Cascaded Feature Selection for Enhancing the Performance of Collaborative Recommender System

Mohammad Yahya H. AL-SHAMRI[1,2], Abdulmajid F. AL-JUNIAD[1,2], Talal S. QAID[1,3], Mahdi H. A. AHMED[1,4], and Abeer A. RAWEH[1,3]
[1]*College of Computer Science, King Khalid University, Abha 91413, Saudi Arabia*
[2]*Faculty of Engineering and Architecture, Ibb University, Ibb, Yemen*
[3]*Faculty of Computer Science, Hodeidah University, Al Hudaydah, Yemen*
[4]*College of Engineering and IT, Taiz University, Taiz, Yemen*
mohamad.alshamri@gmail.com

Abstract—Most of collaborative recommender systems (CRSs) rely on statistical and data analysis methods for comparing users. However, dealing with them using machine learning techniques seems to be more appropriate. This paper investigates the usage of feature selection and classification methods for CRSs. It suggests building a user model suitable for the classification purpose and proposes a density-based feature selection (DBFS) method based on the rating density for each class. The DBFS reduces the effect of sparsity problem and keeps only users having a dense-feature history. Additionally, a cascaded feature selection method is proposed to pick out a subset of features through a two-layer approach. The first layer applies a classical feature selection method while the second layer applied the DBFS on the output of the first layer. The results show that the performance is gradually improved. The cascaded feature selection yields the best results since it improves the system accuracy, reduces the space and processing complexities, and alleviates the sparsity in two cascaded layers. The achieved improvements by cascaded feature selection as compared to SVM are 6.55%, 10.14%, and 3.92% in terms of accuracy, F-measure and MAE, respectively.

***Index Terms*—computational modeling, feature extraction, information filtering, machine learning, recommender systems.**

I. INTRODUCTION

The huge amount of online data makes it very difficult for users to find the right information at the right time with a reasonable complexity. This task of teasing the relevant information out of a vast pile of glut looks like finding a needle in a haystack. This makes machine-learning techniques a must for high and ultrahigh dimensional data, which have been emerged in many online applications [1,2]. Usually, the online services have millions if not billions of records what overwhelm the users with a huge number of alternatives that cannot be surfed easily. Here, the power of machine learning comes to explore automatically good predictors based on the past user history. This is what actually required from online applications with very big data like recommender systems. A vivid example of such systems is the Amazon website that suggests many items to its users through personalized webpages [3-6].

In terms of its domains, recommender systems are powerful tools to suggest movies, or music clips for entertainment or to persuade customers to buy more products. Today, the most successful recommender system is the collaborative recommender system (CRS) which can recommend a variety of items spanning music, jokes, movies, books, restaurants or destination locations for tourism. In general, CRS has a set of users U , and a set of rating for a set of items, S . The rating $r_{a,k}$, of the user u_a , for the item s_k , can be binary value indicating like or dislike preference of the user or an integer number within a given interval indicating the level of user satisfaction with that item. There are two more basic types of RSs called content-based and demographic recommender systems [3]. In fact, these systems differ in the profiling approaches and the way of comparing the interests of their users [7]. Sometimes, authors combine some of the basic types to build a hybrid recommender system to get benefits from multiple systems [8].

In fact, knowledge acquisition for RSs requires a learning process. This is usually associated with the training set that is prepared before the implementation stage of each user. Otherwise, the system can do nothing. However, in many cases, the collected data is very small and varies from one user to another. For this reason, the final user-item matrix for CRS is sparse and hence it is difficult to find close neighbors for some users. Moreover, learning on sparse matrix could misguide the learning process that consequently reduces the system accuracy. Many CRSs employ nearest neighbor algorithms for learning from examples. Some other systems used a pre-computed model and they have proved to produce recommendation results that are similar to neighborhood-based recommender techniques [9].

CRSs are heuristic-based models utilizing similarity measures to find a set of neighbors for the active user. Based on the opinion of this set, the system generates predictions which indicate the usefulness of that item for the user. The philosophy is simple; a user heuristically may like what his close neighbors like [3]. This is equivalent to a classification system which classifies the items according to some criteria and then recommends true classified items to the users. Hence, considering CRS as a classification process is very

The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through General Research Project under Grant number (G.R.P-305-38).

Digital Object Identifier 10.4316/AECE.2018.04003

important and may lead to good results. However, due to the constraints and prior requirements of the classification process, many authors avoid classification and directly use statistical methods. Meanwhile, we can go one step farther to improve performance by considering feature selection. Wes will try to reduce the number of features in the dataset by including only useful features in the dataset without changing them. This is usually done by selecting relevant features, that properly describes the problem in hand, and discarding irrelevant and sometimes redundant ones without affecting the system performance which has to be within an acceptable range [10-12].

In fact, the importance of feature selection and classification for large scale data is beyond doubt but the best method does not exist, and hence the researchers either try to find a good method for specific problem settings or try to merge many methods for a hybrid approach [12]. Feature selection can be thought as a good candidate for reducing sparsity of CRSs. By removing irrelevant features, the system can overcome the sparsity problem especially if this process is related to the user history. This encourages us to explore many classification and feature selection methods. Moreover, we propose a novel feature selection method for better performance of sparse applications. Essentially, our aim is to minimize the effect of the sparsity problem, the processing time and the allocated memory while maintaining high accuracy. The main contributions of this paper are:

- Introducing new models suitable for applying classification and feature selection on CRSs.
- Proposing a density-based feature selection method for CRSs to decrease sparsity and enhance the accuracy.
- Implementing many feature selection methods with fixed and with user-dependent percentages.
- Proposing a cascaded feature selection approach for further improvement in recommendation quality.

The rest of this paper is organized as follows: the related work about classification and feature selection for recommender systems is described in Section II whereas Section III discusses in detail how to build appropriate user models for applying classification and feature selection for collaborative recommender systems. Applying classical classification and feature selection methods for CRS is described in Section IV. The proposed density-based feature selection method and cascaded feature selection approach are discussed in Section V. The conducted experiments and analysis of obtained results are presented in Section VI. The last section concludes the paper and gives some directions for future work.

II. Related Work

In literature, classification and feature selection have been widely adopted to explore and mine hidden information in medicine, astronomy, and biology data. A recent survey on the use of machine learning for recommender systems argued that it is not an easy task for researchers and practitioners to assign a specific tool to a given task of the RS. They concluded that Bayesian and decision tree algorithms are widely used in recommender systems because of their popularity and simple implementation [2]. These techniques are used either to build a model for the system or to reduce the system dimensionality. For example, Basu, Harish, and Cohen [13] discussed an inductive learning approach using a combination of collaborative and content features to predict user preferences. This model formalized the recommendation process as a learning problem. Schmidt-Thieme [14] outlined many classification models for CRSs alone but these efforts did not pay off and did not increase the quality. This encouraged other authors to build classification methods by combining collaborative and content-based recommender systems [8,13].

Wang and Tan [15] proposed a naïve Bayesian method for CRSs that ignored the conditional independence assumption. Miyahara and Pazzani [16] converted multiclass data to binary-class data and then applied simple Bayesian classifier on that binary data for both user-based and item-based CRSs. They found that the performance is enhanced compared to the correlation-based CRS. Bouneffouf, Bouzeghoub and Gançarski [17] suggested a Mobile Context-aware Recommender Systems to recommend items for the mobile user based on the user state and interest. They applied a bandit algorithm and case-based reasoning for context recommendation. Saleh, El Desouky and Ali [18] applied many classification techniques for what they called vertical recommendation system. They discussed vertical recommendation system as a four layers RS for suggesting text documents. Actually, they used classification techniques for the content analyzer layer which is the first layer in their system. Wang, Liao, and Zhang [19] used KNN classifier which can adapt to the changes in the user-item matrix but at the cost of re-computing the similarity matrix again. The context features are used by Bouza, Reif, Bernstein, and Gall [20] to build a decision tree model. They considered two ratings for items as the minimum number of ratings to build the decision tree for the user. This approach showed lower precision than recommendation using average rating.

Some authors combined association rules and decision tree in their system. The decision tree is used to select a target user for recommendations whereas the association rule is used to recommend some items [21]. Another research attempt used decision tree for generating a set of recommendations after frequent itemsets are detected using association rules [22]. Hühn and Hüllermeier [23] used a decision tree to rank purposes for the recommendation applications. Su and Khoshgoftaar [24] went one step farther by applying advanced Bayesian networks on multiclass data. Zhang and Iyengar [25] proposed linear classifiers for a model-based recommender system. Gershman et al. [26] used a decision tree for implementing a recommender system that needs only a single traversal.

Clustering is another approach used by some authors to improve the RS performance. Clustering algorithm itself can be improved in terms of performance and speed. For example, Zhang and Ma [27] improved rough k-means clustering based on weighted distance measure with Gaussian function. Borlea, Precup, Dragan, and Borlea proposed centroid update approach to improve k-means algorithm by reducing the number of iterations needed to

perform the clustering process [28]. Chakraborty and Das [29] replaced the conventional Euclidean distance of k-means with the S-distance. They argued that S-K-means preformed better than k-means with Euclidean distance especially when the distribution of the clusters is not regular.

Some authors tried to improve the performance of recommender systems in terms of sparsity and scalability using the clustering approach. Zahra et al. addressed the scalability issues associated with traditional recommender systems by proposing a k-means clustering-based recommendation algorithm which investigates how centroid selection in k-means based recommender systems can improve performance while saving cost [30]. Sharma [31] used an improved k-means clustering algorithm for RS and showed improvement in quality of recommendations and execution time with changing of centroid selection in k-means algorithm. Bobadilla, Bojorque, Esteban, and Hurtado [32] proposed a Bayesian non-negative matrix factorization (BNMF) method to improve the performance of the recommender system by decreasing the sparsity of rating matrix. They argued that BNMF improved the current clustering results in the collaborative filtering area.

III. Building User Models

In recommendation applications, a user profile is normally a set of ratings some of the available items. This set of ratings is usually acquired either implicitly from the user interaction with the system or explicitly through a questionnaire. However, this profile is not useful for classification as it is. First, it has to undergo some prearrangement or sometimes processing to be suitable for classification. In the following subsections, we will discuss how to build the user model for our approaches in this paper.

A. General Model for Classification Problem

For classification problem, we need a set of examples with a set of features and a class. Table I represents a general model for classification problem where E is the set of examples with cardinality (Q) and F is the set of features with cardinality (M). The last column is the class label column.

TABLE I. GENERAL MODEL FOR CLASSIFICATION

		Set of Features (F)					Class
		f_1	f_2	...	f_{M-1}	f_M	$r_{a,k}$
Set of Examples (E)	e_1	5	-	...	1	3	4
	e_2	-	2	...	-	2	1
	⋮	⋮	⋮	...	⋮	⋮	5
	e_{Q-1}	2	2	...	1	-	4
	e_Q	3	-	...	2	4	5

B. User Model for Classification

At first glance when we talk about recommendations, the set of users seems to be the set of examples and the set of items seems to represent the set of features. However, this way has no clear class label and hence it is not useful for classification. The direct way to resolve this issue is to swap users with items so we have a clear class label which is the active user rating for a given item. Hence, the set of examples for a given active user model will be the set of items rated by that active user whereas the set of features will be the set of training users having common ratings with the active user.

$$S_a = \{s_k : s_k \in S, r_{a,k} \neq 0\} \quad (1)$$

$$U_a = \{u_i : u_i \in U, S_{ia} \neq \phi\} \quad (2)$$

Here, S_a is the set of items rated by user u_a with a cardinality K_a whereas U_a is the set of users who have common ratings with user u_a with a cardinality M_a. The set S_{ia} contains the common items between users u_i and u_a. The model size should be fixed for each active user and therefore we fill unknown ratings by 0. This means that an item will be given zero rating if it is not rated yet by the user.

For simple classification, the set of features is $F = U_a$ with cardinality $N = M_a$. However, the set of examples may be less than that of $E = S_a$ with cardinality $Q = K_a$ because some items may be rated only by the active user himself and not by any other user sharing common items with him. This possibility might be rare but we have to consider it especially for odd users with some unique behavior. Hence we will define the set of examples (items) that are rated by both the active user and at least one user sharing some items with him as below:

$$S_{aa} = \{s_k : s_k \in S_{ia}, u_i \in U_a\} \quad (3)$$

Alternatively, we can define S_{aa} as the union of all common ratings sets of all users having common history with the active user.

$$S_{aa} = \bigcup_{i=1}^{M_a} S_{ia} \quad (4)$$

Hence, the set of examples will be $E = S_{aa}$ with cardinality $Q = M_{aa}$.

C. User Model for Feature Selection

Feature selection reduces the problem space and therefore the size of the general model will be reduced according to the set of selected features (users) and the set of items having common ratings with that active user. For this purpose, assume a binary flag for each feature (user) such that:

$$g_i = \begin{cases} True & f_i \text{ is selected} \\ False & f_i \text{ is not selected} \end{cases} \quad (5)$$

Based on that, we can define the set of selected features as:

$$U_a^{'} = \{u_i : u_i \in U_a, g_i = True\} \quad (6)$$

Here, $U_a^{'}$ with cardinality $M_a^{'}$ is the set of features (users) who have common ratings with user u_a and has been selected as a feature for his model. Consequently, the set of examples will be:

$$S_{aa}^{'} = \{s_k : s_k \in S_{ia}, u_i \in U_a^{'}\} \quad (7)$$

where $S_{aa}^{'}$ with cardinality $K_a^{'}$ is the set of items rated by u_a and has been rated also by at least one training user in

his model $U_a^{'}$. In this case, the general model of Table I is used with the following changes, $E = S_{aa}^{'}$, $Q = K_{aa}^{'}$, $F = U_a^{'}$, and $N = M_a^{'}$. Table II summarizes the parameters for the discussed models. We will utilize these models for recommender systems in the following section.

TABLE II. USER MODELS PARAMETERS

Parameter	General Model	Simple Classification Model	Feature Selection Model
Examples	E	S_{aa}	$S_{aa}^{'}$
Features	F	U_a	$U_a^{'}$
Cardinality of E	Q	K_a	$K_a^{'}$
Cardinality of F	M	M_{aa}	$M_a^{'}$

IV. CLASSIFICATION AND FEATURE SELECTION FOR RECOMMENDER SYSTEMS

The previous section discussed how to build user models for classification and feature selection whereas this section discusses in detail how to apply these models for CRS. Our goal is to give an overview about the usefulness of classical classification and feature selection methods for recommender systems.

A. Classification Approach

Classification is one of the easiest learning ways to predict unknown classes based on previous examples. However, classification requires the data to be arranged in a specific manner and this has been done in the previous section. Classification research detailed many classification techniques mentioning their pros and cons. This paper will test four different classification techniques, namely; Naïve Bayes classifier, decision tree-C5.0 classifier, random forest classifier, and support vector machine classifier. The Naïve Bayes classifier is the simplest classifier that utilizes Bayes theorem for conditional probabilities of random variables given known observations to build the classifiers [33,34]. The appealing thing of this classifier is that it is direct, simple and computationally fast to reach a decision. The bad thing of this classifier is that it assumes a specific form of the feature probability distribution for each class.

The second teste classification technique is the decision tree. It represents the data as a tree having nodes as features, edges as values of these features, and leaf nodes as class labels. This paper uses C5.0 decision tree classifier which is an evolution of ID3. Decision tree classifiers perform well for highly relevant features and poor for features with complex relationship [34]. The third classifier is the random forest which is an ensemble predictor close to the nearest neighbor predictor. Actually, ensemble predictors assume that strong predictors can come up from weak ones. Therefore, random forest starts with decision trees with controlled variance as weak predictors and goes ahead by combining them to form an ensemble. This classifier is robust, requires no normalization, and is immune to collinearity [35].

Support vector machine classifier, the last classifier we examined in this paper, is a supervised learning process which uses a non-linear mapping to map the input vectors into some high dimensional feature space Z. This mapping constructs a linear decision surface with certain properties to ensure high generalization ability of the constructed network and to find an optimal hyper plane to clearly separate the sample points of different class labels [36]. However, SVM is slow and faces a challenge in how to get the appropriate kernel for a given dataset [33]

The block diagram in Fig. 1 illustrates how to apply classification for CRS. A model should be built for each active user to generate a set of classifiers which will be used for predicting unknown classes (ratings for our application case).

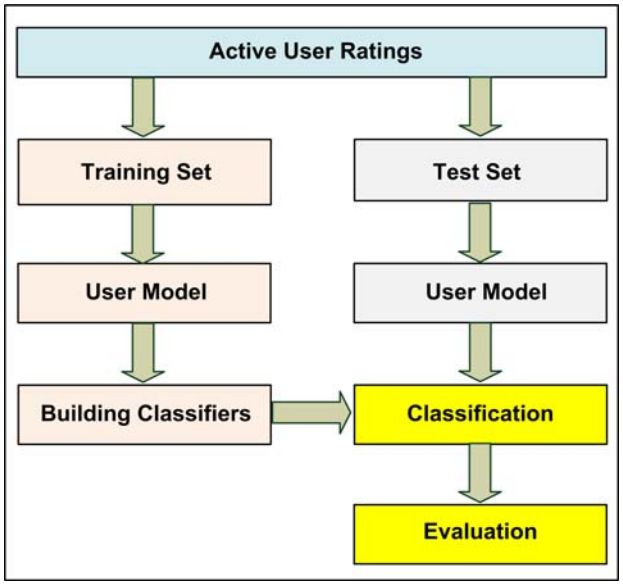

Figure 1. Classification process for CRS.

B. Feature Selection Approach

Usually, feature selection chooses relevant features for predictive modeling problem like recommendation problem. Feature selection has three advantages: first it improves the prediction accuracy of the classifier, second, it gives a faster and more cost-effective classifier, and finally, it shows a clear picture for the features of the underlying problem [37]. This paper utilizes three filtered-type feature selection methods; namely, information gain, correlation-based, and Chi-square [37-39]. The process of this approach is illustrated in Figure 2. The selection process sorts the features according to their relevance and hence selects those ones, within a predefined proportion, which may be fixed for all active users or user-dependent ones.

In recommender systems, each active user has its own identity for dealing with the items in the system. Some users are lenient when rating the items while others are very strict. On another side, some users are very active on the system and try to participate more with the system whilst others have a very limited number of ratings. Moreover, the set of features for each user is different as it relies on the common ratings with the other users. Therefore, giving all users the same selection percentage is not appropriate. The user-dependent ratios seem to be the best choice for this application to reflect the individual properties of each user model. Having agreed on this, a problem arises on how to pick out this value among many. In fact, we have many

variables in terms of the user-item matrix and only one value has to be tuned using a learning technique. The tuned value should reflect the model characteristics. In this problem, we have only a single percentage value to search for with almost limited options. Moreover, small fractions will not affect the results so much. We can go for a genetic algorithm as a learning method but its benefit is low and convergence will be very slow. Moreover, it will take a very long time for the same result we can get with fixed values but with very less processing time. Therefore we will go for stepwise tuning of this value for each user model. The best value is the one that gives the minimum mean absolute error for that user model. Hence our fitness function will be [40]:

$$MAE(u_a) = \frac{1}{\left|S_a^{TE}\right|} \sum_{k=1}^{\left|S_a^{TE}\right|} \left| r_{a,k} - pr_{a,k} \right| \quad (8)$$

where S_a^{TE} is the test ratings for the active user and $pr_{a,k}$ is the predicted rating for item s_k. The learning process will be done offline and hence it will not affect the online processing time. Moreover, we have only one percentage value for each user and hence one array of users' size will be there as an extra storage.

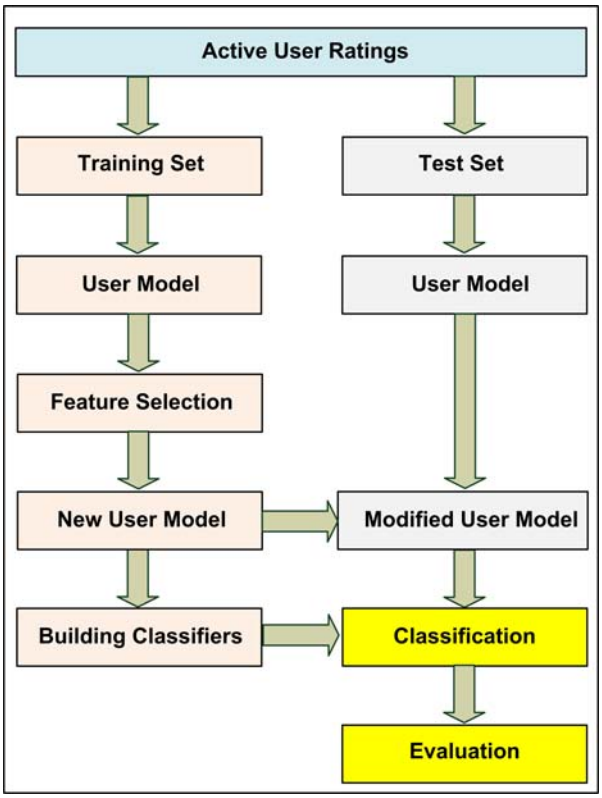

Figure 2. Feature selection process for CRS.

V. Proposed Feature Selection Approaches

This section explores novel methods for applying classification on CRS in an efficient way. Actually, recommender systems suffer from sparsity problem as their users usually rate only a limited number of items. For our application, we have a model for each user and this model is sparse with different degrees of sparsity. Moreover, recommender systems differ from other data applications in that the users mostly rate what they see and like.

In fact, the users of recommender systems are targeted from the beginning and therefore their selection usually is based on some prior information about the items. For this reason, the data is always biased towards high ratings. Another important point here is that the rating values given by the users are not always 100% accurate to reflect each user preference for items and hence they may differ from time to time.

For the mentioned above reasons, the classical feature selection methods may not cope with such applications and also may not be able to identify variable tastes and moods of different users. In the following subsection, we propose a density-based feature selection method which takes into account the user's history of interaction with the system. The result of this approach will be utilized later in building a cascaded feature selection approach. The proposed approaches outperform the previous ones in terms of error performance and accuracy.

A. Density-Based Feature Selection Approach

Many efforts have been devoted to modify the classical feature selection methods. For example, Yu and Liu [11] studied feature redundancy for feature selection. They argued that the focus of most feature selection methods is to find relevant ones. However, this is insufficient especially for high-dimensional datasets. Al-Junaid et al [41] proposed a differential windowed feature selection (DWFS) method for breast cancer identification. However, the proposed approach is specific for cancer, assumes consistent and complete data and it also needs two main datasets for normal and cancer samples.

In terms of sparsity, classical feature selection methods reduce the sparsity as they select only some features from the available ones. This usually happens as a side effect of reducing the number of features because the relevancy of the features is considered but not their sparsity. In this section, we will take the density of the ratings of features as a criterion for selecting features. By this way, we will be sure that the sparsity of the user model is reduced and only those features with high density will be selected. To do so, we have to keep in mind that the user model treats items of the system as examples and users as features. We call the proposed feature selection as density-based feature selection (DBFS) method. The DBFS method suggests parameters at three different levels, model-level, feature-level, and class-level. This feeds the proposed approach with many different parameters and makes it robust in its performance. The following two definitions introduce rating density and density factor for a given user model.

Definition 1 (*rating density*): Rating density of a given feature (user) is simply the set of ratings given by feature u_i, for the model of user u_a. That means the common set of ratings between the two users:

$$rd(u_i) = \left|S_{ia}\right| \quad (9)$$

Definition 2 (*density factor*): Density factor of a given user model is the maximum rating density in that model.

$$d = \max(rd(u_i),...,rd(u_M)) \quad (10)$$

where M is the number of features (users) for this model.

Definition 1 is a feature-level parameter and hence it is a local parameter while definition 2 is a model-level parameter to find the maximum number of ratings between

the active user and any other user and therefore it is a global parameter. The above definitions are not enough as we have to find the same values at the class level for each feature so that we can apply the selection process. For this purpose, the set of ratings S_{ia}^n, given by the feature (user) u_i, for a given class c_n is defined as below:

$$S_{ia}^n = \{s_k : s_k \in S_{ia}, r_{i,k} = c_n\} \quad (11)$$

In this case, S_{ia} will be the union of all S_{ia}^n, where, $n = 1,...,N$. N is the number of classes.

$$S_{ia} = \bigcup_{n=1}^{N} S_{ia}^n \quad (12)$$

Now we can define class density and class density factor as below.

Definition 3 (*class density*): Class density of a given feature (user) is the cardinality of the set of ratings given by the feature for that class:

$$cd_i^n = \left|S_{ia}^n\right| \quad (13)$$

Definition 4 (*class density factor*): Class density factor for a given feature (user) is the maximum class density in the user model for that class:

$$d_a^n = \max\left(cd_i^n,...,cd_M^n\right) \quad (14)$$

We can think of the class density and the class density factor as local and global descriptors for that class. For a given user model, we have many class densities depending on the feature in hand whereas we have only one class density factor for the user model. To complete the selection process, we have to calculate a weight for each feature in the user model before sorting the features and selecting only some of them. In the following, we will define class weights from which we will develop a feature weight.

Definition 5 (*class weight*): Class weight for a given feature is defined as the class density of that feature divided by the class density factor of the user model:

$$cw_i^n = \frac{cd_i^n}{d_a^n} \quad (15)$$

The feature weight is calculated as the weighted mean of all class weights of that feature. Actually, the way of calculating the feature weight from individual class weights can take many forms like simple, weighted or trimmed mean. For weighted mean, we have to obtain the weights of aggregating the class weights. One straightforward way is to use the feature class distribution for this purpose as below:

$$w_i = \frac{\sum_{n=1}^{N} cw_i^n \; cd_i^n}{\sum_{n=1}^{N} cd_i^n} \quad (16)$$

Hence a weighting vector, W, will be formed for all features in the model.

$$W = \langle w_1, w_2,...., w_M \rangle \quad (17)$$

The weighting vector will be used in conjunction with a predefined threshold to select features for the classification process. The set of selected features will vary based on the given threshold; therefore it has to be tuned to get the best result and this usually not easy. The selected set of features for DBFS will be:

$$Set(DBFS) = \{f_{\sigma(i)} : w_{\sigma(i)} \geq THR\} \quad (18)$$

where σ is a permutation that orders features based on their density weights such that $w_{\sigma(1)} \geq w_{\sigma(2)} \geq ... \geq w_{\sigma(M)}$ [41].

The threshold and consequently the selection percentage can be fixed in advance for all users or be user-dependent based on some users' characteristics. For user-dependent percentages, the percentage value is specific for each user and hence may be different for users having the same number of features. In this case, the percentage value for each user has to be learned before the online stage. We have to search for the best percentage selection value based on the active user characteristics. Algorithm 1 depicts the DBFS process which requires a model and a threshold as inputs and returns a set of selected features as an output.

Algorithm 1: DBFS
Input: active user_model, threshold
For each feature
  For each class
    Calculate class density, cd_i^n [equation 13];
  End
End
For each class
  Calculate the class density factor, d_a^n [equation 14];
End
For each feature
  For each class
    Calculate class weight, cw_i^n [equation 15];
  End
  Calculate feature weight, w_i [equation 16];
End
For each feature
  If $w_i > THR$ // feature weight is greater than threshold
    Add the feature to *Set(DBFS)*;
  End
End

B. Cascaded Feature Selection Approach

This approach goes one step further than simple feature selection by applying the density-based feature selection on the newly generated matrix. Hence, we will exploit the benefits of both the classical feature selection and the density-based feature selection. The first stage applies the classical feature selection whereas the second stage applies the density-based feature selection on the selected set of features. Therefore we will keep the most relevant features from the first stage and then refine them according to their values for the rating density. That means we will apply the density-based feature selection on more trustable user model than before.

Moreover, this will reduce the effect of sparsity at two different levels. The first level will be achieved using the classical feature selection by removing irrelevant features whereas the second level will be achieved using the density-based feature selection by removing less dense features. The block diagram in Fig. 3 illustrates the process of the cascaded feature selection process. For this approach, two percentage values will be required and hence they have to be learned offline.

Algorithm 2 depicts the cascaded approach which needs a model and a threshold as inputs and returns a set of selected features as an output. First, the process come up with a set of features based on correlation-based feature selection and then refine this set using DBFS.

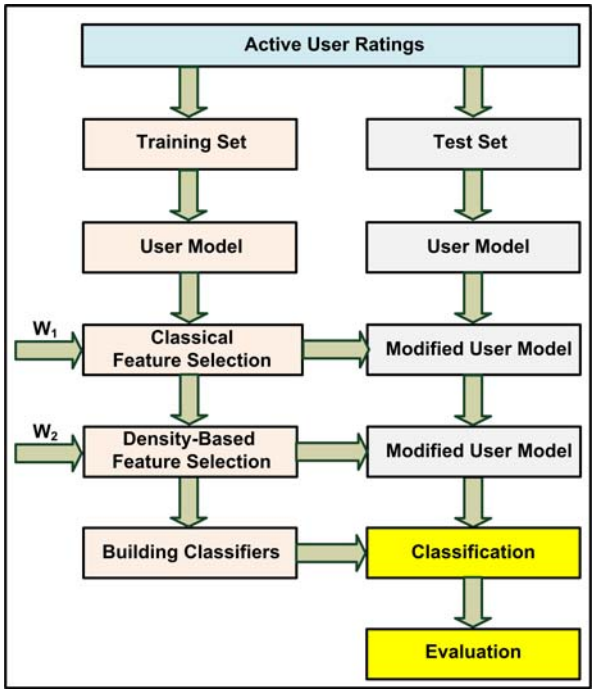

Figure 3. Cascaded feature selection process.

Algorithm 2: Cascaded_CBFS_DBFS
Input: training_users, training_ratings, test_ratings, threshold-1, threshold-2
For each active user
First Stage:
training_model ← **Build_Model**(training_users, training_ratings);
test model ← **Build_Model**(training_users, test_ratings);
For each feature of the training_model
Calculate the feature weight based on CBFS method
If feature weight is greater than threshold-1
Add the feature to $Set(CBFS)$;
End
End
modified_training_model ← **Modify_Model**(training_model, $Set(CBFS)$);
modified_test_model ← **Modify_Model**(test_model, $Set(CBFS)$);
Second Stage:
For each feature of the modified_training_model
Calculate the feature weight based on DBFS method
If feature weight is greater than threshold-2
Add the feature to $Set(DBFS)$;
End
End
modified_training_model ← **Modify_Model**(training_model, $Set(DBFS)$);
modified_test_model ← **Modify_Model**(test_model, $Set(DBFS)$);
classifiers ← **SVM**(modified_training_model);
predictions ← **classify**(modified_test_model, classifiers);
End

VI. EXPERIMENTS

For our experiments, R language and Caret R package are used as an implementation environment. The experiments cover all approaches we have discussed before with different variants for each approach if possible. The following subsections discuss in detail different aspects of experiments implementation like dataset preparation, evaluation metrics, the conducted experiments, and finally the analysis and discussion of the results.

A. Data Preparation

MovieLens dataset is the most popular dataset for recommender systems [www.movielens.net] which has many packages. For our experiments, we used 1M MovieLens dataset which has one million plus ratings for 3952 movies given by 6040 users. Each rating takes a discrete value between 1 and 5. The number of ratings given by each user varies from very small number 20 to a very big number 2314.

To conduct the classification process, we chose the number of ratings to be 250 or more for two reasons: (1) to get a dense classification model reflecting a rich history of the user; (2) to reduce the number of zeros (un-rated items) in the user model which may affect the classification process. Among 6040 users, we found 1225 users, satisfying the above-mentioned condition, from which we selected randomly 50 users for testing and called them active users. The remaining 1175 are considered as training users. The active user ratings will be used to build a model and to test the system. Again, the active user ratings set S_a, has to be divided randomly into training ratings, S_a^{TR} (80%), and testing ratings, S_a^{TE} (20%). During experimentation, the active user is assumed to have only the training ratings for modeling. The test ratings are used to evaluate the system since we assume them as unknown ratings and search for predicted ratings for them.

Fig. 4 illustrates how the dataset is divided in terms of users and ratings of the active user. The training users in conjunction with the training ratings of the active user are used to build model for the classifier. However, the training users in conjunction with the test ratings of the active user are used to build a model for predictions.

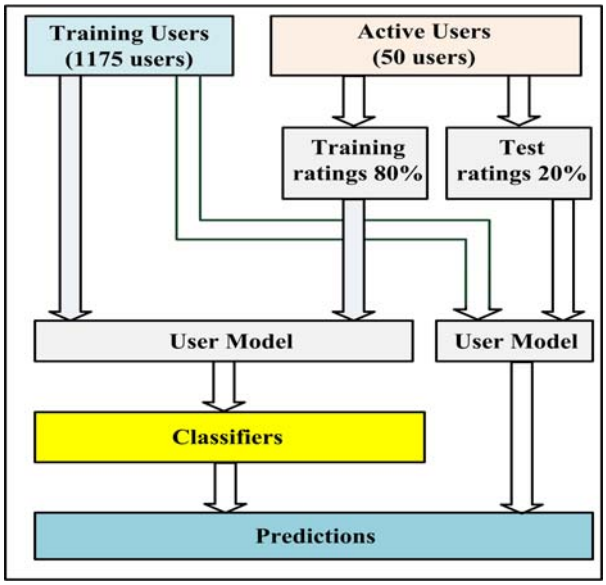

Figure 4. Dataset division in terms of users and ratings.

B. Evaluation Metrics

Four evaluation metrics are used for evaluating the prediction accuracy and error of our experiments, namely, mean absolute error, root mean square error, F-measure, and accuracy. Different measures will help us to assess tested approaches from many aspects. The first error measure is the absolute mean error (MAE) which is defined as the absolute difference between the actual and predicted ratings [40,42]. Low MAE means that the RS predictions are more accurately. The MAE over the test ratings of the active user u_a, is:

$$\mathrm{MAE}(u_a) = \frac{1}{\left|S_a^{TE}\right|} \sum_{j=1}^{S_a^{TE}} \left| r_{a,j} - pr_{a,j} \right| \qquad (19)$$

The net value of MAE over all active users is the weighted average of individual values of all active users.

$$\text{MAE} = \sum_{a=1}^{M_A} w_a \ \text{MAE}(u_a) \tag{20}$$

The weight is calculated based on the number of test ratings for that active user relative to the total number of test ratings in the system.

$$w_a = \frac{\left|S_a^{TE}\right|}{\sum_{i=1}^{M_A}\left|S_i^{TE}\right|} \tag{21}$$

Another important error measure is the RMSE which increases the contribution of high errors by squaring their values and this can be considered as error penalty [40,42]. In fact, large value of RMSE indicates that the predicted value is far away from the actual rating whereas small value indicates the inverse conclusion. The RMSE for all active users (M_A) will be:

$$\text{RMSE} = \sum_{a=1}^{M_A} w_a \left(\sqrt{\frac{1}{\left|S_a^{TE}\right|} \sum_{j=1}^{S_a^{TE}} \left(r_{a,j} - pr_{a,j}\right)^2} \right) \tag{22}$$

The other two prediction accuracy measures, namely accuracy and F-measure will be calculated directly from the confusion matrix [43,44]. For a multi-class recommender system, the classification process will take many classes. Hence the confusion matrix will be as in Table III for 5-grade rating scale.

TABLE III. CONFUSION MATRIX OF A 5-CLASS CLASSIFICATION

		Predicted ratings				
		pr_1	pr_2	pr_3	pr_4	pr_5
Actual ratings	r_1	t_{11}	f_{12}	f_{13}	f_{14}	f_{15}
	r_2	f_{21}	t_{22}	f_{23}	f_{24}	f_{25}
	r_3	f_{31}	f_{32}	t_{33}	f_{34}	f_{35}
	r_4	f_{41}	f_{42}	f_{43}	t_{44}	f_{45}
	r_5	f_{51}	f_{52}	f_{53}	f_{54}	t_{55}

The accuracy for N-class recommender system is:

$$\text{Accuracy}(u_a) = \frac{\sum_{i=1}^{N} t_{ii}}{\sum_{i=1}^{N} t_{ii} + \sum_{i=1}^{N}\sum_{j=1, j\neq i}^{N} f_{ij}} \tag{23}$$

The total value of accuracy for all active users is the weighted average of each active user value.

$$\text{Accuracy} = \sum_{a=1}^{M_A} w_a \ \text{Accuracy}(u_a) \tag{24}$$

To calculate recall, precision, and F-measure, we need to know the sets of correct and predicted ratings for a given user u_a , which are defined as [7]:

$$\text{CorrectSet}(u_a) = \left\{ s_k \middle| s_k \in S_a^{TE}, pr_{a,k} = r_{a,k} \right\} \tag{25}$$

$$\text{PredictedSet}(u_a) = \left\{ s_k \middle| s_k \in S_a^{TE}, pr_{a,k} \neq 0 \right\} \tag{26}$$

In general, precision indicates the number of selected items which are relevant to the user, however recall indicates the number of relevant items which are selected. In recommendation applications, relevant item is the one that was predicted correctly by the system. Therefore, we can define the precision and recall for prediction-based recommender system as:

$$\text{precision}(u_a) = \frac{\left|\text{CorrectSet}(u_a) \cap \text{PredictedSet}(u_a)\right|}{\left|\text{PredictedSet}(u_a)\right|} \tag{27}$$

$$\text{recall}(u_a) = \frac{\left|\text{CorrectSet}(u_a) \cap \text{PredictedSet}(u_a)\right|}{\left|\text{CorrectSet}(u_a)\right|} \tag{28}$$

We will follow the WEKA approach for calculating the individual and average values of precision, recall, and F-measure for multiclass classification. The individual value of recall for the n^{th} class is:

$$R_n = \frac{t_{nn}}{t_{nn} + \sum_{j=1, j\neq n}^{N} f_{nj}} \tag{29}$$

Similarly, the individual value of precision for the n^{th} class is:

$$P_n = \frac{t_{nn}}{t_{nn} + \sum_{i=1, i\neq n}^{N} f_{in}} \tag{30}$$

We mean by individual values a specific rating value in the rating scale. Actually, the system has N classes based on the N values of the rating scale. Therefore we have to aggregate them to find the final value. The individual value of the F-measure for the n^{th} class is:

$$F_n = \frac{2 R_n P_n}{R_n + P_n} \tag{31}$$

The weight for the i^{th} class value will be:

$$w_i = \frac{t_{ii} + \sum_{j=1, j\neq i}^{N} f_{ij}}{\sum_{i=1}^{N}\left(t_{ii} + \sum_{j=1, j\neq i}^{N} f_{ij}\right)} \tag{32}$$

Accordingly, the net value of each metric is the weighted average of the individual class values. Therefore the average F-measure of the active user u_a , is:

$$F(u_a) = \sum_{i=1}^{N} w_i \ F_i \tag{33}$$

The overall value of F-measure for all active users is the weighted average of each active user value.

$$\text{F} = \sum_{a=1}^{M_A} w_a \ \text{F}(u_a) \tag{34}$$

C. Experiments

We conducted six sets of experiments, one set for each approach. The experiment set of some approaches consists of many experiments based on the employed methods for that approach. There are some predefined parameters that will be discussed inside each experiment. Table IV lists the approaches and the corresponding list of experiment(s) associated with them.

One important parameter in our experiments is the selection percentage which may be fixed or user-dependent. For the latter case, a stepwise tuning between 5% and 50% in a step of 5% each time is used. Since we have 10 options for each value, then 100 options will be there for each case. In total, we have 5000 options for our test dataset which is very small compared to genetic algorithm approach with a population size of 10, maximum generations of 30 and runs

of 15. We restricted our tests to 50% only to be consistent with the main aim of the feature selection to filter out irrelevant features. We think that 50% of the features is more than enough for this purpose.

D. Analysis and Discussion

The results show that a good improvement is achieved by applying the proposed approaches especially those having user-dependent selection percentages. This improvement depends on the employed method and the applied approach. In the following, we will discuss in detail each one of them.

TABLE IV. LIST OF EXPERIMENTS FOR ALL APPROACHES

Approach	Experiment(s)
Approach-1	Naive Bayes Classifier
	Decision tree- C5.0
	Random Forest
	SVM
Approach-2	Information Gain (10%, 20%, 30%)
	Correlation-based (10%, 20%, 30%)
	Chi square (10%, 20%, 30%)
Approach-3	Correlation-based feature selection with user-dependent selection percentages
Approach-4	Density-based feature selection (10%, 15%, 20%, 25%, 30%, 35%, 40%)
Approach-5	Density-based feature selection with user-dependent selection percentages
Approach-6	Correlation-based feature selection with user-dependent selection percentages followed by density-based feature selection with user-dependent selection percentages

Approach-1 (Simple Classification)

The results of Naïve Bayes classifier, decision tree-C5.0 classifier, random forest classifier, and support vector machine classifier are listed in Table V.

TABLE V. RESULTS OF APPROACH-1

Classification Method	Accuracy	F-Measure	MAE	RMSE
Naive Bayes Classifier	0.358741	0.323847	0.930429	1.240380
Decision tree - C5.0	0.367258	0.368733	0.908682	1.241110
Random Forest	0.466252	0.402379	0.609317	0.869115
SVM	**0.487030**	**0.433479**	**0.590259**	**0.862162**

The results indicate that the best classification method is SVM in all aspects whereas Naïve Bayes is the worst one. There are big differences between the best and worst method in all metrics. This agrees with the findings of Su and Khoshgoftaar [24], where the simple Bayesian classifier showed worse predictive accuracy than traditional CRS. Hereafter, we will use SVM method for all classification purposes.

Approach-2 (Feature Selection – Fixed-percentage)

The results of information gain, correlation-based, and Chi-square feature selection with fixed selection percentages for all users are listed in Table VI. The feature selection is always followed by SVM classification in our experiments. For the three selection percentages 10%, 20%, and 30%, all results are less than that of the simple SVM classification approach except the results of correlation-based feature selection with 20% selection percentage, which performs better than the simple SVM classification and the other feature selection methods. These results are better than that of the simple SVM classification method with only 20% of the features. This is a good achievement for reducing the processing time, storage requirements, and sparsity.

TABLE VI. RESULTS OF APPROACH-2

		Information Gain	Correlation-based	Chi-square
Accuracy	10%	0.4781631	0.4751191	0.4768396
	20%	0.4778984	**0.4874272**	0.4789571
	30%	0.4817364	0.4833245	0.4817363
F-Measure	10%	0.4256797	0.4337240	0.4246826
	20%	0.4253316	**0.4425498**	0.4263906
	30%	0.4280072	0.4379344	0.4278189
MAE	10%	0.6124934	0.6135521	0.6128904
	20%	0.6109052	**0.5954209**	0.6095818
	30%	0.6017734	0.5983325	0.6019058
RMSE	10%	0.8927831	0.8901927	0.89131
	20%	0.887912	**0.8723921**	0.8868972
	30%	0.4781631	0.4751191	0.4768396

Approach-3 (Feature Selection – User-dependent percentages)

Approach-2 gives good results with one fixed selection percentage, 20% for all active users in the system. However, this may be not suitable for all users as the rating taste for them is different. Hence, it will be an advantage if we assign a user-dependent selection percentage for each user. In this experiment, we will search for this value between 5% and 50% with a step size of 5%. Accordingly, we will test ten percentage values for each active user. The results of this approach are listed in Table VII. Note that the table summarizes the best results of all approaches examined in this paper.

TABLE VII. RESULTS OF APPROACH-3

Approach	Accuracy	F-Measure	MAE	RMSE
Approach-1	0.4870302	0.4334795	0.5902594	0.8621621
Approach-2	0.4874272	0.4425498	0.5954209	0.8723921
Approach-3	0.5051615	0.4623655	0.5759661	0.8582970
Approach-4	0.4928534	0.4475190	0.5872155	0.8643010
Approach-5	0.5088671	0.4645541	0.5733192	0.8581199
Approach-6	**0.5189254**	**0.4774342**	**0.5670990**	**0.8605500**

The results of this approach are better than the best results of the fixed-percentage selection listed also in the same table. The accuracy jumps from 48.7% to 50.5%. The improvement percentages are 3.72% and 6.66% in terms of accuracy and F-measure as shown in Table VIII. We follow Al-Shamri [7] for calculating the improvement percentages. This indicates that the set of features selected for each user model reflects the user mood for the rating scale. Hence, the system needs only a small set of features to ensure more accurate classifications. The results in Table VIII illustrate the improvement percentages for all the approaches compared to the best simple classification method, i.e. SVM.

TABLE VIII. IMPROVEMENT PERCENTAGES OF DIFFERENT APPROACHES COMPARED TO APPROACH-1

Approach	Accuracy	F-Measure	MAE	RMSE
Approach-2	0.081514	2.09244	-0.874446	-1.186552
Approach-3	3.722829	6.663752	2.42153	0.448300
Approach-4	1.195655	3.238792	0.51569	-0.248086
Approach-5	4.483685	7.168643	2.86996	0.468840
Approach-6	**6.548916**	**10.13997**	**3.92377**	**0.186980**

Approach-4 (Density-based Feature Selection - Fixed-percentage)

Density-based feature selection is appropriate for sparse matrices like what we have in recommender systems. In our experiments, we test seven percentage values, 10%, 15%, 20%, 25%, 30%, 35%, and 40%. The results of this approach are listed in Table IX. We found that the best selection percentage is 25%. The results are better than those of approaches 1 and 2. This indicates the appropriateness of DBFS method for recommender systems as sparse applications. Moreover, this selection method always performs better than the classical approaches for all percentage values. The improvement percentages compared to simple SVM classification are 1.2% and 3.24% in terms of accuracy and F-measure with only 25% of the features. By increasing the selection percentage beyond 25%, the results get worse.

TABLE IX. RESULTS OF APPROACH-4

Selection Percentage	Accuracy	F-Measure	MAE	RMSE
10%	0.4842509	0.4419685	0.6019058	0.8816893
15%	0.4862361	0.4446160	0.5974060	0.8754364
20%	0.4883536	0.4459744	0.5910535	0.8663363
25%	**0.4928534**	**0.4475190**	**0.5872155**	**0.8643010**
30%	0.4874272	0.4430250	0.5911858	0.8649675
35%	0.4886183	0.4436868	0.5898624	0.8630716
40%	0.4899418	0.4430058	0.5890683	0.8637018

Approach-5 (Density-based Feature Selection – User-dependent percentages)

The difference between this approach and the previous one lays on how to choose the selection percentages. Here we have to search for the best selection percentage for each user model while considering the individual user identity. The results of this approach are better than the corresponding fixed one by good margin for all measures. The accuracy is now 50.89% instead of 49.29%, whereas the MAE error is 0.5733192 instead of 0.5872155. This indicates better performance of this approach compared to the fixed one. Moreover, this approach reflects the internal characteristics of different users.

Approach-6 (Cascaded Feature Selection)

The results of this approach are the best among all other approaches as shown in Table VII. The achieved accuracy is now 51.89%, while F-measure is 47.74%. The improvements as compared to simple classification are 6.55%, 10.14%, and 3.92% in terms of accuracy, F-measure and MAE respectively. These improvements show the effectiveness of the proposed approach especially in terms of F-measure which indicates that the generated recommendations will be more relevant to users.

The selection percentage values W_1 and W_2 for both stages are trained offline. Some users need a light filtering whereas some others need a heavy filtering. For example, u_{29} keeps high percentage values whereas u_{19} keeps very low value for both stages. Other users select two different values in the two stages. For example, u_{50} keeps 50% of the features in the first stage and only 10% of them in the second stage. Table X gives more examples of the learned selection percentage values. Fig. 5 illustrates the final and best results of each approach as listed also in Table VII. The results show that simple classification has the lowest accuracy whereas cascaded feature selection (approach 6) has the best results.

The accuracy values of both simple classification and correlation-based feature selection are very close. However, F-measure of the correlation-based feature selection is better than that of simple classification. This means that the system is now able to generate more relevant items. The final and best results of each approach in terms of MAE and RMSE are drawn in Fig. 6. The results are very close with little advantage of approach 5 and approach 6. The results of RMSE are higher than that of MAE because of the error squaring.

TABLE X. SOME WEIGHTING VALUES FOR SOME ACTIVE USERS

Active user	W_1	W_2
u_4	0.15	0.45
u_{14}	0.05	0.35
u_{15}	0.1	0.15
u_{19}	0.05	0.05
u_{21}	0.5	0.25
u_{29}	0.4	0.4
u_{50}	0.5	0.1

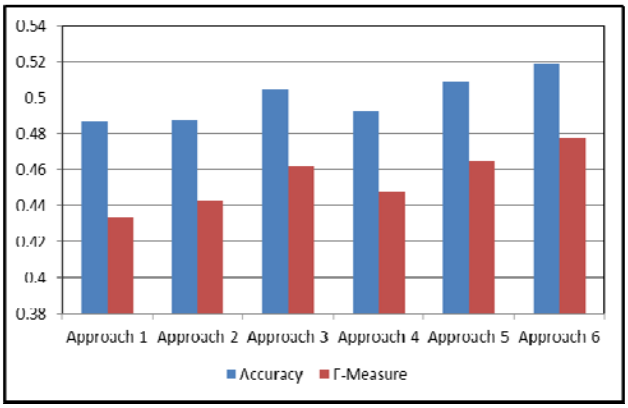

Figure 5. Accuracy and F-measure values for all approaches.

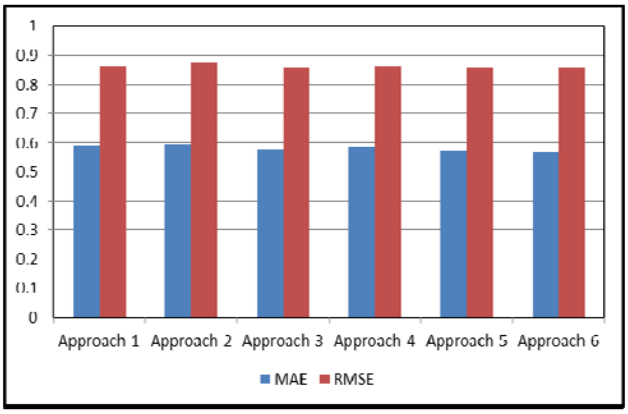

Figure 6. MAE and RMSE values for all approaches.

Fig. 7 summarizes the improvements of all approaches compared to approach-1 which is the simplest classification. The improvements are high for F-measure which indicate that the generated predictions are more relevant to the active user and hence more interesting. Accuracy improvements

are also high especially for approach-6. In fact, there were improvements for all measures. There is one exception about RMSE and MAE for approach-2. The results of error performance of approach-2 are less than that of approach-1. This approach chose only a subset of features based on correlation-based feature selection method. However, this story is changed using the proposed density-based feature selection method which shows good error improvement.

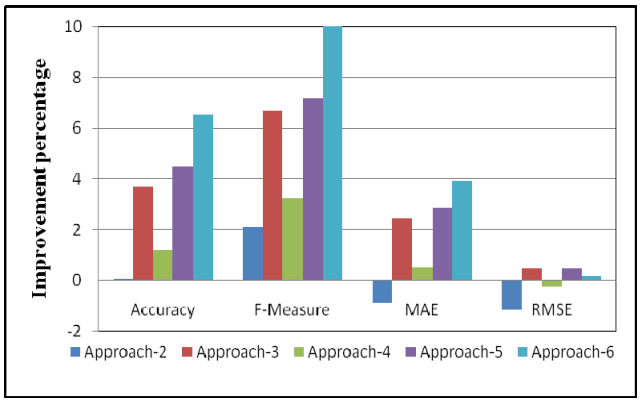

Figure 7. Improvement of all approaches compared to approach-1.

VII. Conclusions

Feature selection is a powerful artificial intelligence method for processing huge datasets like recommender systems. This paper builds a roadmap with clear examples of using classical feature selection and classification for recommender systems.

In this paper, we introduced user models for applying different approaches and then we tested them experimentally. The results showed that feature selection methods indeed improve the performance of the recommender system.

Moreover, the results showed that the suggested density-based feature selection method solves to a good extent the sparsity problem by selecting only dense features. The density-based feature selection gives better results than that of the classical ones especially when we tune the selection percentage based on user characteristics. The tuned percentage values for both classical feature selection and density-based feature selection reflect the users' preference more accurately and help the system to reduce the effect of the sparsity problem.

In addition, the proposed cascaded feature selection takes the results one step farther, since it is fast in terms of processing and light in terms of storage requirements. The heavy job is done offline and hence it will be appropriate for online applications.

The proposed approaches try to alleviate the sparsity problem to some extent. However, many ideas can be introduced in this direction like matrix decomposition, filling the missed values, or extracting useful features from the original ones. This will be left for future work in order to explore the possibility of classical machine learning methods to enhance the performance of the recommender system.

References

[1] D. Donoho, "High-dimensional data analysis: the curses and blessings of dimensionality," in Proc. American Mathematical Society Conference of Math Challenges of the 21st Century, 2000.

[2] I. Portugal, P. Alencar, and D. Cowan, "The use of machine learning algorithms in recommender systems: a systematic review," *arXiv*, vol. 4, pp. 1–16, Nov. 2015. doi:1511.05263v4

[3] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," Knowledge-Based Systems, vol. 46, pp.109–132, 2013. doi:10.1016/j.knosys.2013.03.012

[4] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative filtering recommender systems," Foundations and Trends in Human–Computer Interaction, vol. 4, No. 2, pp. 81–173 2010. doi:10.1561/1100000009

[5] F. Isinkaye, Y. Folajimi, and B.Ojokoh, "Recommendation systems: Principles, methods and evaluation," Egyptian Informatics Journal, vol. 16, pp. 261-273, 2015. doi:10.1016/j.eij.2015.06.005

[6] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," Advances in Artificial Intelligence, vol. 2009, pp. 1-9, 2009. doi:10.1155/2009/421425

[7] M. Y. H. Al-Shamri, "User profiling approaches for demographic recommender systems," Knowledge-Based Systems, vol. 100, pp. 175-187, 2016. doi:10.1016/j.knosys.2016.03.006

[8] R. Burke, "Hybrid recommender systems: Survey and experiments," User Modeling and User-Adapted Interaction, vol. 12, No. 4, pp. 331–370, 2002. doi:10.1023/A:1021240730564

[9] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in Proc. 14th conference on Uncertainty in artificial intelligence, 1998, pp. 43-52. doi:1301.7363

[10] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, Feature Extraction: Foundations and Applications. Springer, pp. 463-470, 2006. doi:10.1007/978-3-540-35488-8

[11] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy", Journal of Machine Learning Research, vol. 5, pp. 1205–1224, 2004.

[12] V. B. Canedo, Novel feature selection methods for high dimensional data. PhD thesis, March 2014.

[13] C. Basu, H. Harish, W. Cohen, "Recommendation as classification: Using social and content-based information in recommendation," AAAI Technical Report, WS-98-08, 1998.

[14] L. Schmidt-Thieme, "Compound classification models for recommender systems," in Proc. 5th IEEE International Conference on Data Mining, 2005, pp. 378-385. doi:10.1109/ICDM.2005.46

[15] K. Wang, Y. Tan, "A new collaborative filtering recommendation approach based on naive Bayesian method," in: Tan Y., Shi Y., Chai Y., Wang G. (eds) Advances in Swarm Intelligence. ICSI 2011. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, vol. 6729, pp. 218–227, 2011. doi:10.1007/978-3-642-25832-9

[16] K. Miyahara and M. J. Pazzani, "Improvement of collaborative filtering with the simple Bayesian classifier," IPSJ journal, vol. 43, no. 11, pp. 3429-3437, 2002.

[17] D. Bouneffouf, A. Bouzeghoub, and A. L. Gançarski, "Hybrid- ε-greedy for mobile context-aware recommender system," in Proc. Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer Berlin Heidelberg, 2012, pp. 468-479. doi:10.1007/978-3-642-30217-6_39

[18] A. I. Saleh, A. I. El Desouky, and S. H. Ali, "Promoting the performance of vertical recommendation systems by applying new classification techniques," Knowledge-Based Systems, vol. 75, pp. 192–223, 2015. doi:10.1016/j.knosys.2014.12.002

[19] B. Wang, Q. Liao, and C. Zhang, "Weight based KNN recommender system," in Proc. 5th International Conference on Intelligent Human–Machine Systems and Cybernetics (IHMSC), 26–27 August 2013, pp. 449–452. doi:10.1109/IHMSC.2013.254

[20] A. Bouza, G. Reif, A. Bernstein, and H. Gall, "SemTree: ontology-based decision tree algorithm for recommender systems," in Proc. *Semantic Web Conference*, Karlsruhe, Germany, 2008.

[21] Y. Cho, J. Kim, and S. Kim, "A personalized recommender system based on web usage mining and decision tree induction," Expert Systems with Application, vol. 23, No, 2, pp. 329–342, 2002. doi:10.1016/S0957-4174(02)00052-0

[22] D. Nikovski and V. Kulev, "Induction of compact decision trees for personalized recommendation," in Proc. ACM Symposium on Applied Computing, Dijon, France, April 2006, pp. 575–581. doi:10.1145/1141277.1141410

[23] W. Cheng, J. Hühn, and E. Hüllermeier. "Decision tree and instance-based learning for label ranking," in Proc. 26th Annual International

Conference on Machine Learning (ICML '09), New York, NY, USA, 2009, pp. 161–168. doi:10.1145/1553374.1553395

[24] X. Su and T. M. Khoshgoftaar, "Collaborative filtering for multi-class data using belief nets algorithms," International Journal on Artificial Intelligence Tools, vol. 17, No. 1, pp. 71-85, 2008. doi:10.1142/S0218213008003789

[25] T. Zhang and V. S. Iyengar, "Recommender systems using linear classifiers," Journal of Machine Learning Research, vol. 2, pp. 313-334, 2002.

[26] A Gershman, A. Meisels, K-H. Luke, L. Rokach, A. Schclar, and A. Sturm, "A decision tree based recommender system," in Proc. 10th Conf. on Innovative Internet Community Services, Trondheim, Norway, 2010, pp. 170-179.

[27] T. Zhang and F. Ma, "Improved rough k-means clustering algorithm based on weighted distance measure with Gaussian function", *International Journal of Computer Mathematics*, vol. 94, no. 4, pp. 663-675, 2017. doi: 10.1080/00207160.2015.1124099.

[28] I. D. Borlea, R. E. Precup, F. Dragan and A. B. Borlea, "Centroid update approach to k-means clustering", *Advances in Electrical and Computer Engineering*, vol. 17, no. 4, pp. 3-10, 2017. Doi: 10.4316/AECE.2017.04001.

[29] S. Chakraborty and S. Das, "k-means clustering with a new divergence-based distance metric: Convergence and performance analysis", *Pattern Recognition Letters*, vol. 100, pp. 67-73, 2017. doi: 10.1016/j.patrec.2017.09.025.

[30] S. Zahra, M. A. Ghazanfar, A. Khalid, M. A. Azam, U. Naeem, A. P. Bennett, "Novel Centroid Selection Approaches for K-Means Clustering Based Recommender Systems", *Information Sciences*, Vol. 320, pp. 156-189, Nov. 2015. doi: 10.1016/j.ins.2015.03.062

[31] S. Sharma, "A Recommender System Based on Improvised K- Means Clustering Algorithm", *International Journal of Research in Advent Technology*, Vol. 6, No. 7, pp. 1477-1483, July 2018.

[32] J. Bobadilla, R. Bojorque, A. H. Esteban, and R. Hurtado, "Recommender Systems Clustering Using Bayesian Non Negative Matrix Factorization", *IEEE Access*, Vol. 6, pp. 3549-3564, 2018. doi: 10.1109/ACCESS.2017.2788138.

[33] C. C. Aggarwal, Data Mining: The Textbook, Springer, pp. 285-343, 2015. doi:10.1007/978-3-319-14142-8

[34] S. Yu, Z-H Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," Knowledge Information Systems, vol. 14, pp. 1–37, 2008. doi:10.1007/s10115-007-0114-2

[35] T. K. Ho, "Random decision forests," in Proc.3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995, pp. 278–282. doi:10.1109/ICDAR.1995.598994

[36] C. Cortes, and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, pp. 273-297, 1995. doi:10.1007/BF00994018

[37] I. Guyon, and A. Elisseeff, "An introduction to variable and feature selection," Journal of Machine Learning Research, vol. 3, pp. 1157-1182, 2003.

[38] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, Feature Selection for High-Dimensional Data, Springer, pp. 31-40, 2015.

[39] G. Chandrashekar, and F. Sahin, "A survey on feature selection methods," Computers and Electrical Engineering, vol. 40, pp. 16–28, 2014. doi:10.1016/j.compeleceng.2013.11.024

[40] M. Y. H. Al-Shamri, Effect of Collaborative Recommender System Parameters: Common Set Cardinality and the Similarity Measure, Advances in Artificial Intelligence 2016 (2016), 10 pages

[41] A. Al-Junaid, T. Qaid, MYH Al-Shamri, M. Ahmed, A Raweh, Vertical and Horizontal DNA Differential Methylation Analysis for Predicting Breast Cancer, IEEE Access, vol. 6, pp. 53533-53545, 2018. doi: 10.1109/ACCESS.2018.2871027.

[42] N. Japkowicz and M. Shah, Evaluating learning algorithms: a classification perspective, Cambridge, pp. 74-159, 2011. doi:10.1017/CBO9780511921803.

[43] A. Zhang, Evaluating Machine Learning Models: A Beginner Guide to Key Concepts and Pitfalls, O'Reilly Media, pp. 7-12, 2015.