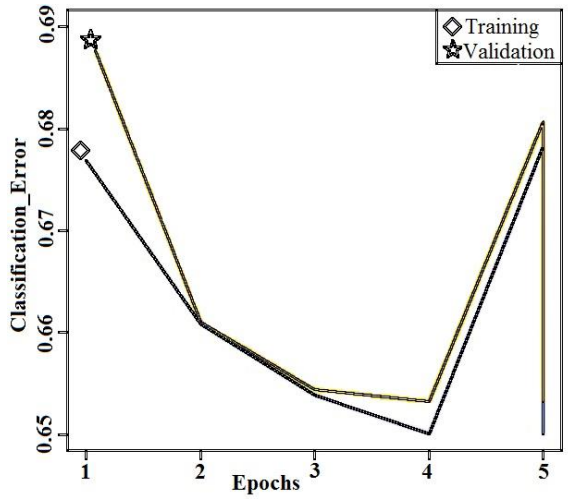

Fig.2. This plot is the result of the DMPNN with (20, 10) hidden layer and Tanh activation function

6. Comparing the packages and the algorithms by their accuracy and training time

As Table 10 indicates, we compared all of our algorithms by their accuracy and training time. The results revealed none of the H2O, Sparklyr, and Caret packages have an absolute advantage over each other.

According to the accuracy comparison, each one of our employed packages has some algorithms which are more accurate than similar algorithms from other packages. For example, the H2O based Random Forest, as well as the Caret based Random Forest, is more accurate than the Sparklyr based Random Forest algorithm, and the Bayesian algorithm from Sparklyr is more accurate than the Bayesian algorithms from the other two packages.

Table 10. The accuracy and Training time comparison of the algorithms

	Algorithms	Accuracy	Time
1	Optimized GBM (H2O)	0.3804%	15.82 seconds
2	GBM (H2O)	0.3694%	21.03 seconds
3	Random Forest (H2O)	0.3617%	39.58 seconds
4	Random Forest (Caret)	0.3612%	12.80 hours
5	SVM (Caret)	0.3602%	1.78 hours
6	Tree (Caret)	0.3559%	2.54 minutes
7	GLM (Caret)	0.3550%	44.93 hours
8	MLP (Sparklyr)	0.3527%	56.68 seconds
9	Tree (Sparklyr)	0.3511%	25.08 seconds
10	Random Forest (Sparklyr)	0.3494%	1.76 minutes
11	Bayesian (Sparklyr)	0.3492%	7.64 seconds
12	DMPNN (H2O)	0.3470%	11.59 seconds
13	Multi-GLM (H2O)	0.3422%	3.00 seconds
14	Bayesian (Caret)	0.3075%	1.22 seconds
15	Bayesian (H2O)	0.3047%	2.58 seconds
16	MLP (Caret)	0.2674%	1.37019 hours

The training time comparison indicates, most of the Caret based algorithms have the highest execution time. For instance, the accuracy of the Caret based GLM and the Random Forest algorithms are 44.93% and 12.80% hours, respectively. These algorithms are the slowest in this research. Also, the Bayesian from Caret package (as only Caret based algorithm which is faster than the H2O and Sparklyr based algorithms) is the fastest algorithm, and the Deep-MLP from H2O is faster than the classical MLP algorithms from Caret and Sparklyr packages.

Regardless of the implemented platforms and packages, the Bayesian is the fastest algorithm compared to all of the other algorithms in the comparison table, and the H2O based GBM is the most accurate algorithm in this research.

Table 11. The RMSE of our algorithms versus the algorithms from other research

	Algorithm	Dataset	RMSE
1	*Our GBM algorithm*	*Movielens-100K*	*0.7265*
2	*Our Deep NN Model*	*Movielens-100K*	*0.743*
3	Deep NN Model [30]	Movielens-100K	0.987
4	Deep NN Model [30]	Movielens-1M	0.935
5	SVD [30]	Movielens-100K	1.103
6	PMF [30]	Movielens-100K	1.068
7	MCoC [30]	Movielens-100K	1.094
8	DsRec [30]	Movielens-100K	0.992
9	PMMMF [30]	Movielens-100K	1.029
10	Hern [30]	Movielens-100K	1.102
11	SCC [30]	Movielens-100K	1.003
12	TyCo [30]	Movielens-100K	1.031
13	A-COFILS [4]	Movielens-100K	0.885
14	MFRC [5]	Movielens-100K	0.898
15	STAR-GCN [46]	Movielens-100K	0.879
16	STAR-GCN [46]	Movielens-1M	0.844
17	CFSVD-TF [47]	Movielens-100K	0.9762
18	NORMA [48]	Movielens-10M	0.7641
19	SVD [49]	Movielens-100K	2.67776
20	NMF [50]	Movielens-100K	0.94
21	ANNInit [51]	Movielens-100K	~0.90
22	IAI_All [51]	Movielens-100K	~0.90
23	IAI_SimMF [51]	Movielens-100K	~0.90
24	IAI_SimMF [51]	Movielens-1M	~0.86
25	KNN+CD RS [52]	Movielens-706u	0.9050
26	DeepFlexEncoder RS [53]	Movielens-100K	0.833
27	trusted k-coRating [54]	Movielens-100K	0.97
28	sim k-coRating [54]	Movielens-100K	0.97
29	average k-coRating [54]	Movielens-100K	0.99
30	random k-coRating [54]	Movielens-100K	1.0
31	Trusted k-coRating [54]	Movielens-1M	0.92
32	RBM Neural Network [55]	Movielens-100K	1.035

7. Comparing our best classifiers with other algorithms from the literature

As Table 11 indicates, we compared many algorithms that all of them classified the Movielens dataset to create a collaborative filtering recommender system. This comparison table includes the results of our GBM and DMPNN algorithms and many different algorithms from recent literature. We use GBM in this table due to its superiority over other classifiers in our experiments and use the DMPNN because the deep neural network-based classifiers are the most popular in recent researches. Studying the comparative table makes us come to a few important points as follows:

The GBM is not only the most accurate algorithm in our experiments, but it also is the best classifier in this comparison.

Our customized version of the DMPNN is outperforming the other deep neural networks in this comparison.

Other methods like SVD algorithms that use matrix factorization to analyze the rating table of the Movielens dataset are not as accurate as our supervised algorithms.

Although some of the researchers created hybrid algorithms to improve the accuracy of rating table classification, our simple supervised method performs more accurate than their approach.

VI. Discussion and Conclusion

In this paper, we discussed the effectiveness of the supervised approach to implementing a collaborative filtering recommender system in the classical and the Apache Spark platforms. Also, we fine-tuned a deep Perceptron neural network model (DMPNN) to study its ability to classify our data. According to the research results, the Apache Spark-based distributed algorithms are significantly better than none-distributed classical algorithms in terms of accuracy and runtime, and all of the implemented supervised classifiers including our deep Perceptron achieved good prediction accuracy than the other methods such as SVD which uses matrix factorization technique to create a collaborative filter, which proves that the applying the machine learning supervised classifiers to solve the collaborative filtering problems for implementing a more accurate recommender system is a successful attempt. In future work, we will build a hybrid recommender system based on the Ensemble method using deep Perceptron algorithm on a parallel processing platform, and explore the effects of different automatic feature selection techniques on our learning algorithm.

Acknowledgment

This work was supported by the Iranian institute for research in fundamental science (IPM).

References

[1] T. Mahmood, F. Ricci, “Improving recommender systems with adaptive conversational strategies.” In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, ACM, pp. 73-82, 2009.

[2] U. Shardanand, P. Maes, “Social information filtering: algorithms for automating “word of mouth”.” In *Chi*, vol. 95, pp. 210-217, 1995.

[3] M. Elahi, F. Ricci, N. Rubens, “A survey of active learning in collaborative filtering recommender systems.” *Computer Science Review*, vol. 20, pp. 29-50, 2016.

[4] Aggarwal, C. Charu, "An introduction to recommender systems." In *Recommender systems*, Springer, pp. 1-28, 2016.

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

[6] X. Su, T. M. Khoshgoftaar. "A survey of collaborative filtering techniques." Advances *in artificial intelligence*, vol. 2009, 2009.

[7] Y. Koren, R. Bell, C. Volinsky, "Matrix factorization techniques for recommender systems" *Computer*, vol. 8, pp. 30-37, 2009.

[8] G. Guo, "Resolving data sparsity and cold start in recommender systems." In *International Conference on User Modeling, Adaptation, and Personalization*, Springer, vol. 13, pp. 361-364, 2012.

[9] G. Guo, J. Zhang, D. Thalmann, "Merging trust in collaborative filtering to alleviate data sparsity and cold start" *Knowledge-Based Systems*, vol. 57, pp. 57-68, Feb 2014.

[10] Y. Zhu, J. Lin, S. He, B. Wang, Z. Guan, H. Liu, D. Cai, "Addressing the item cold-start problem by attribute-driven active learning." *IEEE Transactions on Knowledge and Data Engineering*, 2019, https://doi.ieeecomputersociety.org/10.1109/TKDE.2019.2891530

[11] S. Feng, "Sparsity in Machine Learning: An Information Selecting Perspective" (2019). *Doctoral Dissertations*. 1550. https://scholarworks.umass.edu/dissertations_2/1550

[12] A. Spark, "Apache Spark: Lightning-fast cluster computing." URL http://spark. Apache. org. 2016 Jun.

[13] F. Harper, J. Konstan, "The movielens datasets: History and context." *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, p. 19, 2016, https://grouplens.org/datasets/movielens/100k/

[14] J. R. Quinlan, "Decision trees as probabilistic classifiers." In *Proceedings of the Fourth International Workshop on Machine Learning*, Morgan Kaufmann, pp. 31-37, 1987, https://doi.org/10.1016/B978-0-934613-41-5.50007-6

[15] A. Jehad, R. Khan, N. Ahmad, I. Maqsood, "Random forests and decision trees." *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 272, 2012.

[16] B. Schölkopf, A. J. Smola, F. Bach. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[17] N. Friedman, D. Geiger, D. Goldszmidt, "Bayesian network classifiers." *Machine learning*. Vol. 29(2-3), pp. 131-63, 1997.

[18] P. McCullagh, *Generalized linear models*. Routledge, 2019.

[19] A. Natekin, A. Knoll, "Gradient boosting machines, a tutorial." *Frontiers in neurorobotics*, vol. 7, no. 21, 2013.

[20] L. Goodfellow, Y. Bengio, A. Courville, *Deep learning*. MIT press, 2016.

[21] L. Goodfellow, Y. Bengio, A. Courville, *Deep learning*. MIT press, 2016.

[22] J. Luraschi, K. Ushey, JJ. Allaire, "The Apache Software Foundation. sparklyr: R Interface to Apache Spark." *R package* (2018).

[23] S. Aiello, E. Eckstrand, A. Fu, M. Landry, P. Aboyoun, Machine Learning with R and H2O. 2018.

[24] X. Yuan, L. Han, S. Qian, G. Xu, H. Yan, "Singular value decomposition based recommendation using imputed data." *Knowledge-Based Systems*, vol. 163, pp. 485-94, 2019.

[25] A. Da Costa, M. Manzato, R. Campello, "Boosting collaborative filtering with an ensemble of co-trained recommenders." *Expert Systems with Applications*, vol. 1, no. 115, pp.427-41, 2019.

[26] P. Singh, P. Pramanik, N. Debnath, N. Choudhury, "A Novel Neighborhood Calculation Method by Assessing Users' Varying Preferences in Collaborative Filtering." *Proceedings of 34th International Confer*, vol. 58, pp. 345-55, 2019.

[27] N. AL-Bakri, S. Hashim, "Collaborative Filtering Recommendation Model Based on k-means Clustering." *Al-Nahrain Journal of Science*, vol. 22(1), pp. 74-79, 2019.

[28] L. Luo, H. Xie, Y. Rao, F. Wang, "Personalized recommendation by matrix co-factorization with tags and time information." *Expert Systems with Applications*, vol. 119, pp. 311-21, 2019.

[29] N. Hazrati, B. Shams, S. Haratizadeh, "Entity representation for pairwise collaborative ranking using restricted Boltzmann machine." *Expert Systems with Applications*, vol. 116, pp. 161-71, 2019.

[30] L. Zhang, T. Luo, F. Zhang, Y. Wu. "A recommendation model based on deep neural network." *IEEE Access*, vol. 6, pp. 9454-63, Jan 2018.

[31] Y. Liu, H. Qu, W. Chen, SH. Mahmud, "An Efficient Deep Learning Model to Infer User Demographic Information From Ratings." *IEEE Access*, vol. 7, pp. 53125-53135, 2019.

[32] H. Lee, J. Lee, "Scalable deep learning-based recommendation systems." *ICT Express*, vol. 5, no. 2, pp. 84-88, 2019.

[33] W. Yan, D. Wang, M. Cao, J. Liu. "Deep Auto Encoder Model With Convolutional Text Networks for Video Recommendation." *IEEE Access*, vol. 7, pp. 40333-40346, 2019.

[34] J. Barbieri, LG. Alvim, F. Braida, G. Zimbrão, "Autoencoders and recommender systems: COFILS approach." *Expert Systems with Applications*, vol. 89, pp. 81-90, 2017.

[35] Z. Wu, H. Tian, X. Zhu, S. Wang. "Optimization matrix factorization recommendation algorithm based on rating centrality." In *International Conference on Data Mining and Big Data*, pp. 114-125. Springer, Cham, 2018.

[36] X. Yao, B. Tan, C. Hu, W. Li, Z. Xu, Z. Zhang, "Recommend algorithm combined user-user neighborhood approach with latent factor model." In *International Conference on Mechatronics and Intelligent Robotics*, pp. 275-280. Springer, Cham, 2017.

[37] Y. Ma, M. Gan, "A Random Forest Regression-based Personalized Recommendation Method." In *PACIS*, p. 170, 2018.

[38] R. Berg, T. Kipf, M. Welling, "Graph convolutional matrix completion." *arXiv preprint arXiv:1706.02263* (2017).

[39] M. Fu, H. Qu, Z. Yi, L. Lu, Y. Liu, "A novel deep learning-based collaborative filtering model for recommendation system." *IEEE transactions on cybernetics, vol.49, no. 3, pp. 1084-1096, 2018.*

[40] F. Gedikli, F. Bagdat, M. Ge, D. Jannach, "RF-REC: Fast and accurate computation of recommendations based on rating frequencies." In *2011 IEEE 13th Conference on Commerce and Enterprise Computing*, pp. 50-57. IEEE, 2011.

[41] A. Richter, T. Khoshgoftaar, S. Landset, T. Hasanin, "A multi-dimensional comparison of toolkits for machine learning with big data." In *2015 IEEE International Conference on Information Reuse and Integration*, pp. 1-8,

IEEE, 2015.
[42] M. Kuhn, "The caret package." *R Foundation for Statistical Computing, Vienna, Austria.* URL https://cran. r-project. org/package= caret. 2012 Nov 26.
[43] H. Wickham, R. Francois, L. Henry, K. Müller, "dplyr: A Grammar of Data Manipulation." *R package version 0.4. 3. R Found. Stat. Comput.*, Vienna. https://CRAN. R-project. org/package= dplyr. 2015 Nov 13.
[44] H. Wickham, R. Francois, L. Henry, K. Müller, "dplyr: A grammar of data manipulation." *R package version 0.4*, 2015;3.
[45] White, Lyndon, Roberto Togneri, Wei Liu, Mohammed Bennamoun. "Introduction to Neural Networks for Machine Learning." In *Neural Representations of Natural Language*, pp. 1-21. Springer, Singapore, 2019.
[46] J. Zhang, X. Shi, S. Zhao, I. King, "STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems." *arXiv preprint* arXiv:1905.13129. 2019 May 27.
[47] J. Wang, P. Han, Y. Miao, F. Zhang, "A Collaborative Filtering Algorithm Based on SVD and Trust Factor." In *2019 International Conference on Computer, Network, Communication and Information Systems* (CNCI 2019) 2019 May. Atlantis Press.
[48] D. Li, C. Chen, Z. Gong, T. Lu, S. Chu, N. Gu, Collaborative Filtering with Noisy Ratings." In *Proceedings of the 2019 SIAM International Conference on Data Mining*, 2019 May 6 (pp. 747-755). Society for Industrial and Applied Mathematics.
[49] A. Sahoo, C. Pradhan, B. Mishra, "SVD based Privacy Preserving Recommendation Model using Optimized Hybrid Item-based Collaborative Filtering." In *2019 International Conference on Communication and Signal Processing (ICCSP)* 2019 Apr 4 (pp. 0294-0298). IEEE.
[50] M. Ahamed, S. Afroge, "A Recommender System Based on Deep Neural Network and Matrix Factorization for Collaborative Filtering." In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)* 2019 Feb 7 (pp. 1-5). IEEE.
[51] J. Zhao, X. Geng, J. Zhou, Q. Sun, Y. Xiao, Z. Zhang, Z. Fu, "Attribute mapping and autoencoder neural network based matrix factorization initialization for recommendation systems." *Knowledge-Based Systems, vol. 166, pp. 132-139, 2019*.
[52] H. Zhang, F. Min, Z. Zhang, S. Wang, "Efficient collaborative filtering recommendations with multi-channel feature vectors." *International Journal of Machine Learning and Cybernetics*, vol. 10, pp. 1165-72, 2019.
[53] D. Tran, Z. Hussain, W. Zhang, N. Khoa, N. Tran, Q. Sheng, "Deep Autoencoder for Recommender Systems: Parameter Influence Analysis." *arXiv preprint arXiv*:1901.00415. 2018 Dec 25.
[54] F. Zhang, V. Lee, R. Jin, S. Garg, K. Choo, M. Maasberg, L. Dong, C. Cheng, "Privacy-aware smart city: A case study in collaborative filtering recommender systems." *Journal of Parallel and Distributed Computing*, vol. 127, pp. 145-59, 2019.
[55] D. Chao, L. Kaili, Z. Jing, J. Xie, Chao, Duan, Lu Kaili, Zhang Jing, and Jerry Xie. "Collaborative Filtering Recommendation Algorithm Classification and Comparative Study." In *Proceedings of the 2019 4th International Conference on Distance Education and Learning*, pp. 106-111. ACM, 2019.

Authors' Profiles

Ali M. Mohammadi was born in Ardabil, Iran in 1981. He received the M.Sc. degree in Artificial Intelligence in 31 Jun 2018. His research interests are General A.I. and Machine Learning.

Mahmood Fathy received the B.S. degree in electronics from Iran University of Science and Technology, Tehran, Iran, in 1984, the M.S. degree in computer architecture from Bradford University, West Yorkshire, U.K., in 1987, and the Ph.D. degree in image processing computer architecture from the University of Manchester Institute of Science and Technology, Manchester, U.K., in 1991. His research interests are machine learning, deep learning, and machine vision.

How to cite this paper: Ali Mohammad Mohammadi, Mahmood Fathy, "The Empirical Comparison of the Supervised Classifiers Performances in Implementing a Recommender System using Various Computational Platforms", International Journal of Intelligent Systems and Applications(IJISA), Vol.12, No.2, pp.11-20, 2020. DOI: 10.5815/ijisa.2020.02.02