Recommender systems algorithm selection using machine learning

Nikolaos Polatidis and Stelios Kapetanakis

School of Computing, Engineering and Mathematics, University of Brighton, BN2 4GJ, Brighton, U.K
{N.Polatidis@Brighton.ac.uk, S.Kapetanakis@Brighton.ac.uk}

Elias Pimenidis

Department of Computer Science and Creative Technologies, University of the West of England, BS16 1QY, Bristol, U.K
{Elias.Pimenidis@uwe.ac.uk}

Abstract. This article delivers a methodology for recommender system algorithm selection using a machine learning classifier. Initially, statistical data from real collaborative filtering recommender systems have been collected to form the basis for a synthetic dataset since a real meta dataset doesn't exist. Once the dataset has been developed a classifier can be applied to predict which recommender system among a range of algorithms will predict better for a given dataset. The experimental evaluation shows that tree-based approaches such as Decision Tree and Random Forest work well and provide results with high accuracy and precision. We can conclude that machine learning can be used along with a meta dataset comprised of statistical information in order to predict which recommender system algorithm will provide better recommendations for similar datasets.

Keywords: Recommender Systems; Datasets; Meta recommender; Algorithm selection, Machine learning.

1 Introduction

Recommender systems have been widely used in e-Commerce domains for the recommendation of products or more specific items such as movies, music or jokes among others [1-2]. It is a technology that allows a company to provide personalized recommendations to users while having a domain in mind, which allows the discovery of more relevant items, reduce search time from a user point of view and increased sales from a company point of view. There are different ways to recommend products to users such as Collaborative Filtering which can be used to make recommendations to users according to common history with other users or content-based filtering which can be used to provide recommendations according to user preferences and item descriptions [1]. Furthermore, there are methods that can be used to recommend items if

these are similar to other items which is called item-based collaborative filtering. However, nowadays in most domains various algorithms of combinations of such are being used to provide all kinds of recommendations [2].

Extensive research in recommender systems has taken place usually in aspects such as algorithm improvement, application domains investigation and privacy protection [3-4]. While this is very good, with so much information and different application domains there is the need to be able to identify the best approach to use. Typically, a software developer will have to explore a variety of different algorithms for an approach such as user-based collaborative filtering where user ratings are being used and perform extensive evaluation tests to identify the most accurate recommendation method to use. The most straightforward way to identify the best method is time consuming, which will result to higher development costs, more time and possibly less testing to save time, thus resulting to a less accurate method.

Therefore, in this paper we developed a methodology that uses a meta dataset and machine learning classification to help researchers and software developers to identify which is the best recommendation algorithm to use according to their domain with regards to approaches where user ratings are used, and recommendation are generated according to previous common rating history between users.
The contributions of the paper are:

- Development of a synthetic meta classification dataset that includes characteristics of collaborative filtering recommendation datasets.
- Application of machine learning classification methods to predict the best recommendation algorithm.
- Experimental evaluation that shows that the proposed methodology is both practical and effective.

The rest of the paper is organized as follows: Section 2 is the background, section 3 describes the proposed methodology, section 4 explains the experimental evaluation and section 5 contains the conclusions.

2 Related work

There are many different works regarding algorithm development in recommender systems starting with traditional approaches such as the ones that are based on the Pearson or Cosine similarity to calculate similar users [6]. However, there are many more recommendation algorithms in the literature such as the multi-level collaborative filtering that breaks the Pearson similarity into multiple levels using thresholds [3]. Another method is a hybrid where collaborative filtering is integrated with meta search for product recommendation in e-Commerce [5]. There is another modified collaborative filtering approach where the neighborhood of users is formed according to highest ranking neighbors, thus increasing the accuracy [7]. One more modified collaborative filtering approach based on singularities this time is presented. The idea here is to consider contextual information collected from all users and calculate the singularity for each item [8]. Hybrid recommendation approaches are extensively discussed in ref. [9]. Here the authors explain different ways to combine algorithms together. In another work the

utilization of sparsity measures is presented as a way to increase the accuracy of collaborative filtering [10]. Another similarity method based on collaborative filtering that is based on more co-rated items is presented in [11]. Yet another recommendation method is one that is based on a hybrid user-based fuzzy collaborative filtering approach [12]. A somewhat different method is one that is used to correct noisy rating before collaborative filtering is applied [13]. Continuing in the collaborative filtering domain entropy can be used with collaborative filtering to calculate user similarity in recommender systems and improve accuracy over classical methods [14]. Deep learning can also be used to increase accuracy in collaborative filtering [15]. Personalized diffusions can be used to provide improved list of top-n recommendations [16]. Moreover, in improved top-N recommendation approaches conditional variational auto-encoders can be used [17]. A different collaborative filtering similarity metric where integral equations are used with linear differential equations and non-linear systems to calculate similarities between users is proposed in [18]. An approach that assumes sparsity is important is based on Bhattacharyya coefficient for collaborative filtering to improve accuracy [19]. Finally, a hybrid collaborative filtering algorithm that is based on Kullback-Leibler divergence is presented [20].

3 Proposed method

The proposed methodology section is based on a synthetic meta dataset. Due to the fact that there aren't any available recommender systems meta datasets we manually developed one that is based on characteristics of some of the most known collaborative filtering datasets. The first 14 entries of the dataset are based on statistical information from actual datasets and are shown in table 1. In total there are 150 entries in the meta dataset which are 14 real and 136 synthetic that are based on the statistical info of the first 14. For example, in most datasets the sparsity is very high, and the number of users is usually less than the number of ratings. Furthermore, there is a final prediction value at the final column which represents which algorithm is better for the particular dataset and this is the value that the classifier tries to predict. Numbers 1, 2 and 3 have been used to represent 3 recommendation algorithms with all values are randomly assigned. These 3 numbers represent which algorithm is better for the specific dataset and should be 2 or more options available each representing an algorithm. In this case these values do not represent a particular algorithm but rather explain how algorithm selection can take place.

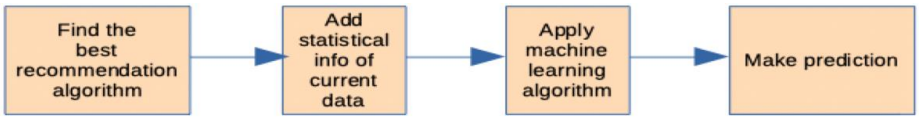
Table 1. Recommender Systems datasets.

Dataset	Users	Items	Ratings	Min value	Max value	Domain	Label
Ciao	7375	99746	278483	1	5	General	1

Duban	129490	58541	16830839	1	5	Movies	1
Epinions1	40163	139738	664824	1	5	General	2
Epinions2	71002	104356	508960	1	5	General	2
Epinions3	120492	755760	13668320	1	5	General	1
Flixster	147612	48794	8196077	0.5	5	Movies	3
FilmTrust	1508	2071	35497	0.5	4	Movies	3
Jester	59132	140	1761439	0	1	Jokes	1
Movielens1	943	1682	100000	1	5	Movies	1
Movielens2	6040	3706	1000209	1	5	Movies	2
Movielens3	71567	10681	10000054	1	5	Movies	2
MovieTweetings	69324	36383	88452	0	10	Movies	3
YahooMovies	7642	11915	211231	1	13	Movies	3
YahooAudio	15400	1000	311704	1	5	Music	1

Once the dataset has been developed a classifier such as the K nearest neighbors, Decision Tree, Random Forest, Artificial Neural Network Multi-Layer Perceptron or other can be applied to predict which recommendation algorithm is better for a given set of data that look alike other datasets for which we have ground truth. The diagram in figure 1 explains how the methodology works.

Figure 1. Proposed methodology



In the first step it is assumed that a researcher or developer wants to identify a good algorithm for their current data. At step 2, these data can be converted into numbers such as shown in table 1, except the label value at the end and in step 3 a classifier can predict which algorithm is good and presented in step 4. In offline evaluation conditions it is assumed that the label value is present, and a training/testing or cross fold validation approach can be used to identify which classifier is better.

4 Experimental evaluation

For the experimental evaluation we have used the dataset explained in section 3 and the Python programming language along with the Scikit learn machine learning API. 5-fold cross validation has been used in the experiments. Furthermore, for the evaluation the synthetic dataset that has been developed in the methodology section has been used which has collaborative filtering statistical information, while from the Scikit learn library the Random forest, Decision Tree, K-Nearest neighbor and Multi-layer perceptron classifiers have been used. The random forest classifier the number of estimators has been setup to 500 and the max features to 0.25. For the other classifiers the default settings have been used. The evaluation metrics have been used are the Accuracy defined in equation 1, Precision defined in equation 2, Recall defined in equation 3, F1 defined in equation and finally the AUC metric. All the metrics have been used from the Scikit learn API using 5-fold cross validation.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

The results are presented below in the nine graphs and one table. Figure 2 presents comparison results of the classifiers over 5 tests for the accuracy metric, figure 3 for the precision metric, figure 4 for the recall metric and figure 5 for the F1 metric. Figures 6, 7, 8 and 9 present the average accuracy, precision, recall and F1 results of the 5 tests presented in figures 2 to 5. Figure 10 presents the AUC results. Table 2 presents further evaluation results for the random forest classifier. It is shown in the results that random forest outperforms all the other classifiers in all test and metrics and that the methodology is accurate to an extend it is practical to use. We executed 5 different sets with different random data each time for training and testing and user different classifiers to be able to verify which classifier is the best for among others and for different random data.

Figure 2. Accuracy results

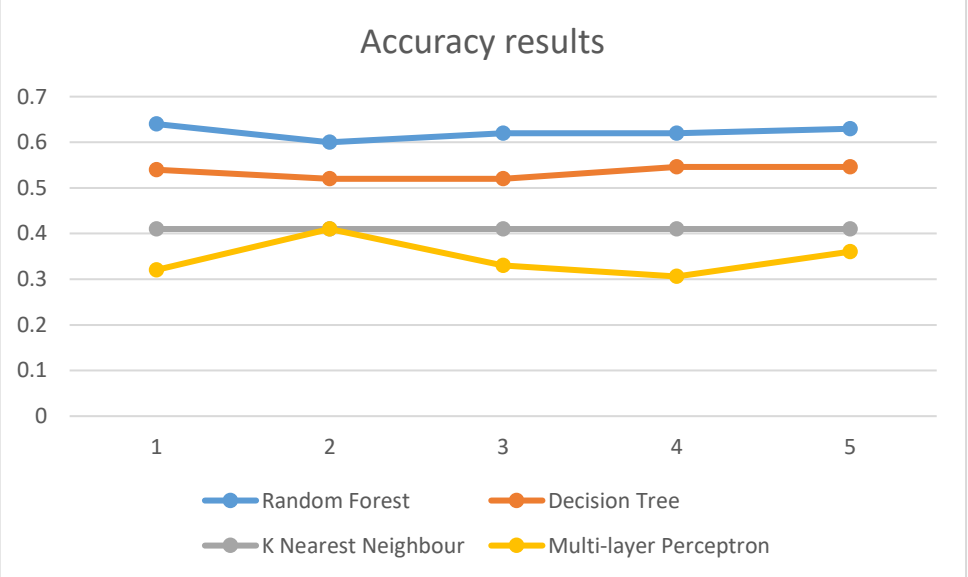


Figure 3. Precision

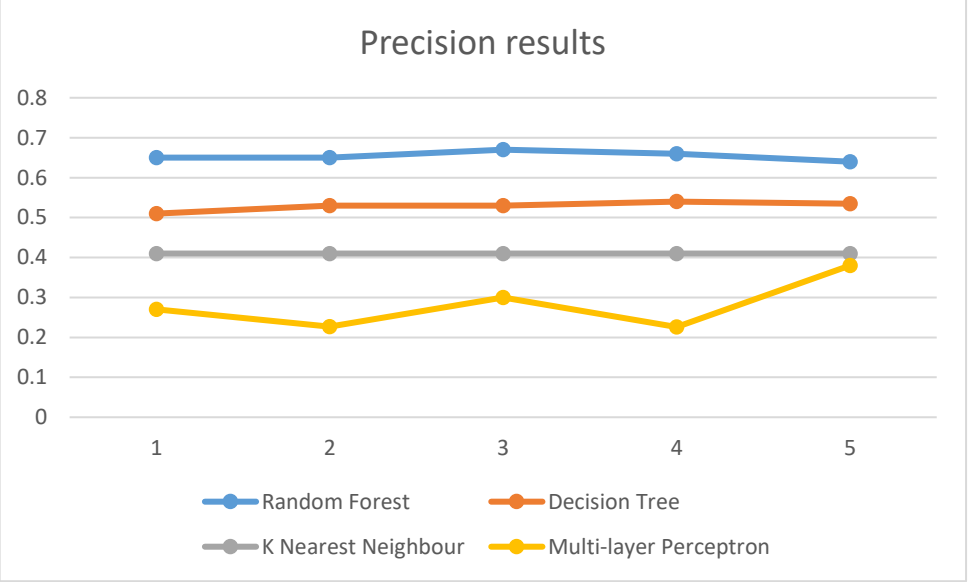


Figure 4. Recall results

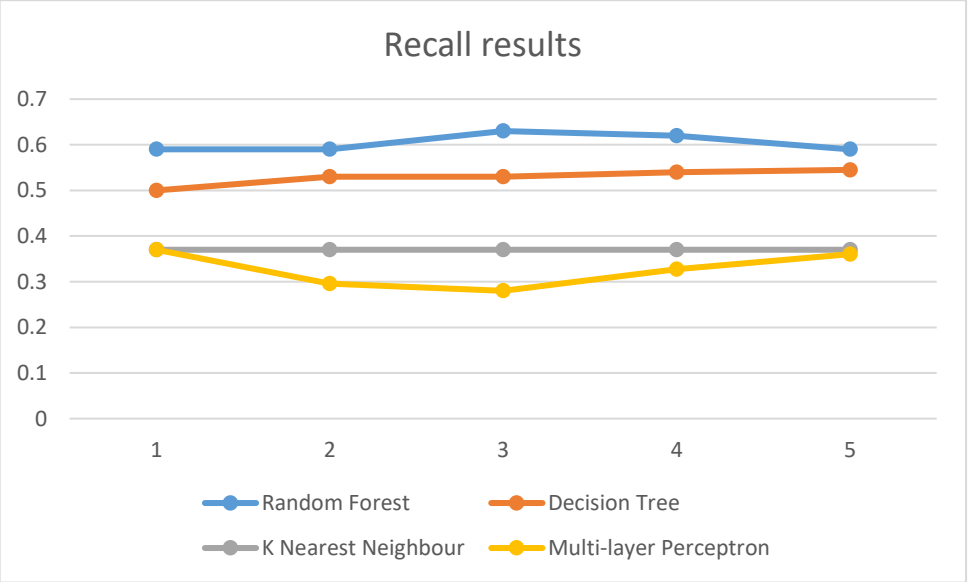


Figure 5. F1 results

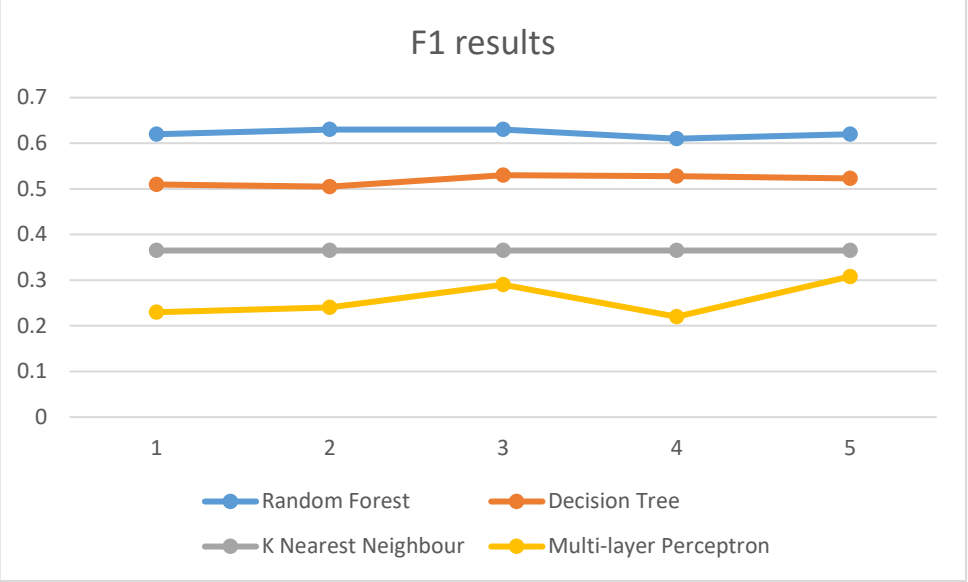


Figure 6. Average accuracy results after 5 tests

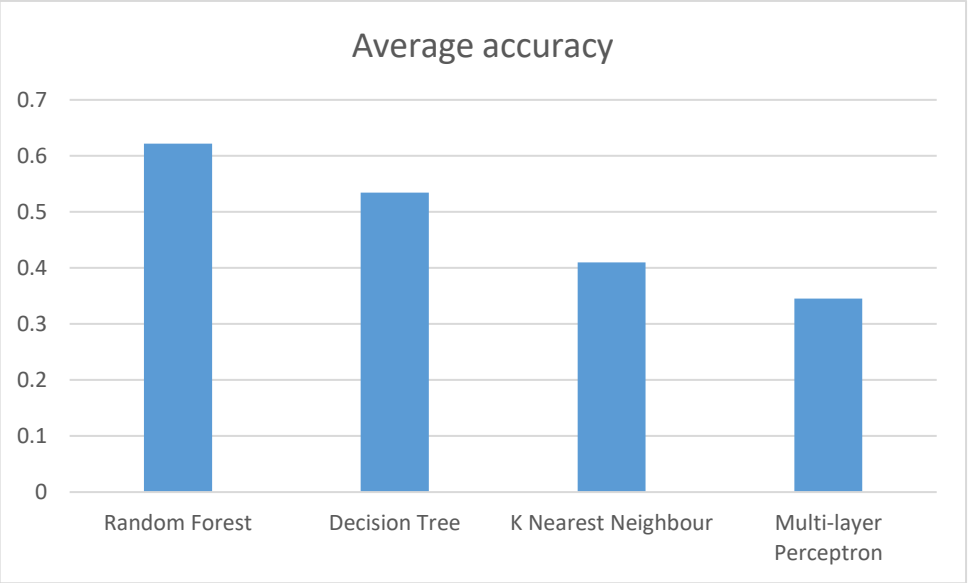


Figure 7. Average precision results after 5 tests

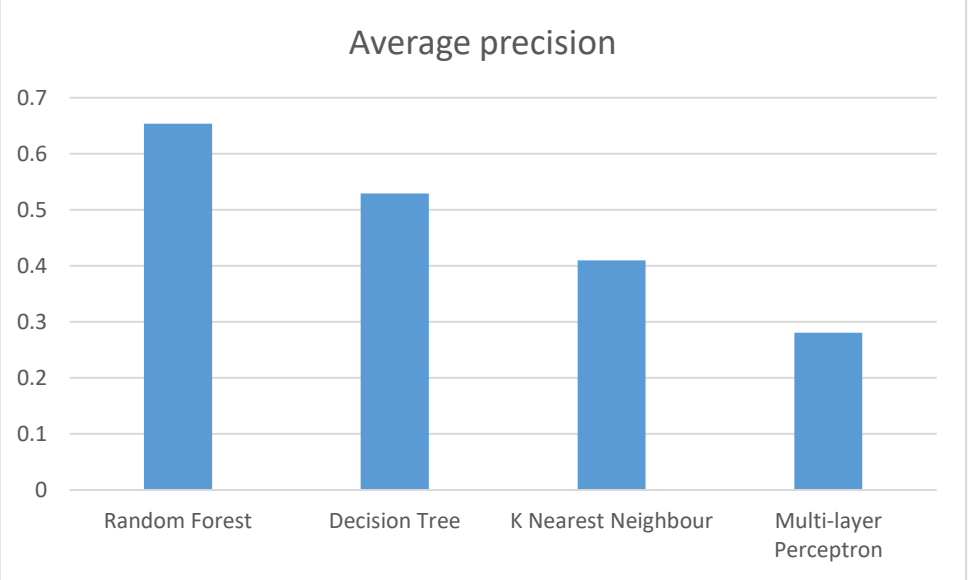


Figure 8. Average recall results after 5 tests

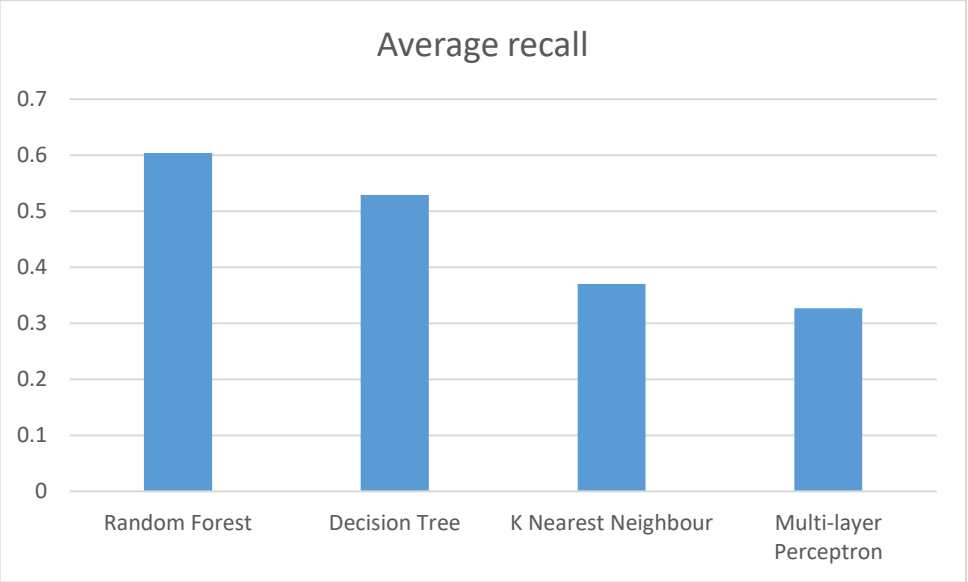


Figure 9. Average F1 results after 5 tests

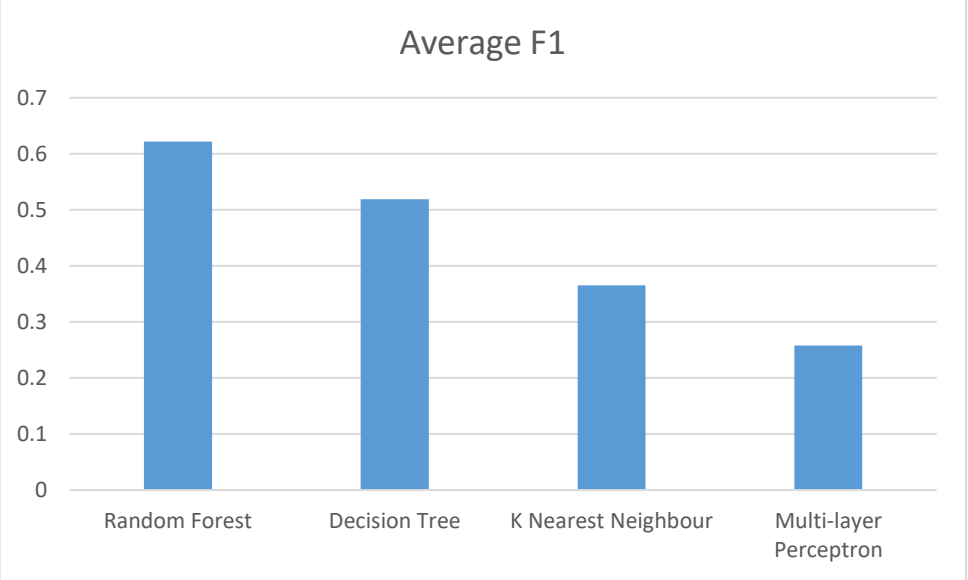
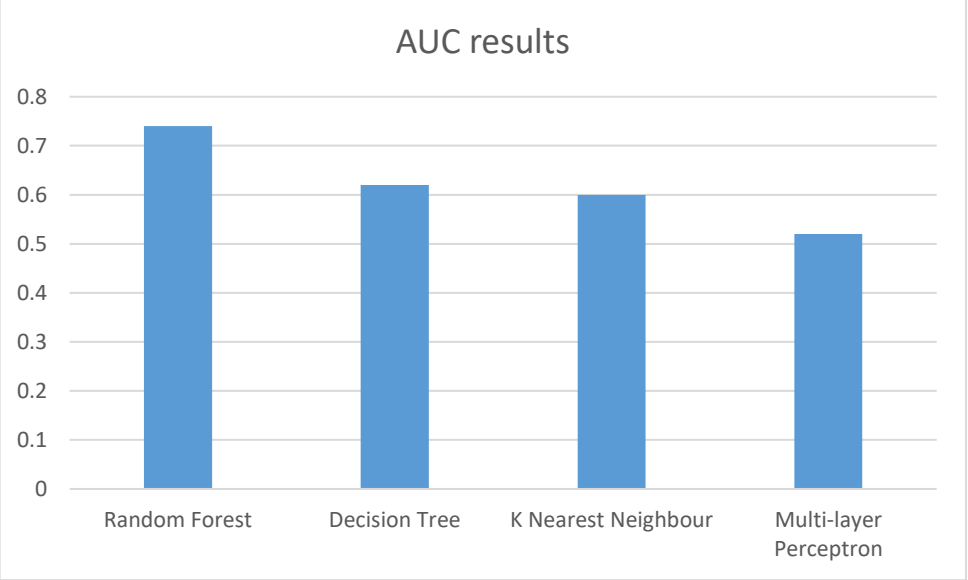


Figure 10. AUC results



The variable max features make significant difference in the output of random forest. The results in table 2 indicate how the algorithm behaves with regards to the use of this variable.

Table 2. Random forest evaluation results.

Metric	Max features = 0.25	Max features = 0.35	Max features = 0.50	Max features = 0.75
Accuracy	64%	62.6%	56.6%	55.3%
Precision	65%	65.9%	59.4%	57.9%
Recall	61%	61%	54.9%	52.1%
F1	63%	61%	56.9%	53.6%
AUC	74%	73.9%	72.3%	71.7%

5 Conclusions

Recommender systems are important in various e-Commerce and other domains to help users and businesses. However, with so many different algorithms it can be time consuming to find which algorithm is better for the data a business has. To this extent we have developed a methodology that helps developers and researchers to identify which recommendation algorithm will perform better if such an algorithm has previously performed well in similar datasets and application domains.

The experimental evaluation results indicate that tree-based algorithms such as the random forest and decision tree perform well compared to other classification approaches. Several metrics have been used and several experiments show the applicability of the methodology. Furthermore, due to the size of the datasets and the values inside the output could vary significantly. However, due to the nature of the data tree-based approaches perform better.

In the future we plan to a) investigate how to improve the accuracy and how machine learning can be used for algorithm selection in other domains such as machine learning and neural networks, b) use SMOTE to generate more synthetic data and c) investigate domains other than collaborative filtering.

References

1. Bobadilla J, Ortega F, Hernando A, Gutiérrez A. Recommender systems survey. Knowledge-based systems. 2013; Volume 46, pp. 109-132.
2. Lu J, Wu D, Mao M, Wang W, Zhang G. Recommender system application developments: a survey. Decision Support Systems. 2015; Volume 74, pp. 12-32.
3. Polatidis N, Georgiadis CK. A multi-level collaborative filtering method that improves recommendations. Expert Systems with Applications. 2016; Volume 48, pp. 100-10.
4. Polatidis N, Georgiadis CK, Pimenidis E, Mouratidis H. Privacy-preserving collaborative recommendations based on random perturbations. Expert Systems with Applications. 2017; Volume 71, pp. 18-25.
5. Abdullah N, Xu Y, Geva S. Integrating collaborative filtering and matching-based search for product recommendations. Journal of theoretical and applied electronic commerce research. 2013; Volume 8(2), pp. 34-48.

6. Konstan JA, Miller BN, Maltz D, Herlocker JL, Gordon LR, Riedl J. GroupLens: applying collaborative filtering to Usenet news. Communications of the ACM. 1997; Volume 40(3), pp. 77-87.
7. Herlocker JL, Konstan JA, Borchers A, Riedl J. An algorithmic framework for performing collaborative filtering. In ACM SIGIR Forum 2017; Volume 51(2), pp. 227-234
8. Bobadilla J, Ortega F, Hernando A. A collaborative filtering similarity measure based on singularities. Information Processing & Management. 2012; Volume 48(2), pp. 204-17.
9. Burke R. Hybrid web recommender systems. InThe adaptive web 2007 (pp. 377-408). Springer, Berlin, Heidelberg.
10. Anand D, Bharadwaj KK. Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities. Expert systems with applications. 2011; Volume 38(5), pp. 5101-9.
11. Liu H, Hu Z, Mian A, Tian H, Zhu X. A new user similarity model to improve the accuracy of collaborative filtering. Knowledge-Based Systems. 2014; Volume 56, pp. 156-66.
12. Son LH. HU-FCF: a hybrid user-based fuzzy collaborative filtering method in recommender systems. Expert Systems with Applications: An International Journal. 201; Volume 41(15), pp. 6861-70.
13. Toledo RY, Mota YC, Martínez L. Correcting noisy ratings in collaborative recommender systems. Knowledge-Based Systems. 2015; Volume 76, pp. 96-108.
14. Wang W, Zhang G, Lu J. Collaborative filtering with entropy-driven user similarity in recommender systems. International Journal of Intelligent Systems. 2015; Volume 30(8), pp. 854-70.
15. Bobadilla J, Alonso S, Hernando A. Deep Learning Architecture for Collaborative Filtering Recommender Systems. Applied Sciences. 2020; Volume 10(7), 2441, pp. 1-14.
16. Nikolakopoulos AN, Berberidis D, Karypis G, Giannakis GB. Personalized diffusions for top-n recommendation. In Proceedings of the 13th ACM Conference on Recommender Systems 2019, pp. 260-268.
17. Pang B, Yang M, Wang C. A novel top-N recommendation approach based on conditional variational auto-encoder. InPacific-Asia Conference on Knowledge Discovery and Data Mining 2019, pp. 357-368.
18. Gazdar A, Hidri L. A new similarity measure for collaborative filtering based recommender systems. Knowledge-Based Systems. 2020; Volume 188, 105058.
19. Patra BK, Launonen R, Ollikainen V, Nandi S. A new similarity measure using Bhattacharyya coefficient for collaborative filtering in sparse data. Knowledge-Based Systems. 2015; Volume 82, pp. 163-77.
20. Wang Y, Deng J, Gao J, Zhang P. A hybrid user similarity model for collaborative filtering. Information Sciences. 2017; Volume 4(18), pp. 102-18.