Expert Systems With Applications 71 (2017) 257–278

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Accurate multi-criteria decision making methodology for recommending machine learning algorithm

Rahman Ali[a,b], Sungyoung Lee[a,*], Tae Choong Chung[a]

[a] Department of Computer Engineering, Kyung Hee University (Global Campus), 1732, Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do 17104, Republic of Korea

[b] Quaid-e-Azam College of Commerce University of Peshawar, Peshawar, Postal Code 25120, Khyber Pakhtunkhawa, Pakistan

ARTICLE INFO

Article history:
Received 30 March 2016
Revised 22 November 2016
Accepted 23 November 2016
Available online 24 November 2016

Keywords:
Multi-criteria decision making
Algorithm recommendation
Algorithm selection
Classification algorithms
Classifiers recommendation
TOPSIS
Ranking classifiers

ABSTRACT

Objective: Manual evaluation of machine learning algorithms and selection of a suitable classifier from the list of available candidate classifiers, is highly time consuming and challenging task. If the selection is not carefully and accurately done, the resulting classification model will not be able to produce the expected performance results. In this study, we present an accurate multi-criteria decision making methodology (AMD) which empirically evaluates and ranks classifiers' and allow end users or experts to choose the top ranked classifier for their applications to learn and build classification models for them.

Methods and material: Existing classifiers performance analysis and recommendation methodologies lack (a) appropriate method for suitable evaluation criteria selection, (b) relative consistent weighting mechanism, (c) fitness assessment of the classifiers' performances, and (d) satisfaction of various constraints during the analysis process. To assist machine learning practitioners in the selection of suitable classifier(s), AMD methodology is proposed that presents an expert group-based criteria selection method, relative consistent weighting scheme, a new ranking method, called optimum performance ranking criteria, based on multiple evaluation metrics, statistical significance and fitness assessment functions, and implicit and explicit constraints satisfaction at the time of analysis. For ranking the classifiers performance, the proposed ranking method integrates Wgt.Avg.F-score, CPUTimeTesting, CPUTimeTraining, and Consistency measures using the technique for order performance by similarity to ideal solution (TOPSIS). The final relative closeness score produced by TOPSIS, is ranked and the practitioners select the best performance (top-ranked) classifier for their problems in-hand.

Findings: Based on the extensive experiments performed on 15 publically available UCI and OpenML datasets using 35 classification algorithms from heterogeneous families of classifiers, an average Spearman's rank correlation coefficient of 0.98 is observed. Similarly, the AMD method has showed improved performance of 0.98 average Spearman's rank correlation coefficient as compared to 0.83 and 0.045 correlation coefficient of the state-of-the-art ranking methods, performance of algorithms (PAlg) and adjusted ratio of ratio (ARR).

Conclusion and implication: The evaluation, empirical analysis of results and comparison with state-of-the-art methods demonstrate the feasibility of AMD methodology, especially the selection and weighting of right evaluation criteria, accurate ranking and selection of optimum performance classifier(s) for the user's application's data in hand. AMD reduces expert's time and efforts and improves system performance by designing suitable classifier recommended by AMD methodology.

1. Introduction

Machine learning algorithm selection is a real-world problem in various domains, such as data mining business, knowledge acquisition and reasoning, research and many others areas (Kumar, Sehgal, & Chauhan, 2012). Large business firms and research institutions hire machine learning experts, such as practitioners, data analysts and knowledge engineers to analyze the business data for different types of strategic planning. Usually, experts choose appropriate machine learning algorithm(s) using their heuristic knowledge about the domain and the available classification algorithms (Luo, 2015). The heuristics-based algorithm(s) selection is a risky task and sometimes result in selection of a sub-optimal performance algorithm(s). The reasons may include lack

* Corresponding author. Fax: +82312012514.
E-mail addresses: rahmanac1@gmail.com (R. Ali), sylee@oslab.khu.ac.kr (S. Lee), tcchung@khu.ac.kr (T.C. Chung).

http://dx.doi.org/10.1016/j.eswa.2016.11.034
0957-4174/

of the complete knowledge about the domain application, i.e., the datasets have different intrinsic characteristics, and the candidate classifiers have different capabilities and strengths. This process become more challenging when the selection of best classifier is based on multiple-criteria under strict conditions and constraints. According to the well-known "no free lunch" theorem (Wolpert & Macready, 1997), no machine learning algorithm performs well on all kind of learning problems. However, it can be made possible to estimate the selection of a suitable machine learning algorithm for an application in hand (Berrer, Paterson, & Keller, 2000). This selection process of the classifiers is an application dependent task, because it has been theoretically and empirically proved that no machine learning algorithm is universally superior on all datasets due to the different characteristics and features of the domain data (Caruana & Niculescu-Mizil, 2006).

In real-world applications, the requirements assessment of the applications and deciding which specific qualities need to be evaluated has great importance. Clear application's requirements easily clarify the ingredients of evaluation criteria and their individual contributions in the final decision making (Lavesson & Davidsson, 2007). The evaluation methods for different domains are different due to different objectives of the domains. Some domains require single evaluation criteria, while others need multi-criteria evaluation. In classification problems, the most commonly used single criterion metric used for evaluations is the accuracy, which can be evaluated using the well-known metrics, such as area under the ROC curve (Fawcett, 2006), success rate, average accuracy, and balanced accuracy. However, the evaluation only on the basis of accuracy may misleads the selection of optimum performance algorithm (Lavesson, Boeva, Tsiporkova, & Davidsson, 2014). To select optimum performance algorithm, multiple evaluation criteria, such as average accuracy, execution time, training time, consistency and many others need to be used. The objective of multi-criteria evaluation is to balance the trade-off between these criteria rather than maximizing a single criterion (Lavesson & Davidsson, 2007). The main issue in multi-criteria evaluation is the selection and prioritization of suitable criteria and excluding those which have conflicting behaviors. This is a subjective issue and requires the involvement of stockholders, such as domain experts and machine learning practitioners and users (Lavesson et al., 2014). In the criteria weight assignment, experts' preferences are quantified as weight scores and assigned to each metric of the evaluation criteria. The weights can be either assigned manually by experts or can be done using some semi-automatic weighting method, such as analytical hierarchy process (AHP) (Saaty, 2003).The manual weight assignment is a hard task, which has been realized by the simple and intuitive measure (SIM) (Soares, Costa, & Brazdil, 2000), measure-based evaluation (MBE) (Andersson, Davidsson, & Lindén, 1999) and application-oriented validation and evaluation (APPrOVE) (Lavesson et al., 2014) approaches. Statistical methods (Elazmeh, Japkowicz, & Matwin, 2006; Wang, Song, Zhang, & Zhang, 2014) have also been used for the evaluation of machine learning algorithms from different.

Apart from the criteria selection and prioritization issues, the non-uniformity of dimensionality of data for the evaluation metrics is another challenging issue (Zavadskas, Zakarevicius, & Antuchevicience, 2006). To overcome this issue, a number of normalization techniques (Hwang & Yoon, 2012; Lavesson et al., 2014) have been proposed in literature in which the unit or scale of measurements are transformed to a common compatible format to be fairly used in the evaluation process (Freitas, 2004).

In literature, a number of studies can be found that evaluates classifiers on the basis of single evaluation criterion, such as accuracy (Aha, 1992; Alexandros & Melanie, 2001; Brodley, 1993; Gama & Brazdil, 1995; Lindner & Studer, 1999; Smith, Woo, Ciesielski, & Ibrahim, 2002). The evaluation of classification algorithms on the basis of multiple criteria, such as accuracy and time, in non-simultaneous way, is presented in Ali and Smith-Miles, 2006, Brazdil, Soares, Costa, and da Costa, 2003, Lim, Loh, and Shih, 2000 and on the basis of sensitivity, precision, F-score, and area under the curve (AUC) is presented in Romero, Olmo, and Ventura, 2013. Ali and Smith, 2006 performed evaluation among 8 classifiers with 100 different classification problems using extended measures of average accuracy (true positive rate, true negative rate and percent accuracy) and time complexity (training time and testing time). Similarly, for various real-world applications, the performance evaluation of various classifiers have been done, for examples, handwritten recognition (Singh, Verma, & Chaudhari, 2016), color prediction of rice paddy plant leaf (Singh & Singh, 2016), prediction of diabetes mellitus (Kandhasamy & Balamurali, 2015; Perveen, Shahbaz, Guergachi, & Keshavjee, 2016) The most commonly used criteria for algorithms evaluation are the adjusted ratio of ratio (ARR) (Brazdil et al., 2003) and performance of algorithm (PAlg) on dataset (Song, Wang, & Wang, 2012), which use accuracy and time. Reif, Shafait, Goldstein, Breuel, and Dengel, 2014) used root mean squared error (RMSE) and Pearson product-moment correlation coefficient (PMCC) (Gayen, 1951) for the evaluation and recommendation of the best classification algorithm. The methods discussed in literature use absolute or partial relative weights to prioritize evaluation criteria. However, recently, the focus of researchers has shifted to relative criteria weighting, using multi-criteria. In medical knowledge acquisition, relative criteria weighting has been proposed (Khanmohammadi & Rezaeiahari, 2014) that uses AHP process (Saaty, 2003). They used average training time, accuracy and memory usage as the criteria. Five multi-criteria decision making methods, including TOPSIS (Tzeng & Huang, 2011 Jun 22), elimination et choix traduisant la realité III (ELECTRE III) (Figueira, Greco, & Ehrgott, 2005), grey relational analysis, vlse kriterijumska optimizacija i kompromisno resenje (VIKOR), and preference ranking organization method for enrichment of evaluations II (PROMETHEE II) have been discussed in article (Kou, Lu, Peng, & Shi, 2012). The contributions of state-of-the-art methods are significant, however a number of shortcomings still exist. These include, lack of appropriate guidelines and method for suitable criteria selection, consistent relative weight assignment to criteria, significance and fitness evaluation mechanism for algorithms performance results generation, constraints satisfaction and selection of the consistent algorithm.

To overcome these issues, an accurate multi-criteria decision making methodology (AMD) is proposed and experimented on 15 real-world datasets, from UCI (Lichman, 2013) and OpenML (Van Rijn et al., 2013) repositories, using 35 classification algorithms from Weka. According to this methodology, The problem of accurate classifier selection is tackled using the steps: (a) a user is given a machine learning task (e.g. predicting diabetes type based on the patient observations data); (b) The AMD method recommends top classifiers (from the basket of 35 classifiers) that would be most effective in solving this machine learning task. The main ingredients of the AMD method are (i) a machine learning task and the associated goal and objectives, (ii) a collection of classifiers and their performance evaluation metrics, called quality meta-metrics (QMM) (as well as sub-classification of QMM, i.e., performance metrics). A stage-wise processing of these metrics with input from domain experts in the forms of preferences (weights) to select correct performance evaluation criteria and weight them (iii) applying each classifier from the basket to training and testing datasets, and computing its performances based on the selected performance evaluation criteria and their weights, and (iv) ranking the classifiers using OPR method based on the results from step (ii) and step (iii), and (v) finally recommending the top k classifier to the user for his/her machine learning task.

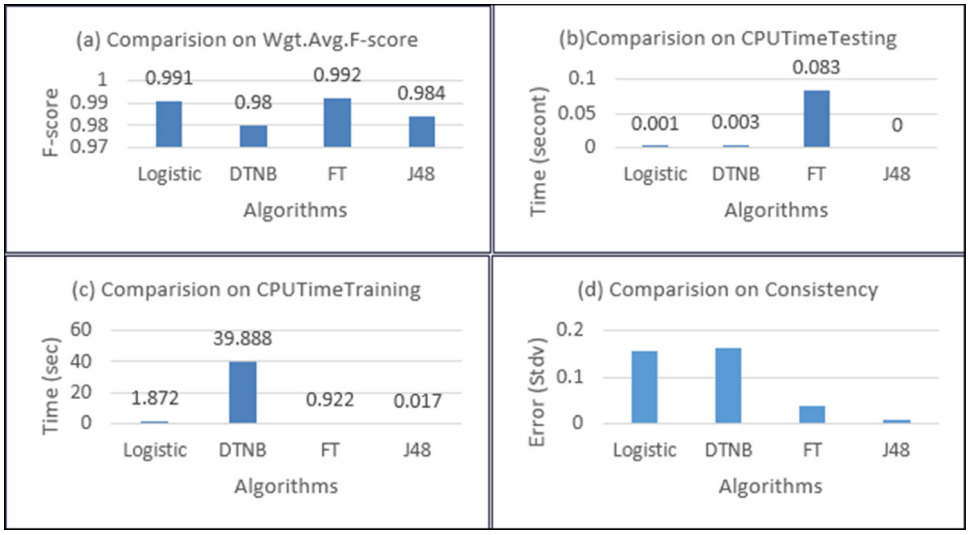

Fig. 1. Evaluation of algorithms on the basis of multiple evaluation criteria.

The key contributions made through this methodology are summarized as follows.

- A list of general guidelines are defined for performance evaluation of classifiers, based on extensive literature study of the classification algorithms.
- We analyzed and categorized classification algorithms' evaluation metrics and introduced the concept of classifiers quality meta-metrics (QMM) to construct QMM classification model, which is useful for non-experts of machine learning who need to make evaluation decision about classifiers selection. The QMM model further assists users in understanding physical meanings of the evaluation metrics.
- Proposed an experts' consensus-based group decision making method that assists experts to first select appropriate QMM and then select suitable evaluation criteria, satisfying interdependence and explicit global constraints, enforced by the objectives of the end user's application.
- An expert group-based relative criteria weighting technique is proposed, which can easily quantify and estimates experts' relative preferences about each evaluation criterion.
- A new ranking criteria, called optimum performance ranking (OPR) is proposed, which ranks classifiers based on Wgt.Avg.F-Score, CPUTimeTraining, CPUTimeTesting and Consistency performance metrics, integrated using TOPSIS method.
- Accurate statistical significance and fitness evaluation functions are defined, which inspect algorithms' fitness, prior to their inclusion in the final list of candidate algorithms for ranking.
- Implicit and explicit constraints are defined at different levels of the evaluation process for accurate ranking of the classifiers.

Rest of the paper is organized as follows. In Section 2, we formulate algorithm selection as a multi-metric decision making problem. In Section 3, methodology of the proposed multi-metric decision making process for algorithms selection is described technically. In Section 4, AMD methodology is validated using step-wise experiments on the classification algorithms and datasets. In Section 5, experiments and results are discussed in detail with comparison against state-of-the-art methods in the area of ranking classifiers on the basis of performance results. Finally, Section 6 concludes the work done and describes possible future extensions.

2. Algorithm selection: multi-metric decision making process

Each machine learning algorithm performs differently on different datasets because of different features of the data. The evaluation of these algorithms on the basis of single criterion sometimes misleads the decision of selecting best algorithm from a list of available candidate algorithms. For example, consider the following scenario with four classification algorithms: multinomial logistic regression, decision table/naive Bayes hybrid classifier (DTNB), functional trees (FT) and J48 which are tested on anneal dataset (Lichman, 2013) using 10×10-fold cross validation and evaluated using the criteria, Wgt.Avg.F-score, CPUTimeTesting, CPUTimeTraining and average consistency, as shown in Fig. 1.

Fig. 1(a) shows that FT algorithm performs well, measured in terms of weighted average f-score (0.992%) and is the winner amongst all algorithms. However, it performs poor from the CPUTimeTesting perspective (0.083 s). Similar interpretations can be made for CPUTimeTraining and the Consistency criteria. This analysis shows that no algorithm can be declared for all criteria.

From the empirical evidence, predictive accuracy is one of the traditional evaluation metric, estimated using cross-validation (Stone, 1974) that focuses on maximizing the accuracy, but ignores other criteria, such as comprehensibility, interestingness (Freitas, 2006) and complexity. The formal measurement of comprehensibility and interestingness may not be possible like accuracy, but it more relevant than accuracy when the objective is discovering accurate knowledge (Lavesson et al., 2014) in medical domain for recommendation generation services. Similarly, time and space complexities are also the key criteria for evaluating algorithms and selecting the right algorithm for an application in hand. In situation, where the datasets are either large or the storage space or computational power is limited (Buciluǎ, Caruana, & Niculescu-Mizil, 2006), the time and space complexities criteria need to be used for evaluation of the algorithms. Thus, in order to select appropriate classifiers or algorithms for such applications we must need to evaluate algorithms performance in terms of space and time complexities.

In light of the results shown in Fig. 1 and the empirical evidences from the literature, the well-known no-free-lunch theorems (Wolpert & Macready, 1997) is confirmed. Hence, we conclude the discussion that no classification algorithms is superior on all problems and is therefore no single evaluation criterion is always superior for their evaluation. If one algorithm outperforms others on

one criterion, it may underperforms on other criteria. As a consequence, the algorithm selection problem is a multiple criteria decision making problem which requires an accurate methodology to evaluate them properly. The rest of the study is focused to find a solution to this problem.

3. Material and methods

In this section, first we define a set of general guidelines and then describe the methodology for evaluating classification algorithms on the basis of multiple evaluation criteria.

3.1. Guidelines for algorithms evaluation

For selecting suitable algorithm(s), a sequence of essential tasks need to be performed. To efficiently perform these tasks, a set of guidelines are presented as follows.

1. Define an unambiguous goal for which the algorithm(s) need to be selected
2. Analyze and specify goal as either single-objective or multi-objectives and specify the corresponding quality meta-metrics (QMM)
   a. Categorize objective(s) as cost and benefit criteria
   b. Define essential constraints on the objective(s), reflecting goal's constraints
3. Analyze the specified objective(s) and constraints against existing criteria
   a. If existing criteria work, then go to step 4.
   b. If existing criteria do not fit well, then go to step 5.
4. Evaluate the algorithms performances using the available criterion under the constraints, defined in step 2(b), and rank them for the best selection
5. Define a generic multi-metrics evaluation criteria using the following steps
   a. Analyze QMM for conflict among evaluation criteria (interdependence/fuzziness)
   b. Select suitable QMM, defining the objectives.
   c. Select suitable evaluation metrics for the selected QMM (objectives)
   d. Prioritize the selected evaluation metrics
   e. Rank algorithms based on the aggregate value of the weighted metrics
   f. Repeat step 5, if any of the constraints, defined in step 2(b), is not satisfied

In the above guidelines, steps 1–4 are straightforward and can be easily followed. However step 5 is more challenging and needs technical contributions to accomplish the task of selecting suitable algorithm in the basis of multiple criteria. Generally, the outlined guidelines are generic, where only the domain specific parameters, such as the goal, objectives, evaluation criteria, and weights for each criterion need to be strictly followed while building a system. These guidelines are mainly focused on two essential aspects of the algorithms evaluation and recommendations systems. These aspects include (a) how to integrate multiple evaluation criteria and (b) what criteria should be integrated. To answer the first question, we designed and proposed a list of guidelines that were partially presented by Lavesson and Davidsson, 2007 and Lavesson et al., 2014. Similarly, to extend answer of the first question and find solution to the second question, we have provided detail description in the next section.

3.2. Multi-metric decision making for algorithm selection

The proposed accurate multi-metric decision making methodology (AMD) consists of the following steps: goal and objectives definition, criteria selection and weighting, measuring algorithm performance, ranking algorithms, and ordering and application.

Abstractly, the working methodology of AMD is described below, which is algorithmically presented in Algorithm 1 and pictorially depicted in Fig. 2.

- **Goal and objective definition:** describes the final goal, its corresponding objectives and the associated constraints to achieve the goal. For example, the selection of optimum performance classification algorithm for multi-class problems. In this statement, goal G is the "selection of optimum performance classification algorithm" and the global constraint C is "multi-class problems". The corresponding objectives against this goal can be, e.g., (o_1) accuracy, (o_2) computational complexity, and (o_3) consistency.
- **Criteria selection and weighting:** contains a set of methods to first select quality metrics for the objectives, then select suitable metric for each of the quality and finally assign consistent weight to each metric.
- **Measuring performance:** includes the tasks of generating performance results for the selected criteria using the candidate algorithms (considered in the study) on the datasets (one at a time) and performing significance and fitness tests. The purpose of this step is to generate significant matrix of the algorithms performance results for the selected evaluation criteria.
- **Ranking algorithms:** is used to rank the list of candidate algorithms by utilizing their performance results and the criteria weights.
- **Ordering and application:** consists of the trivial functions, such as sorting the ranked algorithms and selecting the top-k for the user's application in hand.
- **Constraints:** represent restrictions, i.e., for which family/families of problems the methodology should be activated (single class/multi-class), how experts' preferences should be quantified (explicit criteria weights), introduction of special criteria as constraint i.e., consistency, which is measured in terms of standard deviation.

The proposed AMD methodology is algorithmically represented in Algorithm 1.

In Algorithm 1, each step of the methodology is explicitly described in separate section except steps 9–11. In step 9, average ranking of the relative closeness scores RC of the algorithms are generated using the Microsoft Excel 2010 (Pinto, 2010) built-in function RANK.AVG() with its generic form RANK.AVG(number, ref, [order]). In step 10, the selectTopK() function is used to select top-k ranked algorithms while in step 11, the users build his/her model using the selected algorithms and deploy in their applications.

3.2.1. Selecting suitable quality Meta-metrics

To select an optimal performance algorithm, a machine learning (ML) user/expert must be aware of the physical meaning of the evaluation metrics. For understanding physical meaning of the evaluation metrics, we propose the idea to first abstract the evaluation metrics in the form of classifiers quality meta-metrics and then let the users know to select quality metrics compliant to their goal and objectives. This will help the users in identification of appropriate metrics and figuring out the conflicting (fuzzy) metrics, for example comprehensibility against correctness (accuracy) (Freitas, 1998) and complexity (Gaines, 1996). The conflicting criteria are interdependent among each other and need special treatment during evaluation. The independent (crisp) criteria are simple to evaluate and result in unbiased decisions.

3.2.1.1. Classifiers quality meta-metrics classification model. Classifiers can be evaluated using a number of commonly used evaluation criteria, such as RMSE, predictive accuracy and ROC curves

Algorithm 1 Selection of optimum performance algorithm on the basis of multi-metric evaluation.

Begin
inputs: **d** – dataset
A = {a_1, a_2, …, a_n} // list of n algorithms
output: **R** = top-k algorithms; where, R⊆A
Let QMM = Classifiers quality meta-metrics; // See Section 3.2.1.
1 [Define Goal]
G = {o_1, o_2, …, o_n}; // where, n is the number of objectives, See Section 3.2.
2 [Select Suitable Quality Meta-metrics]
$\tilde{\mathbf{Q}}$ = selectSuitQuality(**QMM**, **G**); // See Section 3.2.1
3 [Select Suitable Evaluation Metrics]
$\bar{\bar{\mathbf{Q}}}$ = selectSuitEvalMetrics($\tilde{\mathbf{Q}}$, **G**); //where, $\bar{\bar{\mathbf{Q}}} \subseteq \tilde{\mathbf{Q}}$. See Section 3.2.2.
4 [Estimate Relative Weight of the Evaluation Metrics]
W = estimateRelativeWeights($\bar{\bar{\mathbf{Q}}}$); //where, W is weight vector. See Section 3.2.3.
5 [Generate Performance Results of the Algorithms]
foreach algorithm **a** in **A** perform 10 × 10-fold cross-validation in Weka to produce an n*m performance matrix P for the evaluation metrics $\bar{\bar{\mathbf{Q}}}$. See Section 3.2.4.
end for
6 [Perform Statistical Significance Test]
$\tilde{\mathbf{P}}$ = performStatSigTest(**P**); //where, $\tilde{\mathbf{P}}$ is the significance labelled matrix. See Section 3.2.5.
7 [Perform Algorithm Fitness Test]
S = Perform Algorithm Fitness Test; See Section 3.2.6, Eq. (8)
8 [Compute Relative Closeness (RC) to Ideal Algorithm]
RC = rankAlgorithms(**S**, **W**); See Section 3.2.7.
9 [Rank the Algorithms]
RankedList = RANK.AVG($\mathbf{RC}_1$, $\mathbf{RC}_1$: $\mathbf{RC}_n$, 1);
10 [Select Top-K Algorithms]
R = selectTopK(RankedList, k);
11 apply **R** to learn **d**
End

(Fawcett, 2006). A general problem with users and domain experts is that they do not know physical meaning of the evaluation metrics. This creates difficulty for them to select suitable metric(s) for their evaluation. To resolve this problem, we define physical meaning of the classifiers evaluation metrics in terms of quality meta-metrics (QMM). We defined eight families of QMM for those evaluation metrics which are implemented in Weka library (Bouckaert et al., 2010). These include: responsiveness or computational efficiency, separability or coherency, robustness or sensitivity, consistency, correctness, complexity or simplicity, reliability and comprehensibility or interestingness or interpretability. The definitions of these qualities along with their evidences are given below.

- *Correctness*. It can be either measured directly from the correct cases or indirectly from the number of errors made. We categorize it into two sub-groups of accuracy ('+'cor) and accuracy ('−'cor). This family contains metrics for binary class problems, multi-class problems and balanced and imbalanced data problems.
- *Complexity*. It can be measured either in terms of time spent in building the model, i.e., computational complexity (ccom) or the memory space consumed to complete the process of building and accommodating the model, i.e., memory/space complexity (scom).
- *Responsiveness*. It measures the computational efficiency of a classifier in terms of testing or execution time. We abbreviated it as *res* that stands for responsiveness of the model.
- *Consistency*. Consistency of a classifier, with respect to an evaluation metric, can be measured in terms of its standard deviation. If the classifier maintain a certain level of performance for a subsets of the main dataset then it will be consistent otherwise inconsistent one. For example, standard deviation of the accuracy measure of a classifier over the 10-fold of a test dataset measures its consistency in terms of predictive accuracy (Lavesson et al., 2014). We abbreviated it as *con* in this study.
- *Comprehensibility, interestingness and interpretability*. It is combination of related subjective metrics that describes the nature of classifiers from the user's understanding and interpretation perspective. It measures the user oriented aspects, such as how well the classifier's output and the process of decision making be understood (Lavesson et al., 2014). These metrics are favored in the knowledge acquisition scenario where understandability matters (Freitas, 1998). Comprehensibility may also results in model complexity. A complex model is intuitively more difficult to understand and interpret as compared to a simple model (Gaines, 1996). Similarly, for a recommender system, the interpretability criterion has great importance, where user needs to understand and verify the results of a trained model. This quality metric is abbreviated as *com*.
- *Reliability*. This family of metrics measures how much the user can trust on the quality of correctness of the performance results of a classifier. It can be measured using error metrics, which are based on the probabilistic understanding of the errors that measures the deviation from the true probability, such as mean absolute error, mean squared error, LogLoss (cross-entropy), etc. (Ferri, Hernández-Orallo, & Modroiu, 2009). Similarly, information-theoretic metrics, also estimate the reliability aspects of classifiers (Sprinkhuizen-Kuyper, Smirnov, & Nalbantov, 2005). We abbreviated it as *rel* and categorized into distance or error measure (erel) and information-theoretic measure (irel).
- *Robustness*. It is a subjective measure used in diverse situations, such as ability of the classifier to make correct predictions on noisy dataset or a dataset with missing values (Han, Kamber, & Pei, 2011) or have high sensitivity or true positive rate (Lavesson & Davidsson, 2007). Sophisticated AUC measures have been reported recently for improving the quality of robustness of classifiers (Vanderlooy & Hüllermeier, 2008). We abbreviated it as *rob* in our study.
- *Separability and coherency*. In the context of binary classification problems, area under the receiver operating curve (AUC) is closely related to the concept of separability (Ferri et al., 2009). AUC can best distinguish the positive and negatives classes of a dataset. We abbreviate it as *sep* in our study.

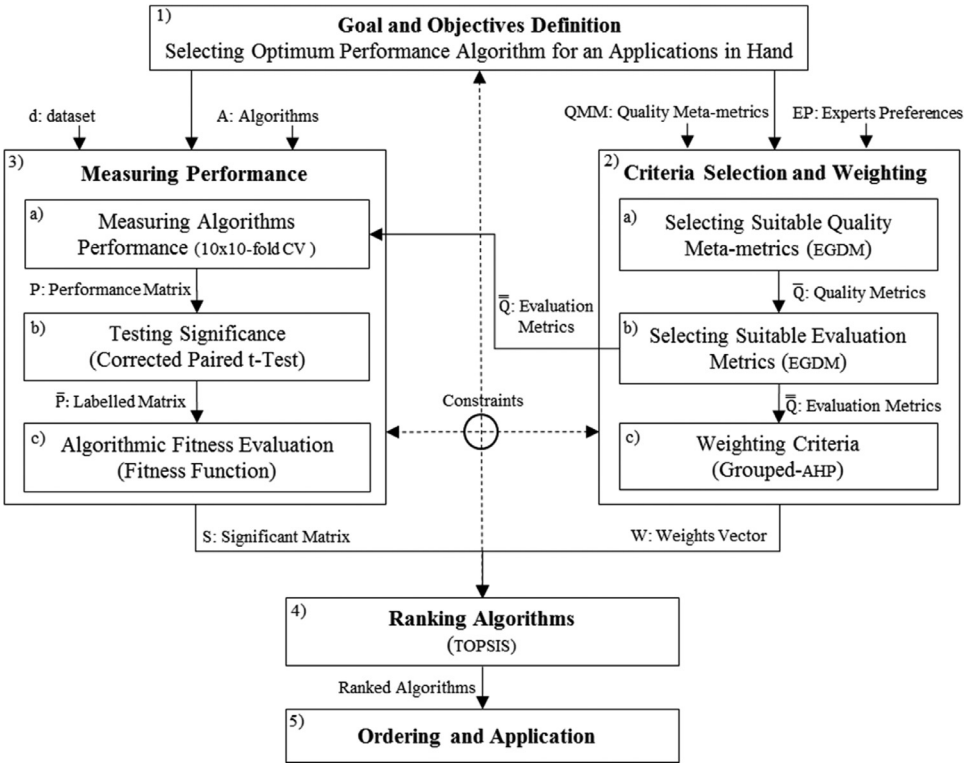

Fig. 2. A general view of the accurate multi-metric decision making methodology (AMD) for algorithms evaluation and selection.

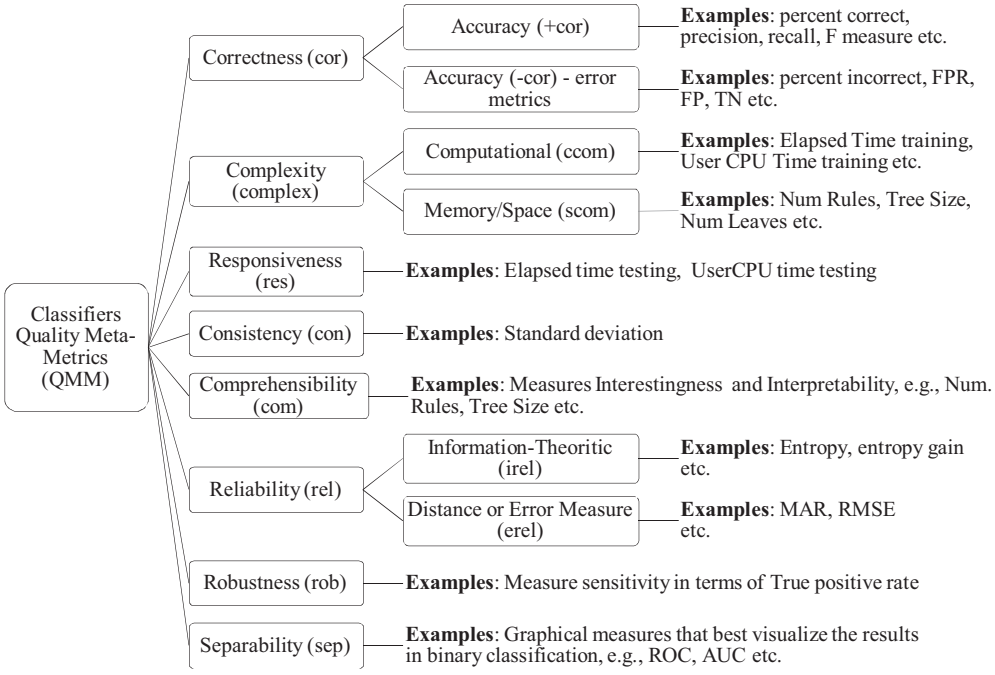

Fig. 3. Classification model of the classifiers quality meta-metrics.

A partly similar concept of classifiers qualities can be found from Tsiporkova, Tourwé, and Boeva, 2010 and Lavesson and Davidsson, 2007 with limited scope and number of qualities defined. We have proposed and defined a classification model for these qualities, as shown in Fig. 3.

While selecting qualities form the QMM classification model, to evaluate classifiers, intensive care should be taken to select only those qualities which satisfy the properties of legibility (containing sufficiently small number of criteria), operational, exhaustiveness (containing all points of view), monotonicity and non-redundancy (each criterion should be counted only once). These properties were initially defined in article (Bouyssou, 1990). A mathematical representation of the proposed QMM is shown in Eq. (1).

$$\mathrm{QMM} = \{\mathrm{cor},\ \mathrm{complex},\ \mathrm{res},\ \mathrm{con},\ \mathrm{com},\ \mathrm{rel},\ \mathrm{rob},\ \mathrm{sep}\} \quad (1)$$

Based on QMM classification model, the list of Weka classifiers' evaluation metrics are categorized, as shown in Table 1.

Table 1
Categorization of Weka classifiers evaluation metrics on the basis of quality meta-metrics.

Id	Evaluation Metric	QMM	Sub-QMM	Id	Metric	QMM	Sub-QMM
1	Number_correct	cor	+cor	27	Elapsed_Time_training	complex	ccom
2	Percent_correct	cor	+cor	28	UserCPU_Time_training	complex	ccom
3	Kappa_statistic	cor	+cor	29	measureNumRules	complex, com	scom
4	True_positive_rate	cor	+cor	30	measurePercentAttsUsedByDT	complex, com	scom
5	Num_true_positives	cor	+cor	31	measureTreeSize	complex, com	scom
6	False_negative_rate	cor	+cor	32	measureNumLeaves	complex, com	scom
7	Num_false_negatives	cor	+cor	33	measureNumPredictionLeaves	complex, com	scom
8	IR_precision	cor	+cor	34	measureNodesExpanded	complex, com	scom
9	IR_recall	cor	+cor	35	Elapsed_Time_testing	res	ures
10	F_measure	cor	+cor	36	UserCPU_Time_testing	res	sres
11	Weighted_avg_true_positive_rate	cor	+cor	37	SF_prior_entropy	rel	irel
12	Weighted_avg_false_negative_rate	cor	+cor	38	SF_scheme_entropy	rel	irel
13	Weighted_avg_IR_precision	cor	+cor	39	SF_entropy_gain	rel	irel
14	Weighted_avg_IR_recall	cor	+cor	40	SF_mean_prior_entropy	rel	irel
15	Weighted_avg_F_measure	cor	+cor	41	SF_mean_scheme_entropy	rel	irel
16	Number_incorrect	cor	-cor	42	SF_mean_entropy_gain	rel	irel
17	Number_unclassified	cor	-cor	43	KB_information	rel	irel
18	Percent_incorrect	cor	-cor	44	KB_mean_information	rel	irel
19	Percent_unclassified	cor	-cor	45	KB_relative_information	rel	irel
20	False_positive_rate	cor	-cor	46	Mean_absolute_error	rel	erel
21	Num_false_positives	cor	-cor	47	Root_mean_squared_error	rel	erel
22	True_negative_rate	cor	-cor	48	Relative_absolute_error	rel	erel
23	Num_true_negatives	cor	-cor	49	Root_relative_squared_error	rel	erel
24	Weighted_avg_false_positive_rate	cor	-cor	50	Area_under_ROC	sep, cor	-'
25	Weighted_avg_true_negative_rate	cor	-cor	51	Weighted_avg_area_under_ROC	sep, cor	-'
26	True_positive_rate	cor, rob	+cor		–	–	–

Procedure 1 selectSuitQuality.

Begin
inputs: QMM – classifiers quality meta-metrics
G – goal
output: Q″ – highly rated/ranked quality meta-metrics
1 [Select key qualities by each expert]
Q = extractSalientQMM(**QMM**, **G**); //where, Q ⊆ QMM
2 [Vote each quality by each expert]
Q′ = preliminaryVoteAggQuality(**Q′**); //where, Q′ is the initial list of selected QMM
a. If Q′ contains *Consistent* qualities, then
i. **Q″** = selectTopKQMM(**Q′**, **k**); // where, k represents the number of qualities experts are interested in
ii. **goto** setp 3;
b. Else
i. **repeat** step 2;
3 **return Q″**;
End

3.2.1.2. Selecting suitable quality meta-metrics. In this section, we proposed a formal expert group-based quality meta-metrics selection method, where a group of experts participate in a closed discussion and rate the quality metrics. We are motivated to the experts' group-based decision making method due to the effectiveness of nominal group technique (NGT) (Gallagher, Hares, Spencer, Bradshaw, & Webb, 1993) that quantifies the experts' preferences in the form of quantitative score. The proposed experts' group-based QMM selection process is represented in Procedure 1.

In Procedure 1, step 1, experts' panel uses extractSalientQMM() to extract those quality metrics from QMM classification model, which are essential for the evaluation of classifiers under the defined goal G. The salient qualities are collected by the head expert and presented for discussion, if needed, otherwise, preliminaryVoteAggQuality() is used (step 2,) to vote salient qualities by each expert. For voting salient qualities, rating or ranking methods can be used. The output of this function is to select top-k qualities, if they are consistent. A quality is said to be consistent if all the experts have uniformly rated/ranked it. For example, if ¾ of the experts rate correctness as rank 1 and only one expert rates it negatively, then it may be due to the inconsistent rating by the experts. In this case, re-voting is done and the process is continued till consensus are made. The final output of Procedure 1 is the list of most desirable qualities for the defined goal.

3.2.2. Selecting suitable evaluation-metrics

Once suitable qualities, $\bar{Q}$, are selected, the next step is to select suitable evaluation metrics. However, in case of classification algorithms, for each $\bar{Q}$, a large number of metrics are available (a few are shown in Table 1).

The selection of suitable metrics (i.e., metrics to integrate) depends on the scope of the classifiers under analysis, which is defined in terms of the number of families of classifiers taken under consideration. A few of the commonly used families of classifiers, i.e., probabilistic family, lazy learners' family, function family, rule family, decision tree family and meta-learners family, are implemented in Weka (Bouckaert et al., 2010), which are focused in this study. Apart from the scope of the classifiers, the domain/application requirements also influence the selection of suitable metrics. To resolve the metrics selection problem, we adopt the idea of experts group-based decision making, motivated by the NGT (Gallagher et al., 1993). The methodology used is algorithmically represented in Procedure 2.

In Procedure 2, step 1, experts' panel uses extractSalientMetrics() to extracts those quality metrics from $\bar{Q}$, which qualify

Procedure 2 selectSuitEvalMetrics.

Begin
inputs: $\bar{\mathbf{Q}}$ –highly rated/ranked quality meta-metrics
G – goal
output: **SM**″ –highly rated/ranked evaluation metrics
Let specAlgEvlMetrics = Specification of evaluation metrics. See Table 1.
1 [Select salient evaluation metrics (SM) from each quality metric]
SM = extractSalientMetrics($\bar{\mathbf{Q}}$, **G**, **specAlgEvlMetrics**);
2 [Vote each evaluation metric by each expert]
SM′ = preliminaryVoteAggMetrics(); //where, SM′ is initial list of selected metrics
a. If **SM**′ contains *Independent* metrics, then
i. **SM**″ = selectTopKSuitMetrics(**SM**′, **k**); //where, SM″ ⊂ SM′ and k is the number of metrics
ii. **goto** setp 3;
b. Else
i. **repeat** step 2;
3 **return SM**″;
End

the goal G. The salient evaluation metrics from each quality are extracted by utilizing specAlgEvlMetrics (see Table 1). This process is completed in step 2 by using preliminaryVoteAggMetrics(). For voting the same method as described in previous section is used. The output of this function is to select top-k metrics, if they are crisp/independent. An evaluation metric is said to be independent if it is not duplicate with other metrics. For example, percent accuracy and percent incorrect/errors are interdependent evaluation metrics and both should not be included in the evaluation metrics. The final output of this procedure is the list of selected suitable evaluation metrics SM″, which are the main ingredients of the generic multi-metric criteria Our focus is to select metrics that have the following features: (a) easily computable, (b) perform best on all types of datasets, (c) coherent with the final decision, (d) non-conflicting/independent of each other, (e) same representation with same scale, (f) quantifiable/measurable and (g) related with the algorithms evaluation. While selecting metrics, preference should be given to those metrics that qualify maximum of these qualities (Majumder, 2015).

3.2.3. Consistent relative criteria weighting

The selected evaluation metrics are the final ingredients of the evaluation criteria that play their corresponding roles in achieving the final goal. The roles define the preference or priority or weight of the metrics, which should be first estimated and then used during evaluation. State-of-the-art algorithm evaluation and recommendation studies, discussed in literature, follow absolute or partial relative weighting techniques that support limited number of criteria. The weights are assigned by experts, utilizing their own knowledge of the domain. In order to resolve shortcomings of the existing work, we proposed the idea of group decision making for consistent relative weights of the criteria. For this task, we are motivated by the AHP weighting method (Saaty, 1990), which has the ability to quantify experts' preferences in the form of weight scores, using the pairwise-wise comparisons procedure utilizing Saaty's preference scale (SPS) (Saaty, 1980), shown in Table 2.

According to the interpretation of this scale, if an evaluation metric e_1 is *extremely more important* than evaluation metric e_2, it is rated as 9 and then e_2 must be *extremely less important* than e_1, which is rated as 1/9. Table 2 has all the possible values of importance of evaluation criteria and its inverse along with their interpretations.

For weighting the evaluation criteria, the AHP expert group-based prioritization mechanism is followed in the sequence: prioritizing experts, creating a pairwise comparison matrix of the selected metrics ($\bar{\bar{\mathrm{Q}}}$), assigning experts' relative priority weights, evaluating consistency of the individual weights and aggregating individual's weights into group weights. The process is described in Procedure 3.

In step 1 of the Procedure 3, an n*n comparison matrix (DMM) is designed to estimate the decision power of each decision maker. These weights are assigned using function estimateDMWgt() (step 2). The weights are estimated using the AHP pairwise comparison procedure. Each entry dm_{ij} of the matrix DMM is entered by the head expert, on the basis of his/her understanding about the expertise of other experts (DM). Each of these values represents the superiority of i^{th} DM relative to the j^{th} DM. If $dm_{ij} > 1$, then the i^{th} DM is more influential in decision making than the j^{th} DM, but if $dm_{ij} < 1$, then the i^{th} DM is less influential than the j^{th}DM. However, if $d_{mij} = 1$ both i^{th} and j^{th} DM have the same level of importance in the decision. For estimating the DM decision weights, $DMM = dm_{ij}$ is first transformed to the normalized matrix, $\overline{DMM} = \overline{dm}_{ij}$, where each entry $\overline{dm}_{ij}$ is computed using Eq. (2) and then a column weight vector $W = w_j$ is produced using Eq. (3),

$$\overline{dm}_{ij} = dm_{ij} \Big/ \sum_{i=1}^{n} dm_{ij} \tag{2}$$

$$w_j = \sum\nolimits_{j=1}^{n} \overline{dm}_{ij}/n = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}, \text{ where } i \,\&\, j = 1, 2, \ldots, n. \tag{3}$$

To verify correctness of the experts' judgment and preferences about the relative weights assigned to each DM, consistency is checked using the eigenvector method (Saaty, 2003), which computes consistency ratio (CR) using Eqs. 4–7,

$$CR = CI/RI, \tag{4}$$

where, RI is the random consistency index value from the random consistency table (Saaty, 1980), shown in Table 3. Similarly, the value of CI measures the deviation which is computed using Eq. (5),

$$CI = (\lambda_{max} - n)/(n - 1), \tag{5}$$

where, λ_{max} is the principal eigenvalue, such as $\lambda_{max} \in \Re > n$. The eigenvalue is computed by averaging values of the consistency vector Cv, as shown in Eq. (6),

$$\lambda_{max} = \left(\sum_{i=1}^{n} Cv_{ij} \right)/n, \tag{6}$$

where, each value Cv_{ij} of the consistency vector, is computed by taking product of the pairwise comparison matrix DMM with the

Table 2
Saaty's preference scale for pair-wise comparison of the evaluation criteria.

Definition	Intensity of importance	Definition	Intensity of importance
Equally important	**1**	**Equally important**	**1/1**
Equally or slightly more important	2	*Equally or slightly less important*	1/2
Slightly more important	**3**	**Slightly less important**	**1/3**
Slightly to much more important	4	*Slightly to way less important*	1/4
Much more important	**5**	**Way less important**	**1/5**
Much to far more important	6	*Way to far less important*	1/6
Far more important	**7**	**Far less important**	**1/7**
Far more important to extremely more important	8	*Far less important to extremely less important*	1/8
Extremely more important	**9**	**Extremely less important**	**1/9**

Procedure 3 estimatRelativeWeight.

Begin
inputs: $\overline{\overline{Q}} = \{e_1, e_2, \ldots, e_m\}$; // selected evaluation metrics
output: **W** – weights vector **Let DM** = $\{dm_1, dm_2, \ldots, dm_n\}$; // Group of experts
SPS = Saaty's preference scale (see Table 2)
GDMM = m*n 'group decision making matrix', where m represents metrics and n represents decision makers
1 [Design comparison matrix for decision makers]
DMM = dm_{ij}; //where, DMM is n*n comparison matrix of decision makers with dm_{ij} is the decision weight of the i^{th}decision maker relative to the j^{th}decision maker
2 [Estimate decision makers decisions weight]
a. **DMWeight** = estimateDMWgt(**SPS**, **DMM**); //where, DMWeight is a single column weights vector containing preferences of decision makers. // See Eqs. 2 and 3
b. Check *consistency* of **DMWeight**; // See Eqs. 4–7
3 [Estimate metrics weights]
for dm = 1 to **n** do
a. [Design comparison matrix for evaluation metrics]
EM = e_{ij}; //where, EM is m*m comparison matrix of the evaluation metrics with e_{ij} is the preference of i^{th} metric against the j^{th} metric
b. $\mathbf{EMWeight_{dm}}$ = estimateEvalMetricsWgt(**SPS**, **EM**); //where, EMWeight is single column weights vector for metrics $\overline{\overline{Q}}$. // See Eqs. 2 and 3
c. Check *consistency* of $\mathbf{EMWeight_{dm}}$; // See Eqs. 4–7
d. Insert ⟨$\mathbf{EMWeight_{dm}}$⟩ into **GDMM**;
End for
4 [Aggregate weights of all decision makers using group decision making]
foreach e ∈ **GDMM**
$\mathbf{W} = \sum(\prod_{dm=1}^{n}(\mathbf{DMWeight^T}, \mathbf{EMWeight}))$; //where, W is the aggregate weights vector
End for
5 **return W**;
End

Table 3
Random consistency indices (RI) for different number of evaluation criteria (n).

Number of evaluation criteria (n)	1	2	3	4	5	6	7	8	9	10	11
Random consistency index (RI)	0.00	0.00	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49	1.51

weight vector W. This relationship is shown in Eq. (7),

$$Cv_{ij} = E * W. \quad (7)$$

In step 3 (a), m comparison matrices (i.e., EM) are created, one for each decision maker to relatively weight all the evaluation metrics $\overline{\overline{Q}}$. In step 3 (b), each decision maker (dm) uses the function estimateEvalMetricsWgt() to assign and estimate the weight for each evaluation metric. In step 3 (c) the consistency of metrics weights are checked using Eqs. 4–7. In step 3 (d), the weight vector EMWeight is added to the group decision making matrix, GDMM. In step 4, the weights estimated for the evaluation metrics $\overline{\overline{Q}}$ by the n decision makers, DM, are aggregated using the group decision making process, which are returned to the main Algorithm 1 using step 5.

3.2.4. Measuring algorithms performance

In this phase, first the candidate list of algorithms are selected from the pool of freely available classification algorithms. We selected 35 multiclass classification algorithms, from six heterogeneous families of the classifiers, implemented in Weka (Bouckaert et al., 2010). The list of these algorithms is shown in Table 4.

Table 4
List of Weka classifiers for evaluation, comparison and selection of the best classifier.

SNo	Classifier	SNo	Classifier
1	bayes.BayesNet	19	trees.J48
2	bayes.NaiveBayes	20	trees.J48graft
3	bayes.NaiveBayesUpdateable	21	trees.LADTree
4	functions.Logistic	22	trees.RandomForest
5	functions.RBFNetwork	23	trees.RandomTree
6	functions.SMO	24	trees.REPTree
7	misc.HyperPipes	25	trees.SimpleCart
8	misc.VFI	26	meta.AdaBoostM1
9	rules.ConjunctiveRule	27	meta.Bagging
10	rules.DecisionTable	28	meta.Dagging
11	rules.DTNB	29	meta.END
12	rules.JRip	30	meta.FilteredClassifier
13	rules.OneR	31	meta.LogitBoost
14	rules.PART	32	meta.RacedIncrementalLogitBoost
15	rules.Ridor	33	meta.RandomSubSpace
16	rules.ZeroR	34	meta.Stacking
17	trees.BFTree	35	meta.Vote
18	trees.FT	—	—

Procedure 5 performStatSigTest.

Begin
inputs: P – performance matrix
output: $\bar{\mathbf{P}}$ – n*m performance matrix, where n is the number of algorithms and m is the number of evaluation metrics;
Let d – given dataset
$\mathbf{A} = \{a_1, a_2, .., a_n\}$ – set of classification algorithms
$\overline{\overline{\mathbf{Q}}} = \{e_1, e_2, .., e_m\}$- set of evaluation metrics
1 **foreach e** $\in \overline{\overline{\mathbf{Q}}}$ in performance matrix **P**
a. **if e** $\in$ benefit metric
i. **referenceAlg** = selectReferenceAlg(maxPerformValue(**e**));
b. **else**
i. **referenceAlg** = selectReferenceAlg(minPerformValue(**e**));
c. $\bar{\mathbf{P}}$ = performCorrectedPairedtTest(**referenceAlg**, **P**, **e**);
2 **end for**
3 **return** $\bar{\mathbf{P}}$
End

To rank these algorithms, A, on a classification dataset, *d*, using the performance results of evaluation metrics $\overline{\overline{Q}}$, all the algorithms (A) are executed sequentially on dataset *d* in Weka environment and the results are stored into the performance matrix P for later use.

3.2.5. Testing significance of performance results

Unlike the traditional ranking methods that directly select top-rank algorithm (without considering significance tests of the results) for learning models, we propose the idea of checking the performance results for statistical significance. According to this idea, the performance results of the candidates algorithms A are first tested for statistical significance and then the for the significance fitness. The objective of significance test is to identify which algorithms perform significantly better, which perform significantly poor and which perform similar with respect to a reference algorithm. For this purpose, we adopted corrected paired *t*-test with significance of 0.05 (Witten & Frank, 2005) implemented in Weka (Bouckaert et al., 2010), which checks the significance of the algorithms results and labels them either 'v' (for better performance), or '*' (for worst performance) or '' for equal significance performance with respect to a baseline algorithm. In our case, the definition of the reference algorithm $a \in A$ is the algorithm which performances best as compared to all the algorithms. The selection of the reference for each metric $e \in \overline{\overline{Q}}$ is done within its local scope rather than the global scope of all metrics $\overline{\overline{Q}}$.

For a performance matrix $P = p_{ij}$, with p_{ij} as the performance value of i^{th}algorithm on the j^{th}evaluation metric, the process of corrected paired *t*-test and the production of final labelled performance matrix $\bar{P} = \bar{p}_{ij}$ is described in Procedure 5.

In Procedure 5, the criteria for selecting reference algorithm is the maximum value for a benefit metric and minimum value for a cost metric, respectively. Benefit metric are those whose higher values are preferred, e.g., accuracy, while cost metrics are those whose lower value is preferred, e.g., training time. For labeling the algorithms as either significant, or poor or equal in performance, step 1(c) is used. For this purpose, Weka corrected paired *t*-test is used, which takes reference algorithm (referenceAlg), single evaluation metric (e) and the performance matrix (P) together as inputs and returns a labelled matrix ($\bar{p} = \bar{p}_{ij}$) as output. Each value $\bar{p}_{ij}$ of the labelled matrix is either labelled as (v) or, (*) or ('').

3.2.6. Algorithmic fitness evaluation

In this step, the algorithms' fitness levels are evaluated for consideration in the next step of evaluation. The motivation for including the fitness evaluation as an additional step is to reduce the algorithm space by filtering out the algorithms that poorly perform on all evaluation metrics on a single dataset. This is reasonable and makes sense that not to allow poor performance algorithms to the next stage of evaluation. Furthermore, it reduces the chance of selection of bad algorithm.

To implement this idea, we proposed a fitness function that evaluates labels in the labeled performance matrix $\bar{P} = \bar{p}_{ij}$. This function can be defined as follows. Let $\overline{\overline{Q}} = \{e_1, e_2, .., e_m\}$ be the set of m evaluation metrics for evaluating performance of an algorithm $a \in A$ on a classification dataset d and $\bar{P} = \bar{p}_{ij}$ be the labeled performance matrix, obtained after significance test. The target significant matrix S, containing the list of significantly fit algorithms, can be generated using the fitness function, defined in Eq. (8),

$$S = \left\{ \forall_{a \in A} : a \in \bar{P} | \forall_e : e \in \overline{\overline{Q}}. \sim \text{nonSignificant}(e) \right\}, \tag{8}$$

where, nonSignificant(e) is the function that determines the significance level of each $a \in A$ for each evaluation metric $e \in \overline{\overline{Q}}$ and returns true if it either performs significantly better or equal and add to the significant matrix S. The process is repeated for all algorithms A against all metrics $\overline{\overline{Q}}$ and the final results are accumulated in S, which is the reduced version of the original labelled matrix $\bar{P}$, in terms of number of candidate algorithms i.e., SizeOf(S) < SizeOf($\bar{P}$). Internally, the function nonSignificant(e) processes the labels, i.e., 'v', '*' and '', of the values of each metric $e \in \overline{\overline{Q}}$, assigned by the corrected paired *t*-test of the procedure 4. In the significant matrix S, each value is represented by s_{ij}, where i represents the algorithm and j represents the evaluation metric.

3.2.7. Ranking algorithms

State-of-the-art methods for ranking algorithms are based on the aggregate score of multiple evaluation metrics $\overline{\overline{Q}}$, combined together in different ways, consuming absolute weights, which are assigned by domain experts and lake appropriate normalization mechanism for the values of the criteria. These methods have minimal support for extension in terms of number of metrics to be added and lack support for implicit and explicit constraints satisfaction. Our idea is to evaluate the candidate algorithms and rank them according to their relative closeness score to the ideal algorithm with the consumption of relative consistent weights and different constraints. To achieve these objectives, we are motivated by the flexibility and ranking power of the TOPSIS multi-criteria decision making method (Garcia-Cascales & Lamata, 2012; Tzeng & Huang, 2011 Jun 22). The TOPSIS steps used during algorithms ranking are shown in Procedure 6.

The value RC lies between 0 and 1, i.e., $0 \leq RC \leq 1$. If $RC = 1$, the TOPSIS has the best condition of the top-k algorithms selection; and if $RC = 0$, the TOPSIS has the worst condition of algorithm selection. Any other value in-between these two values measures the appropriateness level of that algorithm.

Procedure 6 rankAlgorithms.

Begin
inputs: S – n*m matrix containing performance results of significant algorithms
W – 1*m (single row) weight vector
output: RC – n*1 (single column) matrix of the relative closeness score
Let d –dataset
$\mathbf{A} = \{a_1, a_2, .., a_n\}$ – set of classification algorithms
$\overline{\overline{\mathbf{Q}}} = \{e_1, e_2, .., e_m\}$ – set of evaluation metrics
1 [create performance evaluation matrix from S]
$\mathbf{S} = (s_{ij})_{n*m}$; //where, s_{ij} represents value of algorithm i for evaluation metric j
2 Define local/implicit constraints on $\overline{\overline{\mathbf{Q}}}$;
3 [normalize performance evaluation matrix S]
$\bar{\mathbf{S}} = \mathbf{r_{ij}} = s_{ij} / \sqrt{\sum_{i=1}^{n} s_{ij}^2}$; //where, i =1, 2, ..., n and j=1, 2, ..., m
4 [compute weighted normalized decision matrix V]
$\mathbf{V} = (\mathbf{v_{ij}})_{n*m} = \mathbf{r_{ij}} * \mathbf{W_j}$; //where, W_j is the weight vector
5 [compute positive ideal solution (PIS) and negative ideal (NIS) solution]
a. $\mathbf{PIS} = \{(\max_i(v_{ij}) | j \in C_b), (\min_i(v_{ij}) | j \in C_c)\} = \{v_j^* \mid j = 1, 2, \ldots, m\}$
b. $\mathbf{NIS} = \{(\min_i(v_{ij}) | j \in C_b), (\max_i(v_{ij}) | j \in C_c)\} = \{v_j^- \mid j = 1, 2, \ldots, m\}$
6 [compute separation measures using m-dimensional Euclidean distance]
a. $\mathbf{PIS_i^*} = \sqrt{\sum_{j=1}^{m} (v_{ij} - v_j^*)^2}, \; j = 1, 2, \ldots, m$
b. $\mathbf{NIS_i^-} = \sqrt{\sum_{j=1}^{m} (v_{ij} - v_j^-)^2}, \; j = 1, 2, \ldots, m$
7 [compute relative closeness (RC) of algorithms with respect to ideal algorithm]
$\mathbf{RC} = \frac{\mathbf{NIS_i^-}}{\mathbf{PIS_i^* + NIS_i^-}}, \; i = 1, 2, \ldots, n$; where, RC is a n*1 matrix
8 **return RC**;
End

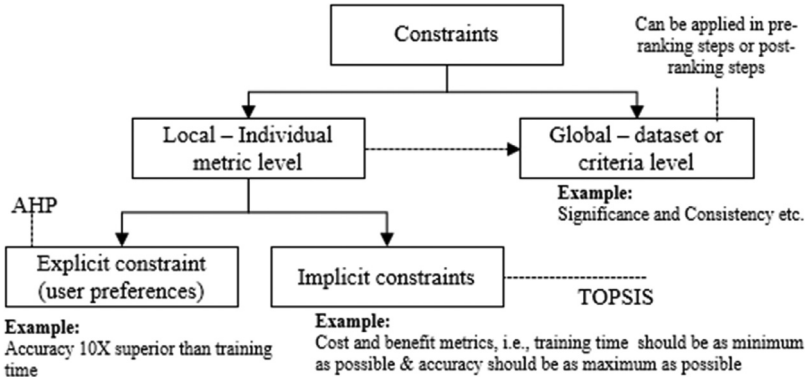

Fig. 4. Classification of constraints on the evaluation criteria.

3.2.8. Constraints satisfaction

The constraints used in our study can be categorized into individual level, limited to a single metric of the evaluation criteria, and global level, applicable to all the metrics in the evaluation criteria. Individual level constraints are satisfied in the pre-ranking and ranking steps of evaluation process. These are further categorized into explicit and implicit constrains. The explicit constraints are defined by the users or experts, such as users' relative preferences on the metrics. An example can be, "the accuracy metric should be favored 10-times as compared to training time". The implicit constraints are the inherit characteristics of individual metrics, such as the value of cost criteria which should be as minimum as possible and the values of benefit criteria should be as higher as possible. Contrary to the local constraints, global constraints are the explicit constraints that are based on the local constraints and applicable to the overall criteria in the pre-ranking evaluation process. Examples of the global constraints are the consistency of estimated weights of the criteria, significance of the performance results of the algorithms and consistency in the performance results of the algorithms. Fig. 4 shows different types of constraints with their examples that are applied at different levels of the algorithms evaluation and ranking process.

In this paper, for satisfaction of the local constraints, we proposed the idea of relative weighting using AHP process, and the idea of cost and benefits analysis of the metrics using the TOPSIS method. Similarly, for the satisfaction of global constraints, we adopted the AHP weights consistency check methods using eigenvector computation, and proposed the idea of paired *t*-test embedded in the algorithmic fitness evaluation function for checking the significance of the algorithms performance results. The local constraints can be satisfied through the configuration of AHP and TOPSIS methods, but the global constraints' satisfaction need more advanced level user-defined functions. We measure the consistency of algorithms in terms of standard deviation of their results. The algorithm that has lowest standard deviation value is the consistent algorithm and vice versa.

4. Validation of the AMD methodology - a scenario

In order to evaluate the effectiveness of AMD methodology, verify its potential use in real-world scenarios and allow other researchers to confirm our results, we perform step-by-step process in this section with the necessary experiments. First consider a scenario in which a user is interested in learning his dataset with

Table 5
Characteristics of UCI machine learning repository/OpenML datasets.

Datasets	Characteristics of Datasets						
	Attributes	Nominal Attributes	Numeric Attributes	Binary Attributes	Classes	Instance Count	Missing
abalone-3class	9	1	7	0	3	4177	0
rabe-148	9	1	7	0	3	4177	0
acute-inflammations-nephr	6	0	5	0	2	66	0
ADA_Agnostic	7	5	1	5	2	120	0
ADA_Prior	49	0	48	0	2	4562	0
adult-4000	15	8	6	1	2	4562	88
adult-8000	15	8	6	1	2	3983	0
aileron	15	8	6	1	2	8000	0
analcatdata-AIDS	41	0	40	0	2	5795	0
analcatdata-apnea2	5	2	2	0	2	50	0
analcatdata-apnea2	4	2	1	0	2	475	0
analcatdata-asbestos	4	2	1	0	2	475	0
analcatdata-authorship	4	2	1	1	2	83	0
analcatdata-bankruptcy	71	0	70	0	4	841	0
analcatdata-birthday	7	1	5	0	2	50	0

Table 6
Experts' group-based rating of quality metrics for heterogeneous classifiers.

Quality Metrics	DM#1	DM#2	DM#3	DM#4	Total
Correctness (cor)	60	50	55	70	**235**
Computational Complexity (ccom)	5	20	15	–	**40**
Responsiveness (res)	15	–	20	20	**55**
Consistency (con)	10	15	–	–	**25**
Comprehensibility (com)	–	15	–	7	**23**
Reliability (rel)	5	–	–	–	**5**
Robustness (rob)	–	–	10	3	**13**
Separability (sep)	5	–	–	–	**5**
Total	**100**	**100**	**100**	**100**	**400**

*[*Each expert distributes* 100 *points across the qualities metrics*]

Table 7
Selected evaluation metrics for evaluation of the heterogeneous classifiers.

Evaluation Metrics	(DM#1 - DM#5) Decision maker
Correctness (cor)	Wgt. Avg. F-score
Computational Complexity (ccom)	CPUTimeTraining
Responsiveness (res)	CPUTimeTesting
Consistency (con)	Consistency (Stdev.)

a classification algorithm, which he does not really know. The key problem he faces is the selection of an optimum performance classification algorithm that fits well into his requirements and expectations, expressed in terms of goal and associated objectives. In this scenario, the user is given a choice to select the best algorithm from a list of most commonly used 35 multi-class classification algorithms, shown in Table 4 for the 15 classification datasets,[1] sown in Table 5. Due to the space issue, the AMD steps are described only for one dataset.

A machine learning practitioner can use the proposed AMD methodology as follows.

Step 1: Goal and objectives definition

The goal of the study is to *select an optimum performance multi-class classification algorithm from the heterogeneous families of algorithms* (see Table 4) for binary and multiclass problems (see Table 5) that has optimum performance.

Step 2: Selecting suitable quality meta-metrics

For the goal in step 1, Procedure 1 is used to select the suitable quality metrics. Four machine learning experts, i.e., machine learning and data mining expert (DM#1), a data and knowledge engineering expert (DM#2), a scientist, researcher and developer (DM#3) and an expert user of the classification algorithms in diverse application area (DM#4) were chosen to select the qualities. Using Procedure 1, the experts selected correctness (accuracy), responsiveness, computational complexity and consistency (as shown in Table 6) as the relevant qualities that are compliant to the goal and satisfy the heterogeneity constraint of the classifiers.

Table 6 shows the importance score of each quality metrics. The top 4 qualities are non-conflicting and reflect the general characteristics of all the classifiers, therefore they are selected. These qualities are represented in Eq. (9),

$$\bar{Q} = \{\text{cor},\ \text{ccom}, \text{res}, \text{con}\}. \tag{9}$$

The physical meaning of Eq. (2), is that the optimum performance algorithm is the one that has high level of correctness in its results, low computational complexity, quick response time to users' requests, and high consistency in its results for a test dataset.

Step 3: Selecting suitable evaluation metrics

Procedure 2 is used to assist expert in the selection of suitable evaluation metrics, shown by Eq. (10) and Table 7, respectively,

$$\bar{\bar{Q}} = \{\text{Wgt. Avg. F} - \text{score},\ \text{CPUTimeTraining}, \text{CPUTimeTesting}, \text{Consistency}\}. \tag{10}$$

In Table 7, the consistency metric cannot be directly measured by any of the metric shown in Table 1. It is defined by the experts in their discussion of voting for metrics selection. It is a global explicit constraint that helps in selecting an algorithm that has consistent results.

Step 4: Weighting Metrics

The estimation of evaluation metrics is done using Procedure 3 and the results are shown in Table 8 and Fig. 5. Weights of the decision power of each decision maker is shown in Table 8(a). The relative weights, for each metric, estimated by each decision maker, are shown in Table 8(b-e). The final, experts' group-based weights are shown in Table 8(f).

According to the weight scores of these metrics, Wgt. Avg. F-score is the most preferable, followed by consistency, followed by CPUTimeTesting followed by CPUTimeTraining.

Step 5: Measuring algorithms performance

For generating real performance results of the 35 classification algorithms, Weka environment is used. Table 10, column 2–5, shows the results for ADA_Agnostic dataset (Van Rijn et al., 2013). The consistency column 5 of Table 10 is not directly measurable using the existing evaluation metrics, therefore we compute it by averaging standard deviations of the first three evaluation metrics,

[1] Some of the datasets are used with minor modifications by changing the type of the class label to nominal etc.

Table 8
Criteria relative weighting, based on experts group decision making using analytical hierarchy process (AHP) pair-wise comparison.

(a). Experts' (decision makers') decisions' prioritization

DM/DM	DM#1	DM#2	DM#3	DM#4	DM Decision Weights
DM#1	1	3	2	5	0.49
DM#2	0.33	1	1	3	0.21
DM#3	0.50	1.00	1	3	0.23
DM#4	0.20	0.33	0.33	1	0.08
				CI: 0.009	**1.00**

(b) DM#1 relative weighting

Criteria/Criteria	WgtAvgF-score	CPUTimeTesting	CPUTimeTraining	Consistency	Weights
WgtAvgF-score	1	8	9	7	0.70
CPUTimeTesting	0.13	1	3	1/2	0.09
CPUTimeTraining	0.11	0.33	1	1/5	0.04
Consistency	0.142857143	2.00	5	1	0.16
				CI:0.050	**1.00**

(c) DM#2 relative weighting

Criteria/Criteria	WgtAvgF-score	CPUTimeTesting	CPUTimeTraining	Consistency	Weights
WgtAvgF-score	1	7	9	5	0.68
CPUTimeTesting	0.14	1	2	1	0.12
CPUTimeTraining	0.11	0.50	1	1/3	0.06
Consistency	0.2	1.00	3	1	0.14
				CI:0.012	**1.00**

(d) DM#3 relative weighting

Criteria/Criteria	WgtAvgF-score	CPUTimeTesting	CPUTimeTraining	Consistency	Weights
WgtAvgF-score	1	7	8	6	0.68
CPUTimeTesting	0.14	1	2	1/2	0.10
CPUTimeTraining	0.13	0.50	1	1/3	0.06
Consistency	0.17	2.00	3.00	1	0.16
				CI:0.021	**1.00**

(e) DM#4 relative weighting

Criteria/Criteria	WgtAvgF-score	CPUTimeTesting	CPUTimeTraining	Consistency	Weights
WgtAvgF-score	1	8	9	8	0.71
CPUTimeTesting	0.13	1	4	1	0.12
CPUTimeTraining	0.11	0.25	1	1/6	0.04
Consistency	0.13	1.00	6	1	0.13
				CI:0.073	**1.00**

(f) Criteria weights based on group decision making

DM Decision Weights	0.49	0.21	0.23	0.08	
Criteria\DM	DM#1	DM#2	DM#3	DM#4	Weight
WgtAvgF-score	0.70	0.68	0.68	0.71	0.70
CPUTimeTesting	0.09	0.12	0.10	0.12	0.10
CPUTimeTraining	0.04	0.06	0.06	0.04	0.05
Consistency	0.16	0.14	0.16	0.13	0.15
					1.00

Table 9
A partial list of the average standard deviation (average consistency) of the classifiers on the ADA_ Agnostic dataset.

Algorithms	Wgt.Avg.F-score (Stdev)	CPUTimeTraining (Stdev)	CPUTimeTesting (Stdev)	Average (Stdev) - Consistency
bayes.BayesNet	0.018	0.015	0.005	0.013
bayes.NaiveBayes	0.017	0.006	0.008	0.010
bayes.NaiveBayesUpdateable	0.017	0.007	0.008	0.011
functions.Logistic	0.015	0.019	0.002	0.012
...	...	...	...	...
meta.Vote	0.017	0.010	0.000	0.009

using Eq. (11),

$$\text{Consistency}_{a\in A} = \frac{\sum_{i=1}^{m} \text{Stdev}_i}{m} \tag{11}$$

where, *a* represents an algorithm belonging to the algorithm space *A* and *m* represents the number of measurable metrics (3 in this case). For simplicity purpose, in this paper, we use the concept consistency instead of the average consistency. The consistency scores for a partial list of the algorithms are shown in Table 9 (last column).

Step 6: Testing significance of performance results

For checking the statistical significance of the algorithms performance results, procedure 4 is used, whose results are shown in Table 10, column 2–4. In this table, the reference classifiers are marked by bold faced keyword "**ref**" and the statistically poor results are marked with symbol "*" . The results, in these three

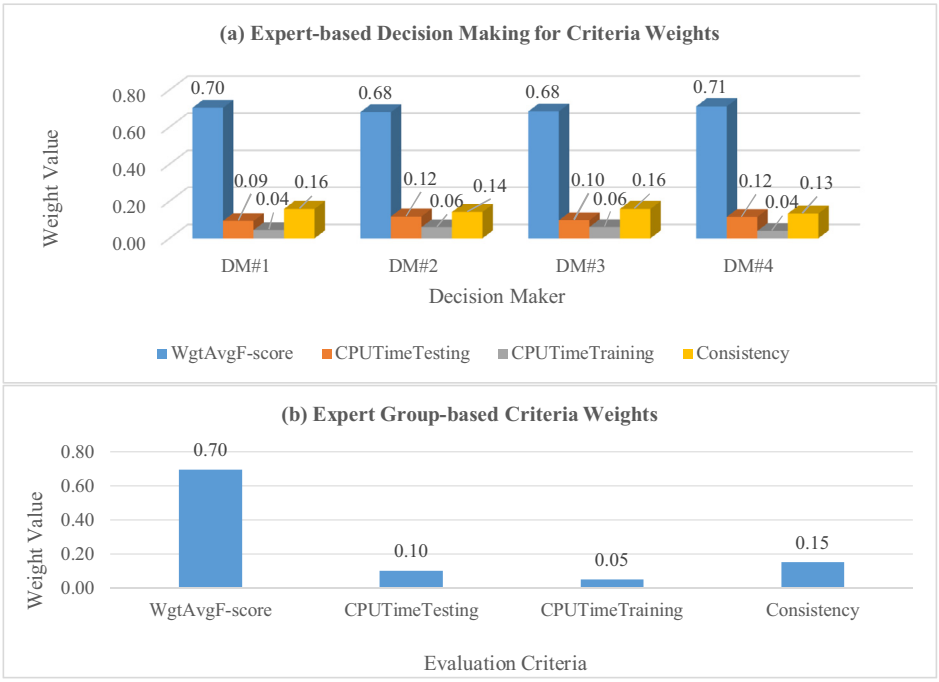

Fig. 5. Weight for the selected criteria, estimated using group-based crisp analytic hierarchy process.

Table 10
Performance results of classification algorithms on ADA_Agnostic dataset and their ranking with respect to relative distance from the ideal algorithm.

	Constraints							
Algorithms	Max F-score	Min CPUTimeTraining	Min CPUTimeTesting	Min Consistency	$PIS^{=}$	NIS^{-}	RC	Ranking
bayes.BayesNet	0.78*	0.027*	0.002	0.013	0.00906	0.03830	**0.80874**	26
bayes.NaiveBayes*	0.825*	0.013*	0.008*	0.010	0.00264	0.04180	**0.94068**	19
bayes.NaiveBayesUpdateable*	0.825*	0.011*	0.01*	0.011	0.00272	0.04171	**0.93882**	20
functions.Logistic	0.836	0.229*	0.000	0.012	0.00088	0.04317	**0.97995**	4
functions.RBFNetwork	0.733*	0.232*	0.004	0.043	0.01593	0.03492	**0.68672**	29
functions.SMO	0.830	1.99*	**(ref) 0.000**	0.041	0.00181	0.04239	**0.95905**	12
misc.HyperPipes	0.66*	**(ref) 0.001**	0.000	0.005	0.02658	0.03309	**0.55457**	32
misc.VFI	0.716*	0.008*	0.004	0.012	0.01841	0.03433	**0.65097**	31
rules.ConjunctiveRule	0.645*	0.043*	0.000	0.006	0.02877	0.03301	**0.53432**	35
rules.DecisionTable	0.829	1.086*	0.000	0.043	0.00195	0.04231	**0.95597**	14
rules.DTNB	0.832	88.16*	0.004	2.611	0.02792	0.03234	**0.53668**	33
rules.JRip	0.825*	0.648*	0.000	0.067	0.00257	0.04180	**0.94203**	18
rules.OneR	0.739*	0.014*	0.000	0.007	0.01504	0.03574	**0.70380**	28
rules.PART	0.819*	1.161*	0.001	0.057	0.00341	0.04126	**0.92367**	23
rules.Ridor	0.795*	0.453*	0.000	0.034	0.00687	0.03942	**0.85156**	24
rules.ZeroR	0.645*	0.000	0.000	0.001	0.02877	0.03305	**0.53463**	34
trees.BFTree	0.838	0.79*	0.000	0.024	0.00063	0.04328	**0.98557**	**2**
trees.FT	0.827	1.38*	0.161*	0.044	0.01790	0.03819	**0.68088**	30
trees.J48	0.828	0.221*	0.000	0.014	0.00205	0.04241	**0.95392**	15
trees.J48graft	0.829	0.29*	0.000	0.014	0.00190	0.04251	**0.95715**	13
trees.LADTree	0.833	1.676*	0.000	0.020	0.00134	0.04281	**0.96967**	10
trees.RandomForest	0.837	2.304*	0.022*	0.022	0.00255	0.04223	**0.94299**	17
trees.RandomTree	0.791*	0.028*	0.001	0.009	0.00745	0.03923	**0.84041**	25
trees.REPTree	0.835	0.084*	0.000	0.012	0.00103	0.04308	**0.97669**	7
trees.SimpleCart	0.836	0.713*	0.000	0.021	0.00090	0.04311	**0.97950**	5
meta.AdaBoostM1	0.822*	1.074*	0.001	0.021	0.00293	0.04176	**0.93440**	21
meta.Bagging	**(ref) 0.842**	0.753*	0.000	0.013	0.00014	0.04373	**0.99681**	**1**
meta.Dagging*	0.824*	0.013*	0.107*	0.010	0.01209	0.03861	**0.76154**	27
meta.END	0.828	0.215*	0.003	0.013	0.00207	0.04228	**0.95323**	16
meta.FilteredClassifier	0.832	0.065*	0.000	0.009	0.00146	0.04282	**0.96697**	11
meta.LogitBoost	0.835	1.948*	0.002	0.058	0.00121	0.04267	**0.97245**	9
meta.RacedIncr.LogitBoost	0.82*	0.062*	0.001	0.012	0.00322	0.04166	**0.92833**	22
meta.RandomSubSpace	0.837	0.412*	0.001	0.012	0.00075	0.04322	**0.98299**	**3**
meta.Stacking	0.834	0.724*	0.001	0.014	0.00118	0.04292	**0.97318**	8
meta.Vote	0.835	0.076*	0.000	0.009	0.00103	0.04310	**0.97676**	6
Relative Weights	**0.69520**	**0.05067**	**0.10097**	**0.15315**				
Positive Ideal Solution (PIS)	**0.12296**	**0.00874**	**0.01776**	**0.02647**				
Negative Ideal Solution (NIS)	**0.09419**	**0.00000**	**0.00000**	**0.00000**				

columns, with no symbol mentioned, are either same in performance or cannot be decided surely.

Step 7: Algorithmic fitness evaluation

The fitness function is performed on the labelled significant matrix of the algorithms results, which are marked as significant, non-significant and equally significant. In our proposed fitness evaluation function, described by Eq. (8), the non-significant algorithms are identified and are either filter out and dropped from the next step of evaluation or leaved as they are but not considered, when final selection is made from the ranked list of algorithms. Applying the fitness function, the algorithms bayes.NaiveBayes, bayes.NaiveBayesUpdateable, and meta.Dagging are identified as significantly poor on ADA_Agnostic dataset (see Table 10). The results of Eq. (8), for all the datasets, are summarized in Table 14.

Step 8: Ranking algorithms

To generate recommended ranking, Procedure 5 is applied on the performance matrix, Table 10, columns 2–5) with the specification of local constraints (i.e., Max and Min) and global constraints (i.e., consistency).

The relative closeness score (RC) (8th column) is produced for which the corresponding ranking is generated in the 9th column. This column is the recommended ranking for the algorithms. According to this ranking, meta.Dagging, trees.BFTree and meta.RandomSubSpace are ranked first, second, and third, respectively, on the ADA_Agnostic dataset. For evaluation of these results, an evaluation criteria and methodology is used, which is described in the next section.

5. Experiments and evaluation

5.1. Classifiers and datasets

We performed the experiments on 35 most commonly used multi-class classification algorithms, shown in Table 4, which are implemented in Weka machine learning library (Bouckaert et al., 2010). These algorithms belong to six heterogeneous families' of classifiers including: probabilistic learners, functions-based learners, decision trees learners, rules-based learners, meta-learners, and miscellaneous learners. The meta-classifiers, i.e., Adaboost M1, Randomspace, and Voting are used with REPTree as the base classifier. Similarly, Dagging and Stacking are used with Naïve Bayes as the base classifier. The rest of algorithms are used with Weka default parameters. Similarly, 15 classification datasets,[2] shown in Table 5, from UCI machine learning repository (Lichman, 2013) and OpenML repositories (Van Rijn et al., 2013) are used.

5.2. Evaluation methodology and criteria

To empirically evaluate the recommended ranking, the follows three steps methodology (Brazdil et al., 2003) is used, which is pictorially depicted in Fig. 6.

i. build a recommended ranking for a dataset *d* using the proposed AMD method
ii. build an ideal ranking for dataset *d*, and
iii. measure the agreement score between the two rankings

In step (i), the recommended ranking is obtained from the relative closeness score, which is computed using the proposed AMD method. In step (ii), the ideal ranking (IR) are constructed by applying ranking operation to the average score of algorithms performances, obtained by taking average of the weighted sum of normalized performance results of all the algorithms, *A*, on dataset *d*. We proposed the weighted sum average multi-criteria ideal ranking method (WAMR), described in Eqs. 12 and 13, where the steps performed follow the sequence: (a) performance results for each metric are estimated (i,e., s_{ij} is produced) using 10×10-fold CV, (b) normalized performance (i.e., NS_{ij}) is estimated using Eq. (13), (c) weighted performance, i.e., $W_j{}^*NS_{ij}$ is computed, (d) weighted sum, i.e., $\sum_{j=1}^{m}(W_j * NS_{ij})$, results are generated for all the metrics, (e) average of the weighted sum score is taken, and finally (f) ranks are generated. This process is described as follow,

$$IR = \text{rank}\left(\frac{\sum_{j=1}^{m}\left(W_j * NS_{ij}\right)}{m}\right), \tag{12}$$

where, W_j is weight vector of evaluation metrics, *E, m* is the number of evaluation metrics and NS_{ij} is the normalized performance value of the i^{th}algorithm for j^{th}evaluation metric, computed using Eq. (13),

$$NS_{ij} = \frac{s_{ij}}{\sqrt{\sum_{i=1}^{n} {s_{ij}}^2}}, \tag{13}$$

where, $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.

The rank operation of Eq. (12) is described in Algorithm 1. Similarly, in Eq. (13), the value s_{ij} is the performance of i^{th}algorithm for j^{th} evaluation metric, obtained using 10×10-fold cross-validation strategy (CV). Moreover the variables *n* and *m* represents number of algorithms and number of evaluation metrics, respectively.

In literature, different methods are used to compute ideal ranking, such as N-orderings, average correlation (AC) and average weighted correlation (AWC) (Brazdil & Soares, 2000; Soares, Brazdil, & Costa, 2000). In *N* orderings method (Brazdil et al., 2003), first 10-fold CV results are generated for all the algorithms on a single dataset and a pair-wise comparison using statistical significance tests is performed. The algorithms are ordered based on their significance results score. In the average correlation method, ranks are computed for each fold of the 10-fold CV results which are then averaged to get the ideal rank. All the algorithms are arranged based on their average correlation score. Similarly, in the AWC method, weights are assigned to the ranks of individual folds and are then averaged together for get the final ranks.

The motivation for proposing the new ideal ranking generation method, so called WAMR, is that it is designed for multiple-criteria rather than single criterion, where the following essential steps take place prior to ideal ranks generation, such as normalizations of the criteria values, weighting the normalized value for uniformity with the AMD method, aggregating the weighted performance of all the criteria and taking average to get global performance results.

In step (iii), the agreement score, which is the mean agreement between the recommended ranking and the ideal ranking, is measured using the Spearman's ranked correlation coefficient (Neave & Worthington, 1992; Neave & Worthington, 1988). The final value of the agreement is a measure of the quality of the recommended ranking and proves the level of correctness of the proposed AMD method. The formula for Spearman's rank correlation coefficient is shown in Eq. (14).

$$R_s = 1 - \frac{6 * \sum_{i=1}^{n}(IR_i - RR_i)^2}{n^3 - n}, \tag{14}$$

where, IR_i and RR_i are the ideal and recommended ranking of algorithm i, respectively, and n is the number of algorithms to compare. If the value of $R_s = 1$, it represents a perfect agreement and if $R_s = -1$, it represents a perfect disagreement. If $R_s = 0$, then both the ranks are not related. Significance of Spearman rank correlation can be determined by looking in the table of critical values for R_s with different levels of significance, i.e., α value (Zar, 1972).

[2] Some of the datasets are used with minor modifications by changing the type of the class label to nominal etc.

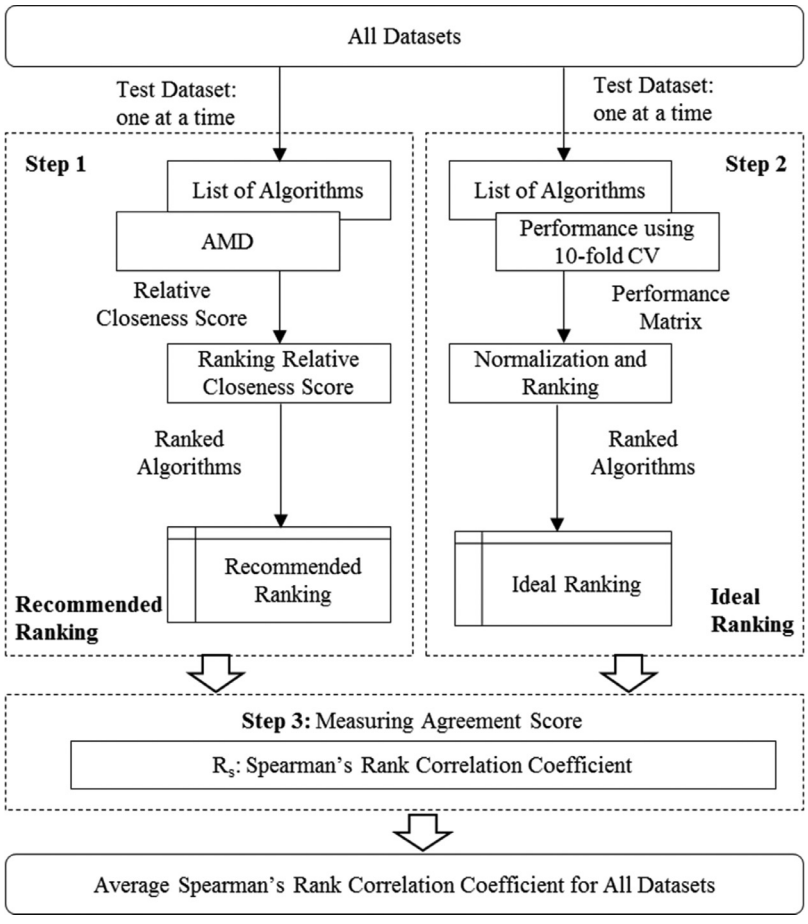

Fig. 6. Three steps evaluation methodology of recommended ranking against ideal ranking.

Similarly, the overall result for all the datasets is evaluated using the average Spearman rank correlation coefficient ($AvgR_s$). This is shown by Eq. (15),

$$AvgR_s = \frac{\sum_{i=1}^{d} R_s(d_i)}{d}, \quad (15)$$

where, $R_s(d_i)$ is the Spearman's rank correlation coefficient for dataset d_i and d is the total number of datasets.

5.3. Experiments and analysis of the results

In this section we perform a set of experiments and analyze the results from diverse perspective to validate the proposed AMD methodology. The set of experiments includes: (a) correctness check using average Spearman's correlation coefficient, (b) generalization power check using sensitivity and consistency, and (c) significance fitness evaluation.

5.3.1. Correctness: average spearman's rank correlation coefficient

To estimate correctness level of the proposed AMD, average Spearman's rank correlation coefficient is computed for all the datasets, using the AMD method, described in section 4.2. The average of recommended rankings for all the datasets is shown in Table 11. The weights used for generating the recommended ranking are: Wgt.Avg.F-score (0.69520), CPUTimeTraining (0.05067), CPUTimeTesting (0.10097), and Consistency (0.15315). In the second step, ideal rankings for all the datasets are generated by taking average of the weighted sum of the normalized values of these evaluation metrics. Finally, the R_s is computed using Eq. (14) and the $AvgR_s$ is calculated using Eq. (15).

Table 11
Average Spearman's rank correlation coefficient for fifteen classification datasets.

Dataset ID	Dataset Name	R_s
1	**abalone-3class**	**0.988**
2	**rabe-148**	0.985
3	**acute-inflammations-nephr**	0.994
4	**ADA_Agnostic**	0.990
5	**ADA_Prior**	0.991
6	**adult-4000**	0.983
7	**adult-8000**	0.975
8	**aileron**	0.979
9	**analcatdata-AIDS**	0.983
10	**analcatdata-apnea2**	0.932
11	**analcatdata-apnea2**	0.963
12	**analcatdata-asbestos**	0.973
13	**analcatdata-authorship**	0.999
14	**analcatdata-bankruptcy**	0.983
15	**analcatdata-birthday**	0.969
	$AvgR_s$	**0.979**

The $AvgR_s$ value is very close to 1, which demonstrates correctness of the proposed AMD methodology. It accurately ranks the algorithms and thus assists experts in the selection of accurate algorithms under the specified criteria. The statistical significance test of Spearman's rank correlation coefficient shows that the value 0.979 is statistically significant at the level of 0.001, with (35–2 = 33) degree of freedom (df), because the average correlation value 0.979 is far greater than the critical value of the correlation, i.e., 0.554.

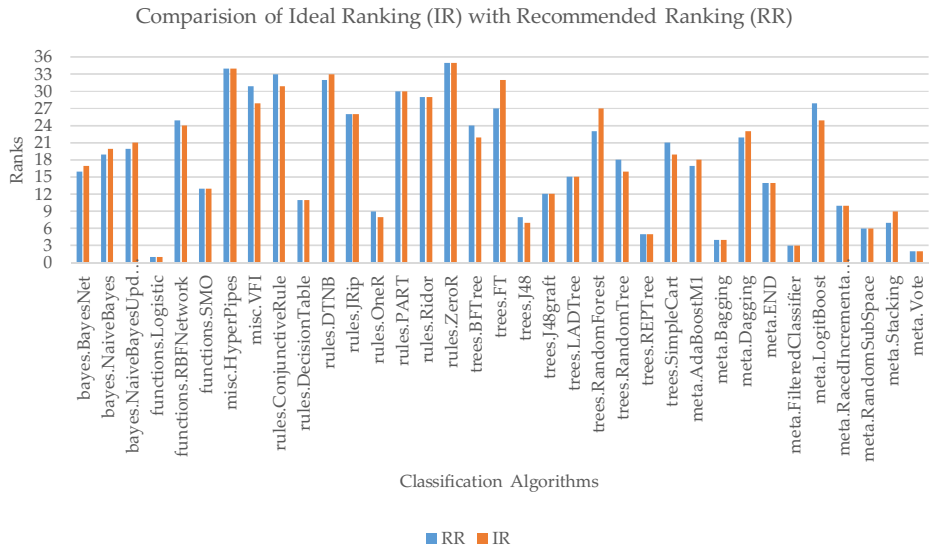

Fig. 7. Comparison of the recommended ranking (RR) and ideal ranking (IR) for the abalone-3class dataset.

Table 12
Spearman's rank correlation coefficient computation for abalone-3class dataset.

Algorithms	RR	IR	(IR-RR)	$(IR-RR)^2$
bayes.BayesNet	16	17	1	1
bayes.NaiveBayes	19	20	1	1
bayes.NaiveBayesUpdateable	20	21	1	1
functions.Logistic	1	1	0	0
functions.RBFNetwork	25	24	−1	1
functions.SMO	13	13	0	0
misc.HyperPipes	34	34	0	0
misc.VFI	31	28	−3	9
rules.ConjunctiveRule	33	31	−2	4
rules.DecisionTable	11	11	0	0
rules.DTNB	32	33	1	1
rules.JRip	26	26	0	0
rules.OneR	9	8	−1	1
rules.PART	30	30	0	0
rules.Ridor	29	29	0	0
rules.ZeroR	35	35	0	0
trees.BFTree	24	22	−2	4
trees.FT	27	32	5	25
trees.J48	8	7	−1	1
trees.J48graft	12	12	0	0
trees.LADTree	15	15	0	0
trees.RandomForest	23	27	4	16
trees.RandomTree	18	16	−2	4
trees.REPTree	5	5	0	0
trees.SimpleCart	21	19	−2	4
meta.AdaBoostM1	17	18	1	1
meta.Bagging	4	4	0	0
meta.Dagging	22	23	1	1
meta.END	14	14	0	0
meta.FilteredClassifier	3	3	0	0
meta.LogitBoost	28	25	−3	9
meta.RacedIncrementalLogitBoost	10	10	0	0
meta.RandomSubSpace	6	6	0	0
meta.Stacking	7	9	2	4
meta.Vote	2	2	0	0
			$\sum_{i=1}^{n}(IR_i - RR_i)^2$	**88**
			$\mathbf{R_s} = 1 - \frac{6*\sum_{i=1}^{n}(IR_i-RR_i)^2}{n^3-n}$	**0.988**

To show the process of calculating R_s, results for the abalone-3class dataset are shown in Table 12.

The interpretation of R_s result is the same as we did for the $AvgR_s$. A pictorial view of the results of recommended and ideal ranking for the abalone-3class dataset is shown in Fig. 7.

This figure shows that the recommended ranking of AMD is closed to the ideal ranking.

5.3.2. Generalization of AMD: sensitivity and consistency analysis

In multi-criteria decision making, the choice and number or weights of the criteria affect the final recommended ranking (Lopez & Carlos, 2005; Opricovic & Tzeng, 2004; Saaty, 2006; Zavadskas, et al., 2006). It has been demonstrated that the choice of criteria or the change in weights transforms the final recommended ranking (Saaty, 2006; Zavadskas, et al., 2006). In majority of the algorithms ranking cases, it is hard for the decision makers to agree on the final ranks generated by a ranking method and is therefore required to perform sensitivity analysis (Goicoechea, Hansen, & Duckstein, 1982; Insua & French, 1991). The significant results of the ranking method under varying parameters demonstrates generalization power of a ranking method. In our case, the scope of sensitivity analysis is limited to the change in relative weights of criteria. We change the weight of each criterion, i.e., Wgt.Avg.F-score, CPUTimeTesting, CPUTimeTraining and Consistency, one at a time, and compute the Spearman's rank correlation coefficient value to see how the proposed AMD behaves with the changed weights. For the criteria Wgt.Avg.F-score, CPUTimeTesting, CPUTimeTraining and Consistency, the R_s results generated by the proposed AMD methodology using weights (0.70,0.05,0.10,0.15), (0.05,0.70,0.10,0.15), (0.05,0.10,0.70, 0.15) and (0.05,0.10,0.15,0.70) are shown in Table 13.

In Table 13, the R_s value for each set of the weights of the evaluation criteria is computed (using Eq. (14)) and evaluated in the same way as in previous section. However, in this case, the ideal ranking is computed for the individual criteria and compared with the recommended ranking. In each set of the weights, more preference, i.e., weight 0.70, is given to only one criterion and thus algorithms are preferred with respect to that criterion, which is natural. In Table 14, the R_s values shown in bold demonstrate negative/weak correlation with respect to the ideal ranking. The $AvgR_s$ (for all datasets, computed using Eq. (15)) in all the cases are positively correlated to ideal ranking, which demonstrate that the AMD is a generalized and consistent methodology that performs well in varying conditions. The statistical significance test of Spearman's rank correlation coefficient for the Wgt.Avg.F-score shows that the correlation value 0.523 is statistically significant at the level of 0.005–0.002, with (35–2 = 33) degree of freedom (df), because it is

Table 13
Sensitivity analysis of the proposed accurate multi-metric decision making methodology (AMD) methodology with varying criteria weights.

Dataset ID	Dataset\Weights, k=35	Sensitivity Analysis			
		R_s Wgt.Avg F-score (0.70,0.05,0.10,0.15)	R_s CPUTimeTraining (0.05,0.70,0.10,0.15)	R_s CPUTimeTesting (0.05,0.10,0.70, 0.15)	R_s Consistency (0.05,0.10,0.15,0.70)
1	**abalone-3class**	0.454	0.913	0.523	0.999
2	**rabe-148**	0.904	0.758	0.500	0.992
3	**acute-inflammations-nephr**	0.858	0.798	0.501	0.979
4	**ADA_Agnostic**	0.880	0.368	0.819	0.433
5	**ADA_Prior**	0.295	0.943	0.565	0.985
6	**adult-4000**	0.276	0.890	0.599	0.979
7	**adult-8000**	0.488	0.792	0.670	0.943
8	**aileron**	0.946	0.223	0.806	0.563
9	**analcatdata-AIDS**	0.654	0.766	0.500	0.995
10	**analcatdata-apnea2**	0.107	0.844	0.652	0.986
11	**analcatdata-apnea2**	0.158	0.936	0.618	0.972
12	**analcatdata-asbestos**	0.508	0.838	0.500	0.999
13	**analcatdata-authorship**	0.880	−0.265	0.738	−0.074
14	**analcatdata-bankruptcy**	0.945	0.863	0.543	0.998
15	**analcatdata-birthday**	−0.506	0.777	0.618	0.990
	AvgR_s	**0.523**	**0.696**	**0.610**	**0.849**

Table 14
Analysis of significantly poor performing algorithms produced by significant fitness function.

Algorithm	ADA_Agnostic (rank)	ADA_Prior (rank)	adult-4000 (rank)	adult-8000 (rank)	aileron (rank)	analcatdata-authorship (rank)
bayes.BayesNet*	26	4	2	7	27	**4**
bayes.NaiveBayes*	**19**	11	12	21	**30**	**7**
bayes.NaiveBayesUpdateable*	**20**	10	15	20	**31**	8
trees.FT*	30	**32**	**32**	**32**	25	2
trees.RandomForest*	17	**25**	23	**24**	**17**	6
meta.Dagging*	**27**	18	21	26	**32**	**30**

greater than the critical value 0.482 for R_s. Similar interpretations can be made for the rest of criteria.

5.3.3. Significance fitness evaluation

The results of Eq. (8), which identifies significantly poor algorithms for the datasets are shown in Table 14.

These results show that the classification algorithms bayes.BayesNet and bayes.NaiveBayes get higher ranks (4 and 7) on the analcatdata-authorship, however their performance on this dataset does not remain significant for all the criteria. Hence, prior applying the ranking process, the significance fitness function is required to execute to filter out insignificant algorithms from the competition. The values presented in bold represent the rank of algorithms on the dataset shown in the columns.

5.4. Comparison with existing methods

In this section, we compare the results of AMD methodology with two well-known methods: adjusted ratio of ratios (ARR) (Brazdil et al., 2003) and automatic recommendation of classification algorithms based on data set characteristics, abbreviated as PAlg (Song et al., 2012). These methods evaluate and rank classification algorithms on the basis of accuracy and time.

The equation of ARR ranking methodology (Brazdil et al., 2003) is shown in Eq. (16),

$$\mathrm{ARR} = \frac{\frac{\mathrm{SR}_{ap}^{di}}{\mathrm{SR}_{aq}^{di}}}{1+\propto *\log\left(\frac{\mathrm{T}_{ap}^{di}}{\mathrm{T}_{aq}^{di}}\right)}. \tag{16}$$

The accuracy is represented as the ratio of success rates of algorithm *ap* to algorithm *aq* on a dataset *d* as the numerator of the ARR. The time, which is the total of training and execution times, which is represented as a ratio of times is used as the denominator. To enforce preferences on the criteria, parameter $\propto$ is introduced with its value $\propto = 0.1$, 1, and 10 to specify 10% preference of the accuracy on time, equal preferences of both the accuracy and time and 10% preference of time over the accuracy, respectively.

In the algorithm selection article (Song et al., 2012), the performances of algorithms are evaluated using Eq. (17), where accuracy and total time are directly used instead of their ratios. The setting for $\propto$ is the same as that of the ARR method.

$$\mathrm{P}_{\mathrm{Alg}} = \frac{\mathrm{Accuracy}_{\mathrm{Alg.\ D}}}{1+\propto *\log\left(\mathrm{RTime}_{\mathrm{Alg.\ D}}\right)} \tag{17}$$

As these two methods are only based on accuracy and execution and training time (T/RTime), therefore to create a fair comparison, we formulate our proposed criteria accordingly. We picked Wgt.Avg.F-score, CPUTimeTraining and CPUTimeTesting and omitted the Consistency criterion. The values of CPUTimeTraining and CPUTimeTesting are averaged to get the uniform value for T/RTime, used in Eqs. 16 and 17, respectively. For simplicity, we performed experiments only for $\propto = 0.1$ with three different sittings, such as ranking for all 35 algorithms (k = 35), ranking for only top 5 algorithms (k = 5) and ranking for top 3 algorithms (k = 3). The weight for accuracy (Wgt.Avg.F-score) and T/RTime, in our proposed AMD method, were taken as 0.55 and 0.45, which are compliant to $\propto = 0.1$.

We performed comparison experiments on the same 15 datasets and the results generated are shown in Table 15 and Fig. 8(a-c).

The performance results of the proposed AMD method are significantly better than the results of the PAlg and ARR under the three different setup: all (k = 35) algorithms, top k = 5 algorithms and top k = 3 algorithms. For the proposed method, the statistical significance test of Spearman's rank correlation coefficient

Table 15
Comparison of the proposed method with state-of-the-art methods.

Id	Dataset	AMD			PAlg			ARR		
		R_s with $\alpha=0.1$ (Wgt.F-Score= 0.55, Rtime= 0.45)			R_s with $\alpha=0.1$ (Wgt.F-Score= 0.55, Rtime= 0.45)			R_s with $\alpha=0.1$ (Wgt.F-Score= 0.55, Rtime= 0.45)		
		k= 35	k= 5	k= 3	k= 35	k= 5	k= 3	k= 35	k= 5	k= 3
1	**abalone-3class**	0.9720	0.9978	1.0000	0.8473	0.9926	0.9944	0.6012	0.9769	0.9842
2	**rabe-148**	1.0000	1.0000	1.0000	0.9900	1.0000	1.0000	0.5200	0.9450	0.9520
3	**acute-inflammations-nephr**	1.0000	1.0000	1.0000	0.9641	1.0000	1.0000	0.5199	0.9940	0.9908
4	**ADA_Agnostic**	0.9852	0.9974	0.9989	**0.3187***	0.9171	0.9521	**0.2696***	0.8752	0.8865
5	**ADA_Prior**	0.9899	0.9992	0.9993	0.8081	0.9699	0.9863	0.4966	0.8975	0.9515
6	**adult-4000**	0.9922	1.0000	1.0000	0.8314	0.9715	0.9851	**0.3482***	0.8641	0.9342
7	**adult-8000**	0.9824	0.9997	1.0000	0.7028	0.9556	0.9697	**0.2529***	0.8871	0.9158
8	**aileron**	0.9882	0.9986	0.9997	0.7541	0.9724	0.9869	0.5646	0.9956	0.9987
9	**analcatdata-AIDS**	0.9801	0.9985	0.9987	0.9908	1.0000	1.0000	0.5039	0.8929	0.9399
10	**analcatdata-apnea2**	0.9916	1.0000	1.0000	0.9748	0.9987	1.0000	0.5162	0.9799	0.9910
11	**analcatdata-apnea2**	0.9955	1.0000	1.0000	0.9501	1.0000	1.0000	0.5292	0.9636	0.9854
12	**analcatdata-asbestos**	0.9711	1.0000	1.0000	0.9706	1.0000	1.0000	0.4764	0.9359	0.9410
13	**analcatdata-authorship**	0.9980	0.9992	0.9993	0.5070	0.9164	0.9637	**0.2524***	0.7271	0.7921
14	**analcatdata-bankruptcy**	0.9975	1.0000	1.0000	0.9756	0.9997	1.0000	0.4574	0.8694	0.9185
15	**analcatdata-birthday**	0.9854	1.0000	1.0000	0.9728	0.9977	1.0000	0.5298	0.9107	0.9567
	AvgR_s	**0.9886**	**0.9993**	**0.9997**	**0.8372**	**0.9794**	**0.9892**	**0.4559**	**0.9143**	**0.9426**

shows that the correlation values, $R_s = 0.9886$, $R_s = 0.9993$, and $R_s = 0.9997$, for k = 35, k = 5 and k = 3, respectively, are statistically significant at the level of 0.001, with (35–2 = 33) degree of freedom (df). Similar interpretation can be made for PAlg method. However, this method produces ranks for the algorithms (with k = 35) on the ADA_Agnostic dataset, which is statistically insignificant with respect to the ideal ranking. Similarly, the results of ARR method are significantly poor as compared to the proposed methods under all the conditions of k = 35, k = 5 and k = 3. Under the setting, k = 35, the ARR results are significant with respect to the critical value of R_s at the level of 0.01–0.005 with 33°degree of freedom. Using this method, four datasets, represented with '*' has the ranks which are significantly poor and not correlated to the ideal ranking.

Fig. 8 shows that AMD performs significantly better as compared to the state-of-the art methods under all the settings of top k = 35, top k = 5 and top k = 3 algorithms.

5.4.1. Statistical significance test for comparison of ranking methods

To test whether the results produced by AMD methodology are statistically significant or not as compared to the comparing methods, we performed Friedman's test (Friedman, 1940). First we set the following hypotheses:

- H0: There is no difference in the mean average correlation coefficients, AvgR_s, for the three ranking methods (AMD, ARR and PAlg with all the datasets).
- H1: There are some differences in the mean average correlation coefficients, AvgR_s, for the three ranking methods.

For illustrating Friedman's test process and the corresponding results, we compare the three ranking methods (i.e., j = 1, 2, 3) on the 15 datasets. All the steps are shown in Table 16(a-c). The steps are performed as follows: (a) rank the correlation coefficients for each dataset, i.e., RR, (b) calculate the mean rank for each method, i.e., $RR_j = \sum_j RR_j/n$, where n is the number of datasets (15 in this case), (c) calculate the overall mean rank (mR) across all the methods, i.e., $mR = (m+1)/2 = 2$, where m is the number of methods to compare (m = 3 in this case), (d) calculate sum of the squared differences of mean rank for each method and the overall mean rank, i.e., $S = \sum_j (RR_j - mR)^2$, and (e) calculate Friedman's statistic, $M = (12nS)/(k(k+1))$.

The calculation of these steps is shown in Table 16(a-c), for all the fifteen datasets, and the results are summarized in Table 17. In the example of Table 16, where $n = 15$ and $m = 3$, the critical value C is 10.99 for a confidence level of 95%. The Friedman's test values (M) for k= 35, k= 5 and k= 3 > C(10.99) is true, therefore the null hypothesis is rejected, which means 0.083 s that the average performance of the three methods is not similar and hence AMD is significantly better than state-of-the-art methods in comparison.

6. Conclusions and future work

In this paper, Firstly, we introduced the concepts of algorithms' quality meta-metrics (QMM), describing physical meaning of the evaluation criteria, and developed a classification model with the help of extensive literature to assist experts in the selection of suitable evaluation criteria for comparison of the classifiers. Motivated from the experts' consensus-based nominal grouped technique, we proposed a standard, experts group-based, method for the selection of suitable evaluation metrics from a large set of evaluation metrics and satisfying the constraints defined by the users/experts at the goal and objectives definition time.

Secondly, we estimated consistent relative weights for the evaluation metrics using the expert group-based decision making using the analytical hierarchy process. The experts' preferences on the criteria are quantified effectively and the weights are checked for consistency. We have analyzed performance of classification algorithm using statistical significance test and our proposed fitness function to filter out algorithms, which are statistically insignificant on all the evaluation criteria. For ranking the algorithms, we computed the relative closeness value of all the algorithms with respect to the ideal ranking, using the AHP-based estimated weights and local and global constraints on the evaluation criteria. The local constraints on criteria are used to encourage and discourage some of the criteria based on the categorization as cost and benefit criteria. The global constraints are imposed in the form of consistency measure that takes the standard deviation of all the criteria and consider an aggregate value to evaluate the quality of the selected/recommended algorithm.

Finally, we evaluated our AMD methodology by conducting a series of experiments on 15 different classification datasets using 35 classification algorithms. We compared the results of AMD with two stat-of-the-art methods. Results shows that the proposed AMD methodology performing significantly and produce good results.

In future, we intend to combine meta-learning with the multi-criteria decision making for automatically selecting a suitable algorithm for a new dataset. We also intend to use multi-view

Table 16
Steps of the Friedman's test for compare ranking methods for statistical significance.

(a) Friedman's test steps for comparing ranking methods with k= 35

Dataset	**d_1**		**d_2**		**d_3**		**d_4**		**d_5**		**d_6**		**d_7**		**d_8**		**d_9**		**d_{10}**		**d_{11}**		**d_{12}**		**d_{13}**		**d_{14}**		**d_{15}**			
Method\Rs	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**R_s**	**RR**	**RR_j**	**$(RR_j\text{-}mR)^2$**
AMD	0.9720	**1.0**	1.0000	**1.0**	1.0000	**1.0**	0.9852	**1.0**	0.9899	**1.0**	0.9922	**1.0**	0.9824	**1.0**	0.9882	**1.0**	0.9801	**2.0**	0.9916	**1.0**	0.9955	**1.0**	0.9711	**1.0**	0.9980	**1.0**	0.9975	**1.0**	0.9854	**1.0**	1.1	0.871111111
PAlg	0.8473	**2.0**	0.9900	**2.0**	0.9641	**2.0**	0.3187	**2.0**	0.8081	**2.0**	0.8314	**2.0**	0.7028	**2.0**	0.7541	**2.0**	0.9908	**1.0**	0.9748	**2.0**	0.9501	**2.0**	0.9706	**2.0**	0.5070	**2.0**	0.9756	**2.0**	0.9728	**2.0**	1.9	0.004444444
ARR	0.6012	**3.0**	0.5200	**3.0**	0.5199	**3.0**	0.2696	**3.0**	0.4966	**3.0**	0.3482	**3.0**	0.2529	**3.0**	0.5646	**3.0**	0.5039	**3.0**	0.5162	**3.0**	0.5292	**3.0**	0.4764	**3.0**	0.2524	**3.0**	0.4574	**3.0**	0.5298	**3.0**	3.0	1
																															S	**1.875555556**

(b) Friedman's test steps for comparing ranking methods with k= 5

Dataset	**d_1**		**d_2**		**d_3**		**d_4**		**d_5**		**d_6**		**d_7**		**d_8**		**d_9**		**d_{10}**		**d_{11}**		**d_{12}**		**d_{13}**		**d_{14}**		**d_{15}**			
Method\Rs	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	**RR_j**	**$(RR_j\text{-}mR)^2$**
AMD	0.9978	**1.0**	1.0000	**1.5**	1.0000	**1.5**	0.9974	**1.0**	0.9992	**1.0**	1.0000	**1.0**	0.9997	**1.0**	0.9986	**1.0**	0.9985	**2.0**	1.0000	**1.0**	1.0000	**1.5**	1.0000	**1.5**	0.9992	**1.0**	1.0000	**1.0**	1.0000	**1.0**	1.2	0.64
PAlg	0.9926	**2.0**	1.0000	**1.5**	1.0000	**1.5**	0.9171	**2.0**	0.9699	**2.0**	0.9715	**2.0**	0.9556	**2.0**	0.9724	**3.0**	1.0000	**1.0**	0.9987	**2.0**	1.0000	**1.5**	1.0000	**1.5**	0.9164	**2.0**	0.9997	**2.0**	0.9977	**2.0**	1.9	0.017777778
ARR	0.9769	**3.0**	0.9450	**3.0**	0.9940	**3.0**	0.8752	**3.0**	0.8975	**3.0**	0.8641	**3.0**	0.8871	**3.0**	0.9956	**2.0**	0.8929	**3.0**	0.9799	**3.0**	0.9636	**3.0**	0.9359	**3.0**	0.7271	**3.0**	0.8694	**3.0**	0.9107	**3.0**	2.9	0.871111111
																															S	**1.528888889**

(c) Friedman's test steps for comparing ranking methods with k= 3

Dataset	**d_1**		**d_2**		**d_3**		**d_4**		**d_5**		**d_6**		**d_7**		**d_8**		**d_9**		**d_{10}**		**d_{11}**		**d_{12}**		**d_{13}**		**d_{14}**		**d_{15}**			
Method\Rs	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	R_s	**RR**	**RR_j**	**$(RR_j\text{-}mR)^2$**
AMD	1.0000	**1.0**	1.0000	**1.5**	1.0000	**1.5**	0.9989	**1.0**	0.9993	**1.0**	1.0000	**1.0**	1.0000	**1.0**	0.9997	**1.0**	0.9987	**2.0**	1.0000	**1.5**	1.0000	**1.5**	1.0000	**1.5**	0.9993	**1.0**	1.0000	**1.5**	1.0000	**1.5**	1.3	0.49
PAlg	0.9944	**2.0**	1.0000	**1.5**	1.0000	**1.5**	0.9521	**2.0**	0.9863	**2.0**	0.9851	**2.0**	0.9697	**2.0**	0.9869	**3.0**	1.0000	**1.0**	1.0000	**1.5**	1.0000	**1.5**	1.0000	**1.5**	0.9637	**2.0**	1.0000	**1.5**	1.0000	**1.5**	1.8	0.054444444
ARR	0.9842	**3.0**	0.9520	**3.0**	0.9908	**3.0**	0.8865	**3.0**	0.9515	**3.0**	0.9342	**3.0**	0.9158	**3.0**	0.9987	**2.0**	0.9399	**3.0**	0.9910	**3.0**	0.9854	**3.0**	0.9410	**3.0**	0.7921	**3.0**	0.9185	**3.0**	0.9567	**3.0**	2.9	0.871111111
																															S	**1.415555556**

Table 17
Summary of the Friedman's test for compare ranking methods based on different criteria.

Friedman's Test	S	M	C	M vs. C	Interpretation
Top-K = 35	1.876	28.133	10.99	M > C	M(28.13) > C(10.99) → null hypothesis is rejected at the confidence level $\alpha = 0.001$
Top-K = 5	1.529	22.933	10.99	M > C	M(22.93) > C(10.99) → null hypothesis is rejected at the confidence level $\alpha = 0.001$
Top-K = 3	1.416	21.233	10.99	M > C	M(21.23) > C(10.99) → null hypothesis is rejected at the confidence level $\alpha = 0.001$

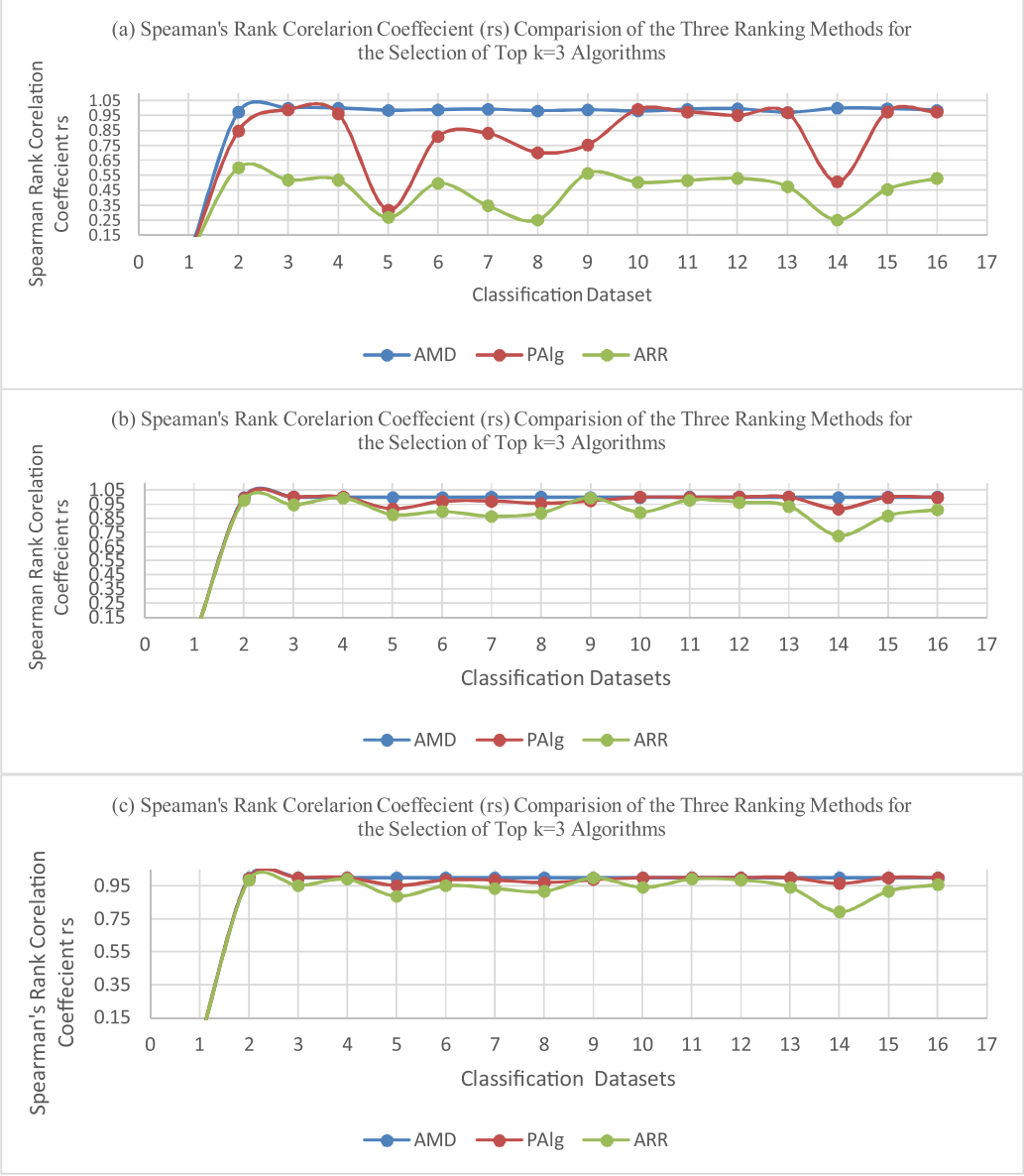

Fig. 8. Comparison of the proposed method with state-of-the art methods.

multi-label learning using the meta-features extracted from the multiple views of the datasets.

Author contributions

Rahman Ali conceived, designed and performed the experiments, analyzed the data and wrote the paper, while Sungyoung Lee and Tae Choong Chung provided the necessary advisory comments along and financial assistance.

Conflicts of interest

"The authors declare no conflict of interest."

Acknowledgements

This research was supported by the MSIP, Korea, under the G-ITRC support program (IITP-2015-R6812-15-0001) supervised by the IITP, by basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2011-0030079), by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) NRF-2014R1A2A2A01003914.

References

Aha, D. W. (1992). Generalizing from case studies: A case study. In *Proc. of the 9th international conference on machine learning* (pp. 1–10).

Alexandros, K., & Melanie, H. (2001). Model selection via meta-learning: A comparative study. *International Journal on Artificial Intelligence Tools, 10*, 525–554.

Ali, S., & Smith-Miles, K. A. (2006). A meta-learning approach to automatic kernel selection for support vector machines. *Neurocomputing, 70*, 173–186.

Ali, S., & Smith, K. A. (2006). On learning algorithm selection for classification. *Applied Soft Computing, 6*, 119–138.

Andersson, A., Davidsson, P., & Lindén, J. (1999). Measure-based classifier performance evaluation. *Pattern Recognition Letters, 20*, 1165–1173.

Berrer, H., Paterson, I., & Keller, J. r. (2000). Evaluation of machine-learning algorithm ranking advisors. In *Proceedings of the PKDD-2000 workshop on data-mining, decision support, meta-learning and ILP: forum for practical problem presentation and prospective solutions*. Citeseer.

Bouckaert, R. R., Frank, E., Hall, M. A., Holmes, G., Pfahringer, B., Reutemann, P., et al. (2010). WEKA—experiences with a java open-source project. *The Journal of Machine Learning Research, 11*, 2533–2541.

Bouyssou, D. (1990). Building criteria: A prerequisite for MCDA. In *Readings in multiple criteria decision aid* (pp. 58–80). Springer.

Brazdil, P. B., & Soares, C. (2000). A comparison of ranking methods for classification algorithm selection. In *Machine learning: ECML 2000* (pp. 63–75). Springer.

Brazdil, P. B., Soares, C., & da Costa, J. P. (2003). Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning, 50*, 251–277.

Brodley, C. E. (1993). Addressing the selective superiority problem: Automatic algorithm/model class selection. In *Proceedings of the tenth international conference on machine learning* (pp. 17–24).

Buciluǎ, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 535–541). ACM.

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on machine learning* (pp. 161–168). ACM.

Elazmeh, W., Japkowicz, N., & Matwin, S. (2006). *A framework for measuring classification difference with imbalance*. Menlo Park: AAAI press (technical report ws-06-06).

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27*, 861–874.

Ferri, C., Hernández-Orallo, J., & Modroiu, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters, 30*, 27–38.

Figueira, J., Greco, S., & Ehrgott, M. (2005). Multiple criteria decision analysis: state of the art surveys *(Vol. 78)*. Springer Science & Business Media.

Freitas, A. (1998). On objective measures of rule interestingness. *Second european symposium on principles of data mining & knowledge discovery*.

Freitas, A. A. (2004). A critical review of multi-objective optimization in data mining: A position paper. *ACM SIGKDD Explorations Newsletter, 6*, 77–86.

Freitas, A. A. (2006). Are we really discovering interesting knowledge from data. *Expert Update (the BCS-SGAI Magazine), 9*, 41–47.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics, 11*, 86–92.

Gaines, B. R. (1996). Transforming rules and trees into comprehensible knowledge structures. *Advances in Knowledge discovery and Data mining*, 205–226.

Gallagher, M., Hares, T., Spencer, J., Bradshaw, C., & Webb, I. (1993). The nominal group technique: A research tool for general practice. *Family Practice, 10*, 76–81.

Gama, J., & Brazdil, P. (1995). Characterization of classification algorithms. In *Progress in artificial intelligence* (pp. 189–200). Springer.

Garcia-Cascales, M. S., & Lamata, M. T. (2012). On rank reversal and TOPSIS method. *Mathematical and Computer Modelling, 56*, 123–132.

Gayen, A. K. (1951). The frequency distribution of the product-moment correlation coefficient in random samples of any size drawn from non-normal universes. *Biometrika, 38*, 219–247.

Goicoechea, A., Hansen, D. R., & Duckstein, L. (1982). *Multiobjective decision analysis with engineering and business applications*. Wiley Online Library.

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: concepts and techniques: Concepts and techniques*. Elsevier.

Hwang, C.-L., & Yoon, K. (2012). Multiple attribute decision making: Methods and applications a state-of-the-art survey *(Vol. 186)*. Springer Science & Business Media.

Insua, D. R., & French, S. (1991). A framework for sensitivity analysis in discrete multi-objective decision-making. *European journal of operational research, 54*, 176–190.

Kandhasamy, J. P., & Balamurali, S. (2015). Performance analysis of classifier models to predict diabetes mellitus. *Procedia Computer Science, 47*, 45–51.

Khanmohammadi, S., & Rezaeiahari, M. (2014). AHP based classification algorithm selection for clinical decision support system development. *Procedia Computer Science, 36*, 328–334.

Kou, G., Lu, Y., Peng, Y., & Shi, Y. (2012). Evaluation of classification algorithms using MCDM and rank correlation. *International Journal of Information Technology & Decision Making, 11*, 197–225.

Kumar, P., Sehgal, V. K., & Chauhan, D. S. (2012). A benchmark to select data mining based classification algorithms for business intelligence and decision support systems. *International Journal of Data Mining & Knowledge Management Process (IJDKP), 2*(5), 25–42.

Lavesson, N., Boeva, V., Tsiporkova, E., & Davidsson, P. (2014). A method for evaluation of learning components. *Automated Software Engineering, 21*, 41–63.

Lavesson, N., & Davidsson, P. (2007). Analysis of multi-criteria methods for algorithm and classifier evaluation. In *Proceedings of the 24th annual workshop of the Swedish artificial intelligence society* (pp. 11–22). Linkoping.

Lope, & Carlos, J. (2005). Multicriteria decision aid application to a student selection problem. *Pesquisa Operacional, 25*, 45–68.

Lichman, M. (2013). *UCI machine learning repository*. Irvine, CA: University of California, School of Information and Computer Science *http://archive.ics.uci.edu/ml* In.

Lim, T.-S., Loh, W.-Y., & Shih, Y.-S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning, 40*, 203–228.

Lindner, G., & Studer, R. (1999). AST: Support for algorithm selection with a CBR approach. In *Principles of data mining and knowledge discovery* (pp. 418–423). Springer.

Luo, G. *A review of automatic selection methods for machine learning algorithms and hyper-parameter values* Available at http://pages.cs.wisc.edu/~gangluo/automatic_selection_review.pdf.

Majumder, M. (2015). Multi criteria decision making. In *Impact of urbanization on water shortage in face of climatic aberrations* (pp. 35–47). Springer.

Neave, H., & Worthington, P. (1992). *Distribution-free tests*. London: Routledge.

Neave, H. R., & Worthington, P. L. (1988). *Distribution-free tests*. London: Unwin Hyman.

Opricovic, S., & Tzeng, G.-H. (2004). Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS. *European Journal of Operational Research, 156*, 445–455.

Perveen, S., Shahbaz, M., Guergachi, A., & Keshavjee, K. (2016). Performance analysis of data mining classification techniques to predict diabetes. *Procedia Computer Science, 82*, 115–121.

Pinto, M. r. P. (2010). *Microsoft excel 2010. In (14.0.4760.1000 (34-bit), ed.)*. Microsoft Corporation.

Reif, M., Shafait, F., Goldstein, M., Breuel, T., & Dengel, A. (2014). Automatic classifier selection for non-experts. *Pattern Analysis and Applications, 17*, 83–96.

Romero, C., Olmo, J. L., & Ventura, S. (2013). A meta-learning approach for recommending a subset of white-box classification algorithms for Moodle datasets. In *EDM* (pp. 268–271).

Saaty, T. L. (1980). *The analytical hierarchy process: Planning, setting priorities, resource allocation*. New York: McGraw-Hill International Book Co.

Saaty, T. L. (1990). How to make a decision: The analytic hierarchy process. *European Journal of Operational Research, 48*, 9–26.

Saaty, T. L. (2003). Decision-making with the AHP: Why is the principal eigenvector necessary. *European Journal of Operational Research, 145*, 85–91.

Saaty, T. L. (2006). Rank from comparisons and from ratings in the analytic hierarchy/network processes. *European Journal of Operational Research, 168*, 557–570.

Singh, A., & Singh, M. L. (2016). Performance evaluation of various classifiers for color prediction of rice paddy plant leaf. *Journal of Electronic Imaging, 25*, 061403.

Singh, P., Verma, A., & Chaudhari, N. S. (2016). Performance evaluation of classifier combination techniques for the handwritten devanagari character recognition. In *Information systems design and intelligent applications* (pp. 651–662). India: Springer.

Smith, K. A., Woo, F., Ciesielski, V., & Ibrahim, R. (2002). Matching data mining algorithm suitability to data characteristics using a self-organizing map. In *Hybrid information systems* (pp. 169–179). Springer.

Soares, C., Brazdil, P., & Costa, J. (2000). Measures to evaluate rankings of classification algorithms. In *Data analysis, classification, and related methods* (pp. 119–124). Springer.

Soares, C., Costa, J., & Brazdil, P. (2000). A simple and intuitive measure for multicriteria evaluation of classification algorithms. In *ECML2000 workshop on meta-learning: Building automatic advice strategies for model selection and method combination* (pp. 87–96). Citeseer.

Song, Q., Wang, G., & Wang, C. (2012). Automatic recommendation of classification algorithms based on data set characteristics. *Pattern recognition, 45*, 2672–2689.

Sprinkhuizen-Kuyper, I. G., Smirnov, E. N., & Nalbantov, G. I. (2005). Reliability yields information gain. In *Proceedings of the 14th annual machine learning conference of Belgium and the Netherlands: BENELEARN: 3* (pp. 89–96). Citeseer.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 111–147.

Tsiporkova, E., Tourwé, T., & Boeva, V. (2010). A collaborative decision support platform for product release definition. In *2010 fifth international conference on internet and web applications and services* (pp. 351–356). IEEE.

Tzeng, G.-H., & Huang, J.-J. (2011, June 22). *Multiple attribute decision making: methods and applications*. CRC press.

Van Rijn, J. N., Bischl, B., Torgo, L., Gao, B., Umaashankar, V., Fischer, S., et al. (2013). OpenML: A collaborative science platform. In B. Hendrik (Ed.), *Machine learning and knowledge discovery in databases* (pp. 645–649). Springer.

Vanderlooy, S., & Hüllermeier, E. (2008). A critical analysis of variants of the AUC. *Machine Learning, 72*, 247–262.

Wang, G., Song, Q., Zhang, X., & Zhang, K. (2014). A generic multilabel learning-based classification algorithm recommendation method. *ACM Transactions on Knowledge Discovery from Data (TKDD), 9*, 7.

Witten, I. H., & Frank, E. (2005). *Data mining: practical machine learning tools and techniques*. Morgan Kaufmann.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on, 1*, 67–82.

Zar, J. H. (1972). Significance testing of the Spearman rank correlation coefficient. *Journal of the American Statistical Association, 67*, 578–580.

Zavadskas, E. K., Zakarevicius, A., & Antucheviciene, J. (2006). Evaluation of ranking accuracy in multi-criteria decisions. *Informatica, 17*, 601–618.