An up-to-date comparison of state-of-the-art classification algorithms

Expert Systems With Applications 82 (2017) 128–150

Contents lists available at ScienceDirect

Expert Systems With Applications

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

An up-to-date comparison of state-of-the-art classification algorithms

Chongsheng Zhang[a], Changchang Liu[a], Xiangliang Zhang[b], George Almpanidis[a,*]

[a] School of Computer and Information Engineering, Henan University, KaiFeng 475001, China
[b] King Abdullah University of Science & Technology, Thuwal 23955-6900, Saudi Arabia

ARTICLE INFO

Article history:
Received 10 December 2016
Revised 1 April 2017
Accepted 2 April 2017
Available online 5 April 2017

Keywords:
Classification benchmarking
Classifier comparison
Classifier evaluation

ABSTRACT

Current benchmark reports of classification algorithms generally concern common classifiers and their variants but do not include many algorithms that have been introduced in recent years. Moreover, important properties such as the dependency on number of classes and features and CPU running time are typically not examined. In this paper, we carry out a comparative empirical study on both established classifiers and more recently proposed ones on *71* data sets originating from different domains, publicly available at *UCI* and *KEEL* repositories. The list of 11 algorithms studied includes Extreme Learning Machine (ELM), Sparse Representation based Classification (SRC), and Deep Learning (DL), which have not been thoroughly investigated in existing comparative studies. It is found that Stochastic Gradient Boosting Trees (GBDT) matches or exceeds the prediction performance of Support Vector Machines (SVM) and Random Forests (RF), while being the fastest algorithm in terms of prediction efficiency. ELM also yields good accuracy results, ranking in the *top-5*, alongside GBDT, RF, SVM, and C4.5 but this performance varies widely across all data sets. Unsurprisingly, top accuracy performers have average or slow training time efficiency. DL is the worst performer in terms of accuracy but second fastest in prediction efficiency. SRC shows good accuracy performance but it is the slowest classifier in both training and testing.

1. Introduction

Classification refers to the problem of assigning an observation x_i to one hidden qualitative class $y_i \in \{1, 2, \ldots, L\}$, where it is generally assumed that $L < \infty$. Data classification has important applications that cover a wide spectrum including engineering, e-commerce, and medicine.

In the long history of machine learning, a great many classification algorithms have been proposed, some of which have been acknowledged as highly accurate, in particular Support Vector Machines (SVM) (Cortes & Vapnik, 1995) and Random Forests (RF) (Breiman, 2001).

Stochastic Gradient Boosting Decision Trees (GBDT) (Friedman, 2002) can also achieve outstanding prediction performance, but they are not as popular as SVM and RF. This algorithm has commonly been neglected in classification benchmarking in the literature, e.g. it was not included for comparison in Macia and Bernado-Mansilla (2014) and Fernández-Delgado, Cernadas, Barro, and Amorim (2014). Other algorithms, such as AdaBoost (AB) (Freund & Schapire, 1996), Naive Bayes classifier (NB) (Duda, Stork, & Hart, 2000), and C4.5 (Quinlan, 1993), have also been adopted in many classification tasks.

In 2008, Wu et al. (2008) reported the top-10 data mining algorithms. Among them, there are 6 classifiers, including SVM, C4.5, KNN, AB, NB, CART (Classification And Regression Tree). Due to the complex nature and increasing scale of many real-world problems in different domains, new classifiers are continuously being proposed. Among them, the most noteworthy are Extreme Learning Machine (ELM) (Huang, Zhou, Ding, & Zhang, 2012), Sparse Representation based Classification (SRC) (Wright, Yang, Ganesh, Sastry, & Ma, 2009), and Deep Learning (DL) (Bengio, 2009).

An up-to-date comparative study of existing classifiers is essential for two reasons: first, new classifiers need to be evaluated to compare their performance with the existing ones; second, both researchers and engineers need empirical reference/guidance in selecting the most appropriate classifiers for their specific problems.

In this paper, we aim at comparing state-of-the-art classification algorithms, including both the newly proposed classifiers and the established ones, on a large number of data sets that cover various application domains. This work contributes on answering the following questions: (a) are SVM and RF truly the best classifiers in terms of accuracy? If not, which classifiers are the top performers? (b) how do the newly proposed algorithms (i.e. ELM, SRC, and DL) perform in comparison with the established ones (i.e. SVM, RF, NB, C4.5, GBDT etc.)? (c) how do existing classification algorithms rank

* Corresponding author.
E-mail addresses: chongsheng.zhang@yahoo.com (C. Zhang), 25475510@qq.com (C. Liu), Xiangliang.Zhang@kaust.edu.sa (X. Zhang), almpanidis@gmail.com (G. Almpanidis).

http://dx.doi.org/10.1016/j.eswa.2017.04.003
0957-4174/

in terms of running time? We are motivated to solve these problems because of the insufficiency of current comparative study, as summarised in Table 1.[1]

Various researchers have addressed the problem of empirically comparing classifiers on multiple data sets, but most of the existing work in the literature focuses on well-established algorithms, without considering recent advances in classification in the Machine Learning field. New classifiers may have been compared to a reference set of classifiers by domain experts with fine-tuned settings, but the evaluation is limited to their specific domains, e.g. Ngai, Xiu, and Chau (2009) and Ballings, den Poel, Hespeels, and Gryp (2015), and a small number of available data sets.

Classification performance has been commonly evaluated on classification accuracy. However, the running time efficiency on classification model training, parameter-tuning, and testing is missing, which is a very important concern for most real-world applications.

The value of this work lies in the following points:

1. We compare newer classifiers, namely ELM, SRC and DL, with established classifiers; SVM, RF, AB, C4.5, NB, K Nearest Neighbours classifier (KNN) and Logistic Regression (LR).
2. Several well-performing algorithms are firstly included in comparison, which are GBDT, SRC, and DL.
3. We conduct large-scale experiments on 71 data sets to evaluate the performance of the classifiers. We use accuracy, ranking-based methods, and statistical classifier comparison methods to evaluate the performance.
4. We examine separately the training time, parameter-tuning time, and prediction time of both the newer and the established classification algorithms. As far as we know, this has not been studied in the literature.

The remainder of the paper is organised as follows. In Section 2, we present state-of-the-art on the problem of comparative study of classifiers. In Section 3, we briefly introduce the classifiers to be compared and discuss common approaches to improving classification performance. We then illustrate classifier comparison metrics in Section 4. Next, we give details in experiments setup in Section 5. We show and analyse the comparison results in Section 6. Finally, in Section 7, we conclude the work with directions for future research.

2. State-of-the-art in classifier comparison

Comparison of classifiers is important for both academic and industrial fields. A number of papers have conducted comparative study on data classification algorithms, as summarised in Table 1. We will discuss these papers in detail in this section.

Choosing the correct algorithm for a specific classification task based primarily on a-priori knowledge of the classifiers' behaviour across different domains is not safe, unless the evaluation is done in a systematic manner so that the results can be replicated and generalised. The recent growth of publicly available data sets and collaboration frameworks over the last years has enabled the Machine Learning community to easily share and validate experiments and results. Consequently, this has advanced much broader investigation and more thorough analysis in comparative studies of the performance of different classification algorithms based on a large set of real-world data sets. For example, the popular UCI Machine Learning Repository (Blake & Merz, 1998) contains about 240 data sets, two thirds of which are for classification problems, such as predicting multi-scale disease severity, recognising signs and handwritten characters, and identifying material types.

Zheng (1993) proposed the use of 16 dimensions (such as accuracy, the number of attributes/classes/instances/density of the data sets, etc.) to evaluate the accuracy of 3 classification algorithms developed up to the year of 1993, including C4.5 and ID3.

In StatLog project, King, Feng, and Sutherland (1995) compared multiple classification algorithms from symbolic learning (including C4.5), statistics (including NB, KNN, LR), and neural networks (NN), on large real-world problems. They found that the performance depends critically on the data set investigated and there is no single best algorithm. This argument is in accordance with the No-Free-Lunch theorem (Wolpert, 1996), which states that the best classifier will not be the same for all the data sets.

Lessmann, Baesens, Seow, and Thomas (2015) compared different classifiers on 8 real-world credit scoring data sets. They found that individual classifiers (such as C4.5, ELM, LR, NN) predict, with few exceptions, significantly less accurate than RF.

Lorena et al. (2011) presented an experimental study comparing the use of 9 classifiers (Repeated Incremental Decision Trees, RF, KNN, NB, LR, SVM, and Artificial Neural Networks) to model the potential distribution of 35 Latin American plant species. The results show that RF has remarkable performance in all data sets. While the performance of SVM is statistically similar with RF, all the other classifiers were outperformed by them. The authors also found that, LR results were not stable along different data sets.

Brown and Mues (2012) studied the performance of 8 classifiers, including SVM, RF, GBDT, NN, C4.5, KNN, LR, and LDA, for credit scoring, which is a binary (2-class) imbalanced classification problem. The results of their experiments show that GBDT and RF performed well in dealing with samples where a large class imbalance was present.

Macia and Bernado-Mansilla (2014) analysed the type, complexity, and use of UCI data sets and contrasted the accuracy rate obtained by 8 classifiers from different learning paradigms that represent some of the fundamental algorithms of Machine Learning, including C4.5, RF, Multilayer Perceptron (MLP), SVM, LR, and NB. Using the implementations provided by Weka, they found that the accuracy of C4.5, LR, NB, and RF over the UCI repository is very similar for most of the data sets and the deviation of accuracy rates is small.

Recently, Fernández-Delgado et al. (2014) evaluated 179 classifier settings arising from 17 families, implemented in Weka, R, C, and Matlab and found out that parallel RF implemented in R performed best, followed by SVM (LibSVM with Gaussian kernel). Among the top-10 performers there are also two NN settings. The authors show that, all the classifiers of RF and SVM families are included among the 25 best classifiers, with accuracies above 79% (while the best is 82.3%), which identify both families as the best ones. In general, these two families perform better than C4.5, AB, LR, and NB settings and ensembles. However, the tested algorithms did not include GBDT, ELM, SRC, and DL.

Lim, Loh, and Shih (2000) conducted experiments on 32 data sets to compare 22 decision trees, 9 statistical, and 2 neural network algorithms in terms of classification accuracy and training time. The authors found that (Multinomial) LR algorithms were the most accurate classifiers. Yet, the tested algorithms did not include GBDT, RF, ELM, SVM, SRC, and DL.

Jones, Johnstone, and Wilson (2015) examined the predictive performance of a wide class of binary classifiers using a large sample of international credit rating changes data sets from the period 1983-2013. Their results indicate that generalised boosting, AB, and RF strongly outperformed SVM, LDA, and LR. However, this study is limited to binary classifiers.

Nanni, Brahnam, Ghidoni, and Lumini (2015a) investigated the performance of general-purpose heterogeneous ensembles for

[1] More details of related work about comparative study of classifiers will be discussed in Section 2.

Table 1
Summary of related work on classifier comparison in chronological order.

Related work	Classifiers compared	Missing classifiers	Data sets tested	Evaluation criteria	Conclusions reached
Zheng (1993)	C4.5, IB1, ID3	AB, DL, ELM, GBDT, LR, RF, SRC, SVM	13 UCI data sets (Diabetes, Hepatitis, Thyroid, Mushroom, etc.).	ACC, Average Information Score, Relative Information Score, Entropy	C4.5 is much better than the ID3 and IB1.
King et al. (1995)	C4.5, CART, KNN, LR, NB, NN	AB, DL, ELM, GBDT, RF, SRC, SVM	11 UCI data sets (Satellite, Vehicle, Shuttle, German, etc.).	ACC, Misclassification cost, Confusion matrices, Training/testing time	There is no single best algorithm. C4.5 has the best accuracy when the data is skewed; NB should not be considered unless the data set has very low correlations.
Lim et al. (2000)	C4.5, CART, KNN, LDA, LR, NN, etc.	DL, ELM, GBDT, RF, SRC, SVM	15 UCI datasets (Contraceptive, Breast cancer, Diabetes, Segmentation, Thyroid, Waveform, etc.).	Minimum/maximum/naive plurality rule error rate, Ordering by mean error rate and mean rank of error rate, Statistical significance of error rates (ANOVA, rank analysis), Training time, Scalability tests	(Multinomial) LR is the most accurate classifier.
Caruana and Niculescu-Mizil (2006)	LR, NB, NN, RF, SVM, decision/boosted/bagged trees, boosted stumps	AB, C4.5, DL, ELM, SRC	8 UCI data sets for binary classification (Adult, Covertype, SLAC, etc.).	ACC, F-measure, Lift, AUC, Average Precision, Precision/Recall break even point, Squared Error, Cross-Entropy	NN give the best average performance in all performance metrics, followed by bagged trees and RF. Calibration dramatically improves RF and boosted methods, SVM, NB, and RF, but not NN, LR, and bagged trees.
Lorena et al. (2011)	KNN, LR, NB, NN, RF, SVM	AB, DL, ELM, GBDT, SRC	35 plant species data sets (with 60–193 instances).	Mean and standard deviation AUC values, Statistical significance comparison	The overall performance of SVM and RF are statistically similar, while RF outperforms the rest classifiers.
Brown and Mues (2012)	C4.5, GBDT, KNN, LDA, LR, NN, RF, SVM	AB, DL, ELM, SRC	5 credit data sets (binary, imbalanced).	AUC, Mean ranks (Friedman test with Nemeyi's critical difference tail)	GBDT and RF performed well on binary imbalanced data sets.
Macia and Bernado-Mansilla (2014)	8 classifiers including C4.5, LR, NB, NB, RF, SVM	AB, DL, ELM, GBDT, SRC	166 data sets (Abalone, Ecoli, Isolet, Diabetes, Page Blocks, Shuttle, Spambase, Yeast, Wine, Thyroids, Mush, etc.).	ACC	The accuracy of C4.5, MLP, SVM, RF are very similar for most of the data sets.
Fernández-Delgado et al. (2014)	AB, C5.0, ELM, KNN, MLP, NB, RF, SVM, etc.	DL, GBDT, SRC	121 data sets, 108 of which are UCI data sets.	Average ACC, Cohen κ, AUC, Friedman ranking, paired T-tests	The 5 best classifiers are RF, SVM, ELM, C5.0, and MLP, respectively; ELM ranks first in the number of data sets for which a classifier achieves the highest accuracy. RF and SVM rank best in the number of data sets in which a classifier achieves 95% or more of the maximum accuracy.
Lessmann et al. (2015)	C4.5, ELM, LR, NN, RF SVM	AB, DL, GBDT, SRC	7 Credit Scoring data sets.	Percentage correctly classified, AUC, Partial Gini index, H-measure, Brier Score, Kolmogorov-Smirnov statistic	RF performs significantly more accurate than the rest.
Jones et al. (2015)	Generalised boosting, AB, RF, LDA, LR, NN, SVM, etc.	C4.5, DL, ELM, GBDT, SRC, C4.5	Credit rating datasets (binary classification).	Average overall AUC, H-measure	Generalised boosting, AB and RF strongly outperform the other classifiers.
Nanni et al. (2015a)	Variants of AB, DL, GPC, SVM, and their heterogeneous ensembles.	C4.5, ELM, GBDT, LR, RF, SRC	14 image classification data sets, 11 data mining data sets from UCI.	AUC	General-purpose ensembles outperform SVM.
Nanni et al. (2015b)	SVM, ensembles based on AB, DT, or SVM.	C4.5, ELM, GBDT, LR, RF, SRC	44 binary (including folds), 6 multi-class, and 12 strongly imbalanced data sets from KEEL and UCI repositories.	AUC, F-measure, G-mean, Q-statistic	The proposed HardEnsemble approach does not need parameter tuning and outperforms existing ensembles.

pattern classification using different feature perturbation techniques such as random subspace, over single classifiers, such as SVM, Gaussian Process Classifier (GPC), and Deep Learning methods. The authors assessed the classifiers' generalisability using 14 image classification data sets and 11 UCI data mining data sets. The authors reported that, while there is no single approach that can outperform all the other classifiers, general-purpose systems are capable of handling a broader range of classification problems than state-of-the-art classifiers, in specific SVM.

In comparison to our work, the authors in Nanni et al. (2015a) focus on heterogeneous ensemble approaches rather than single classifiers. Moreover, they aim to study the performance of these methods on image classification and data mining domains, while, in our work, we use a collection of data sets from various domains. Some of the classifiers that we have investigated are not included in Nanni et al. (2015a), such as ELM, GBDT, and SRC. Furthermore, training and testing time efficiency is not considered in their work. Both Nanni et al. (2015a) and our work make classifier performance comparisons using non-parametric statistical tests. In our work we use Dunn's test instead of Wilcoxon signed ranked test; the reasons for our choice are stated in Section 4.

Nanni, Fantozzi, and Lazzarini (2015b) compare different ensemble methods that use either AB, KNN, or SVM as the base classifiers. They report area under the receiver operating characteristic curve (AUC), F1-measure, and geometric mean values for both binary and multiclass imbalanced data sets from KEEL repository (Alcalá-Fdez et al., 2010; 2009). They propose a new ensemble construction method, namely HardEnsemble, which does not require parameter tuning but still outperforms all the other tested approaches. The aim of their work is to evaluate the performance of state-of-the-art classification methods for handling the imbalance problem, in particular the "ensemble of ensembles" strategy. In comparison, the goal of our work is to provide an up-to-date comparative study of classification algorithms that are known to be good performers with algorithms that are relatively new, as well as several other established classifiers. In our work, we have not considered the class imbalance issue.

In summary, as can be seen in Table 1, most of existing works for classifier comparison have not considered several important classifiers, i.e. GBDT, ELM, SRC, and DL. Moreover, the findings in these works are not always consistent, e.g. the accuracy of RF and SVM were reported to be very similar in Macia and Bernado-Mansilla (2014), while in Brown and Mues (2012), Fernández-Delgado et al. (2014) and Lessmann et al. (2015), RF was reported to outperform SVM. Therefore, it is essential to conduct an up-to-date comparative study on the current state-of-the-art classification algorithms, taking into consideration GBDT and newer classifiers, i.e. ELM, SRC, and DL.

3. Classification algorithms to be compared

For the sake of convenience and clarity, we group existing classification algorithms investigated in this work into three groups. The first category mainly includes Support Vector Machines (SVM) and Random Forests (RF), which are known to be among the best performers (thus usually used as the default classifiers). The second category consists of relatively new classifiers, i.e. ELM, SRC, and DL. These algorithms were proposed in recent years, so they have not been included in most comparative studies. The last category contains other established classifiers, including C4.5, AdaBoost (AB), K Nearest Neighbours classifier (KNN), Logistic Regression (LR), and Stochastic Gradient Boosting Trees (GBDT). Such algorithms are not as popular as SVM and RF, but they are also important and have found applications in many domains. In particular, GBDT is commonly underutilised by many researchers and practitioners, given the fact that it has been reported in the literature to achieve high classification accuracy, e.g. Caruana and Niculescu-Mizil (2006), Brown and Mues (2012), and Chapelle and Chang (2011).

3.1. SVM and RF

Both SVM and RF are popular and accurate algorithms, thus commonly used as the default classifiers in a vast amount of applications. SVM (Cortes & Vapnik, 1995) blends linear modelling with instance-based learning, it selects a small number of critical boundary samples from each category and builds a linear discriminate function that separates them as widely as possible. In the case that no linear separation is possible, the technique of kernel will be used to automatically inject the training samples into a higher-dimensional space and to learn a separator in that space. SVM is acknowledged to be among the most reliable and accurate algorithms in most Machine Learning applications.

Random Forests (RF) (Breiman, 2001) is an ensemble learning method for classification which operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. RF corrects for decision trees' habit of overfitting to their training set. Like SVM, RF has also been recognised to be among the most accurate classifiers.

Khoshgoftaar, Golawala, and Hulse (2007) conducted experiments to analyse the performance of the RF learner implemented in Weka. The author suggested a default value of 100 for *numTrees* and $\lfloor \log_2 M + 1 \rfloor$ for *numFeatures* for binary classification in general.

Pal (2005) compared the performance of RF with SVM in terms of classification accuracy, training time and user-defined parameters. The author found that, RF can achieve a classification accuracy which is comparable to that achieved by SVM. Moreover, RF can handle categorical data, unbalanced data, as well as data with missing values, which is not possible with SVM.

Vanschoren, Blockeel, Pfahringer, and Holmes (2012) also performed experiments with UCI data sets and they attested that RBF-based SVM and RF yield good results in predictive accuracy but the variation in the performance is large due to parameters of the data that heavily affect their performance. They also observed that it is more rewarding to fine-tune RF and SVM than to bag or boost their default setting, whereas for C4.5, bagging and boosting outperform fine-tuning.

3.2. Newer classifiers

Most existing works in classifier comparison have yet to fully research newer classifiers, in particular, ELM, SRC, and DL (for classifying numeric data, i.e. non-image data).

Extreme Learning Machine (ELM) (Huang et al., 2012) is a recently available learning algorithm for single layer feedforward neural network. Compared with classical learning algorithms in neural networks, e.g. Back Propagation, ELM can achieve better performance with much shorter learning time. In some of the existing work, it is claimed to yield better performance in comparison with SVM.

However, Liu, Loh, and Tor (2005) conducted comparisons between ELM and SVM for text classification. They show that, while ELM is easy to tune with a single parameter and is robust to the parameter settings, SVM still outperforms ELM for the majority of categories.

Nevertheless, Huang et al. (2012) compared ELM with SVM in regression and multiclass classification and found that ELM tends to have better scalability and achieve similar or much better (for multiclass cases) generalisation performance at much faster learn-

ing speed (up to thousands times) than SVM. Moreover, ELM requires less human intervention than SVM.

Sparse Representation based Classification (SRC) (Wright et al., 2009) is a powerful tool for face recognition and classification (Wei et al., 2013). In SRC, each test sample is represented, without computing features, as a sparse combination of training samples, then assigned to the class that minimises the residual between itself and the reconstruction by training samples of this class. SRC shows its effectiveness in face recognition experiments. However, except image/audio data, few experiments have been carried out to comprehensively investigate the accuracy of SRC on numeric data.

Deep Learning (DL) (Bengio, 2009) is a new and very hot area in Machine Learning research. It has demonstrated outstanding performance in image/speech recognition and related applications. Deep Neural Networks (DNN) are large multilayer neural networks (MLP). They are typically designed as feedforward networks trained with the standard backpropagation algorithm. Usually, a Convolutional Neural Network (CNN) is a DNN consisting of one or more pairs of convolutional layer(s) and max-pooling layer(s), followed by one or more fully-connected hidden layers. The goal of CNN is to find a set of locally connected neurons. It continuously extracts many low-level features (e.g. image pixels) into the compressed high-level representations and abstractions (e.g. edges, eyes, ears). CNNs are easier to train and have fewer parameters than DNNs. CNNs have been explored for speech recognition and image classification, showing improvements over DNNs (Krizhevsky, Sutskever, & Hinton, 2012). A Deep Belief Network (DBN) (Hinton, Osindero, & Teh, 2006) is a type of DNN that uses restricted Boltzmann machines to derive good initial parameters. In recent years, DL has drawn a great many research efforts in computer vision (Deng & Yu, 2014). Yet, few experiments have been carried out to comprehensively investigate the performance of DL on numeric data (non-image data).

3.3. Other established classifiers

The gradient boosting method represents an ensemble of single regression trees built in a greedy fashion. It produces a prediction model in the form of an ensemble of weak prediction models, such as decision trees. Stochastic Gradient Boosting Trees (GBDT) (Friedman, 2002; Ye, Chow, Chen, & Zheng, 2009) combines gradient boosting with bootstrap bagging. At each iteration of the algorithm, a new decision tree model is built based on the residuals of the previous decision trees. GBDT is a simple yet very effective method for learning non-linear functions (Chapelle & Chang, 2011).

AdaBoost (AB) (Freund & Schapire, 1996) is a technique that builds an ensemble of classifiers sequentially, one classifier at a time, until the predefined number of classifiers is reached. Each subsequent classifier is trained on a set of samples with weights to emphasise the instances misclassified by the previous classifiers. The ensemble decision is made by weighted voting. The weights are determined by the individual accuracies. AB has been found to be very useful (e.g. in face detection) but too sensitive to noise in the data.

Naive Bayes classifier (NB) (Duda et al., 2000) is a probabilistic learner based on the Bayesian rule. It is based on the assumption that the features are conditionally independent given the class label, which may not be the case for data sets such as gene expression data because of co-regulation. Nevertheless, it is among the most practical approaches to certain types of learning problems (e.g. spam filtering).

The K Nearest Neighbours (KNN) classifier (Altman, 1992) is a type of non-parametric, instance-based learning algorithm, where objects are classified by a majority vote of their K nearest neighbours having known class labels, with K being a positive, typically small, integer. It is considered to be among the simplest and most commonly-used classification algorithms but has the disadvantage of high degree of local sensitivity (when the parameter K is small), making it highly susceptible to noise in the training data, and reduced predictive power (when the value of K is high).

Logistic Regression (LR) (Cox, 1958) is a regression analysis method when the dependent variable is categorical. Extending linear regression, LR measures the relationship between the dependent variable and one or more independent variables by estimating probabilities using a logistic function. This can also be generalised to multiclass problems (i.e. multinomial logistic regression). Thus, it is possible to do multiclass classification-by-regression using a multinomial logistic regression model with a ridge estimator (Le Cessie & Van Houwelingen, 1992).

3.4. Techniques for improving classification performance

Classifier ensemble and feature selection are two common practices for improving classification performance.

3.4.1. Classifier ensemble

Ensemble classifiers are expected to perform better than single classifiers.

Hu, Li, Plank, Wang, and Daggard (2006) conducted a comparative study of classification methods, namely LibSVM, C4.5, BaggingC4.5, AB, and RF for microarray data analysis. The experimental results show that ensemble decision tree methods can improve the accuracy over single decision tree methods.

Kuncheva and Rodríguez (2010) experimented on classifier ensembles for fMRI data analysis. They found that the best ensemble group is the one where SVM is used as the base classifier in the ensemble.

Kuncheva (2002) also explored the switching between selection and fusion in combining classifiers. The author found that for different classifier models in the team, it is difficult to predict whether a combination can achieve a better accuracy than the best individual classifier. The result depended on the data set, the only consistent pattern being that the improvement from classifier ensembles over the best individual classifier was negligible.

In the "Yahoo! Learning to Rank Challenge", Chapelle and Chang (2011) pointed out that decision trees (especially GBDT) were the most popular class of functions among the top competitors. Moreover, comparing the best solution of ensemble learning with the baseline of regression model (GBDT), the relevance gap is rather small.

3.4.2. Feature selection and extraction

Feature selection methods have been well studied in data classification (Dash & Liu, 1997; Guyon & Elisseeff, 2003; Liu & Yu, 2005; Saeys, Inza, & Larrañaga, 2007).

Li, Zhang, and Ogihara (2004) provide a comparative study on feature selection and multiclass classification for gene expression data. The study suggests that multiclass classification problems are more difficult than binary ones in general. SVM is shown to be the best classifier, followed by C4.5, NB, and KNN, for tissue classification based on gene expression. It achieves better performance than any other classifiers on almost all the data sets.

Liu (2004) evaluates feature selection methods in dimensionality reduction for drug discovery. The author's experiments show that NB benefits significantly from feature selection, while SVM performs better when all features are used. When information gain was used to select the features, SVM was much less sensitive to the reduction of feature space.

Arauzo-Azofra, Aznarte, and Benitez (2011) conducted an empirical study on feature selection methods based on individual fea-

ture evaluation. Their experiments show that, while reducing the number of features used, methods such as Information Gain ratio, Relief and Gini ratio, improved classification results in most of the problems considered.

Saeys, Abeel, and de Peer (2008) investigated on high-dimensional data, the use of ensemble of multiple feature selection methods for improving classification stability and accuracy. The authors show that ensemble of RFs clearly outperforms other feature selection methods in terms of stability, yet with decreased classification accuracy in comparison to a single RF classifier. However, for the ensemble of other feature selection methods, the classification accuracy remains the same (or slightly improved), but the corresponding classification stability can be improved over single feature selection method.

Another approach for improving classification performance is representing feature vectors as matrices. Several studies have explored different feature reshaping techniques using matrix representations and the resulting methods have shown to significantly outperform existing classifiers in both accuracy and computation time (Nanni, Brahnam, & Lumini, 2010). Nanni, Brahnam, and Lumini (2012) proposes a method that combines the Meyer continuous wavelet approach for transforming a vector into a matrix with the a variant of the local phase quantisation for feature extraction. Using generic pattern recognition data sets from the UCI repository, as well as real-world protein and peptide data sets, the authors show that their proposed method yields similar or even better performance than that obtained by the state-of-the-art.

4. Methodology of classifier comparison

4.1. Statistical classifier comparison methods

Different statistical methods have been proposed for multiple classifier comparison. Demšar (2006) and Garcia and Herrera (2009) reviewed common practices in the literature for the comparison between different classifiers over multiple data sets and examined, both theoretically and empirically, the suitability of these methods. The authors point out that, non-parametric tests assume some, but limited, commensurability, while at the same time they do not assume normal distributions or homogeneity of variance, thus they are statistically safer and more robust than parametric tests. Demsar recommends that the Friedman test (Friedman, 1937), with the corresponding post-hoc tests, is appropriate for comparison of more than two classifiers over multiple data sets and it can be applied to classification accuracies, error ratios or any other measure for evaluation of classifiers, such as computation time.

Friedman test (Friedman, 1937) is a non-parametric randomised block analysis of variance, which is equivalent to parametric ANOVA with repeated measures. Thus, assumptions of a normal distribution and equal variances (of the residuals) are not required. Friedman test is used to detect differences in groups of results.

Given k classifiers and n data sets, a classifier $j = 1, \ldots, k$ is assigned a s_{ij} performance metric value (e.g. classification accuracy) in data set $i = 1, \ldots, n$, which can be modelled as a $\{s_{ij}\}_{n \times k}$ matrix, where each column represents a specific classifier's accuracy values across all data sets, while each row represents all classifiers' accuracies in a specific data set. The procedure involves ranking each row (or block) together, then considers the values of ranks by columns, which gives as an output a new matrix $\{r_{ij}\}_{n \times k}$, with each entry r_{ij} corresponding to the rank of s_{ij} within row i. This ranks the algorithms for each data set separately, the best performing algorithm getting the rank of 1, the second best ranks 2nd, etc. In case of ties, average ranks are assigned. The resulting test statistic Q is computed according to Eq. (1)

$$Q = n^2(k-1)\frac{\sum_{j=1}^{k}\left(\bar{r}_{\cdot j} - \bar{r}\right)^2}{\sum_{i=1}^{n}\sum_{j=1}^{k}\left(\bar{r}_{ij} - \bar{r}\right)^2} \tag{1}$$

where

$$\bar{r}_{\cdot j} = \frac{1}{n}\sum_{i=1}^{n} r_{ij} \qquad \bar{r} = \frac{1}{nk}\sum_{i=1}^{n}\sum_{j=1}^{k} r_{ij} \tag{2}$$

and it follows approximately (when n or k is large, typically $n > 15$ or $k > 4$) a chi-squared distribution. The p-value for the corresponding hypothesis testing is given by $P(\chi^2_{k-1} \geq Q)$.

Another non-parametric test is the *sign test*, which only makes use of information of whether a value is greater, less than or equal to the other in a pair. Wilcoxon signed rank test (Wilcoxon, 1945) calculates differences of pairs. The absolute differences are ranked after discarding pairs with the difference of zero. The ranks are sorted in ascending order. When several pairs have absolute differences that are equal to each other, each of these several pairs is assigned as the average of ranks that would have otherwise been assigned. The hypothesis is that the differences have zero mean.

While Friedman test (followed by post-hoc analysis) can be applied to multiple matched or paired groups, Wilcoxon test can only be used in pairwise comparisons. Therefore, in the experimental analysis of our work, we use Friedman test to compare the performance of different classification algorithms.

Dunn's test is a multiple comparison test that controls the family-wise error rate (Dunn, 1964). Dunn's test can be used as a post-hoc analysis to Friedman test, by comparing pair-wise the difference in the sum of ranks between two classifiers with the expected average difference (based on the number of groups and their size) (Daniel, 2000). For a family of classifiers' comparisons, when comparing each classifier with every other classifier, the number of possible comparisons is $c = k(k-1)/2$. For a family significance threshold α, using a reverse lookup table or calculator, we calculate the critical value of z from the normal distribution that corresponds to the two-tailed probability α/c. When comparing classifier j with j', we check whether the ratio of the absolute difference value between their mean ranks and $\sqrt{\frac{k(k+1)}{6n}}$ is larger than z, i.e.

$$\frac{\bar{r}_{\cdot j} - \bar{r}_{\cdot j'}}{\sqrt{\frac{k(k+1)}{6n}}} > z \tag{3}$$

or, equivalently, if their absolute difference value between their sum ranks is

$$\sum_{i=1}^{n} r_{ij} - \sum_{i=1}^{n} r_{ij'} > \frac{n}{\sqrt{\frac{k(k+1)}{6n}}} \cdot z \tag{4}$$

If the inequality Eq. (4) holds, then the difference is statistically significant, otherwise it is not.

4.2. Measuring a Classifier's prediction performance

Classification performance metrics can be divided into different groups/types; threshold metrics, which assess the discriminatory ability of the classifier, probability metrics that assess the accuracy of the classifier probability predictions, ordering/rank metrics, and those that assess the correctness of the classification algorithm's categorical predictions (Caruana & Niculescu-Mizil, 2006; Lessmann et al., 2015).

The predictive accuracy (ACC) is a singular assessment, threshold metric that has been the most commonly used evaluation criterion in classification. It measures the probability that a classifier

makes the correct prediction. It is the ratio between the number of correct predictions and the total number of predictions. Accuracy provides a simple way of describing a classifier's performance on a given data set. However, since it does not take into account the class prior probabilities and class distribution of a dataset it can not provide adequate information on a classifier's behaviour in the case where data is imbalanced (He & Garcia, 2009; Zheng, 1993).

A receiver operating characteristic (ROC) curve shows the balance between a classifier's true positive rate and false positive rate at various threshold settings. ROC curves are usually used for binary (two-class) classification problems, although Landgrebe and Duin (2007) proposed to approximate the multiclass ROC by pairwise analysis.

AUC, namely Area Under Receiver Operating Characteristic (ROC) Curve, is another metric that has been widely used for assessing classification performance. AUC is equal to the probability that a classifier will rank a uniformly drawn, randomly chosen positive instance higher than a randomly chosen negative example. It measures the classifier's skill in ranking a set of patterns according to the degree to which they belong to the positive class. The AUC measure is very similar to Wilcoxon Rank Sum Test and Mann-Whitney U Test (Bradley, 1997). For multiclass classification problems, AUC can be extended to multiclass settings through pairwise coupling between the classes (Hand & Till, 2001); there will be $n(n-1)/2$ combinations, where n is the number of classes. Having all the combinations of classes, we then average the AUC values from each combination, which will be considered as the multiclass AUC result.

A great many different classification performance measures have been used in the literature; Recall and Precision, Recall/Precision break-even point, F-measure (Rijsbergen, 1979), Squared Error, Cross-Entropy, Kolmogorov-Smirnov statistic, Cohen's κ (Carletta, 1996), Lift (Giudici, 2005), Q-statistic (Yule, 1900), Brier score (Hernández-Orallo, Flach, & Ferri, 2012), H-measure (Hand, 2009), Average/Relative information score (Kononenko & Bratko, 1991), Partial Gini index (Baesens et al., 2003), etc. In Table 1 we give the metrics that are used in related studies about multiple classifier comparison.

Since different metrics capture different characteristics of a classifier, depending on the properties of the data, the choice of metrics influences how the performance of classification algorithms is measured and compared. For example, some measures are inappropriate for imbalanced data or multiclass learning, while others can reveal more information in such cases (Caruana & Niculescu-Mizil, 2006; He & Garcia, 2009).

In this work, we use both ACC and AUC metrics for evaluating classification algorithms. AUC values were computed using the implementations in R-package *caTools* (Tuszynski, 2008). To compare the efficiency of different classifiers, we also examine the separate training, parameter-tuning, and prediction time of each individual classifier.

4.3. Ranking-based classifier comparison

We also compare the classifiers based upon their ACC and AUC rankings.

For each classifier, we calculate the total number of data sets where this classifier is ranked as *top-x* ($x = 1..5$). Then, we obtain the overall probability (the percentage of data sets) for each classifier to be ranked as *top-x*, when classifying a data set.

In this paper, *top-1* denotes the number of data sets that a classifier achieves the best accuracy or AUC in comparison to the rest of the classifiers, divided by the total number of data sets. In accordance, *top-x* ($x = 1..5$) means the number of data sets where a classifier ranks within the *top-x* classifiers in terms of ACC or AUC, divided by the total number of data sets.

For a group of classifiers and a given data set, if there exists a classifier that ranks best in the group as *top-x* on the data set, then the group is considered as *top-x* on the data set.

To investigate the best possible performance for an assemble of heterogeneous classifiers, we calculate, for every specified group of classifiers, the percentage of data sets where the group is ranked as *top-x*.

5. Experimental setup

We perform all the experiments on a system with dual Intel Xeon E5-2620 v2 CPUs at 2.10GHz and 128GB RAM, running Windows Server 2008 R2 Enterprise.

In the experiments, we use 10-fold cross-validation to assess the performance of the classifiers on each data set. Each data set is split into 3 parts; 80% of the total instances in each data set are used for training a classification model (denoted as *training_data*), another 10% of the instances of the data set are used for tuning the parameters of the classification model (denoted as *validation_data*), and the last 10% of the instances are used for testing only (denoted as *test_data*). If a classifier does not need to tune its parameters, *validation_data* will be merged into *training_data*. All the classifiers use the same training and test data.

5.1. Data sets utilised in comparison

In total, 71 data sets are used for experiments. On these data sets, we give accuracy results of the 11 classifiers. For the efficiency evaluation of the algorithms we use a subset that includes 36 out of the 71 data sets, to demonstrate the running time efficiencies of different classification algorithms.

All the data sets are publicly-available classification data sets from the UCI Machine Learning Repository[2] (Blake & Merz, 1998), KEEL[3] (Alcalá-Fdez et al., 2010; 2009), and LibSVM[4] (Chang & Lin, 2011).

Since there are 71 data sets, it's not feasible to describe all of them. For data sets from UCI, detailed descriptions are available on the UCI website. For example, the *Pima* (Indian Diabetes) data set is used for predicting diabetes. There are 8 features, including *number of times pregnant, diastolic blood pressure, age*, etc.

Data sets from the KEEL repository are standard data sets but added with aleatory attribute noise. In order to corrupt each attribute A with a noise level of m%, the m% of the examples in the data set are chosen approximately and the value of A of each of these examples is assigned a random value between the minimum and maximum of the domain of that attribute, following a uniform distribution (Alcalá-Fdez et al., 2010; 2009; Zhu, Wu, & Yang, 2004). For instance, data set *spambase-10an-nn* derives from the standard *spambase* data set but with 10% of the original instances of the training set replaced with noise for each attribute.

Within these data sets, the number of classes is between 2 and 26, the number of features ranges from 2 to 256, and the number of instances ranges from 132 to 20,000. The distributions of the number of classes and the number of features, as well as the number of instances in different data sets are given in Fig. 1, Fig. 2, and Fig. 3, respectively.

5.2. Implementation selection for the classification algorithms

The selected classification algorithms in this work are the most popular ones, including almost all the classifiers from the top-10 data mining algorithms, reported in Wu et al. (2008). They are

[2] http://archive.ics.uci.edu/ml
[3] http://sci2s.ugr.es/keel/datasets.php
[4] https://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html

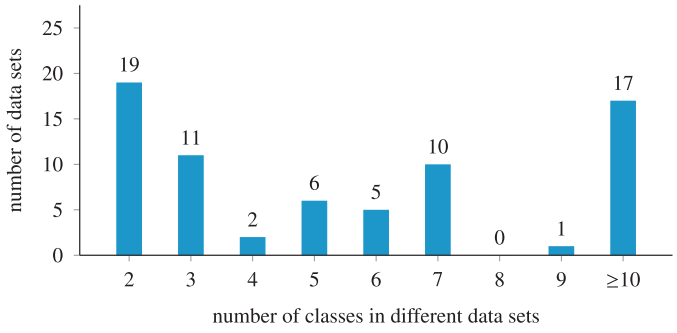

Fig. 1. Distribution of the number of classes in different data sets.

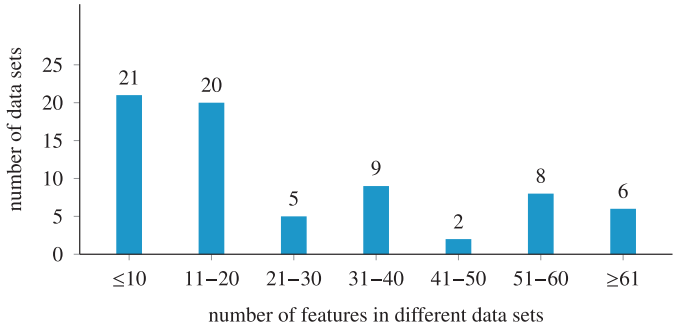

Fig. 2. Distribution of the number of features in different data sets.

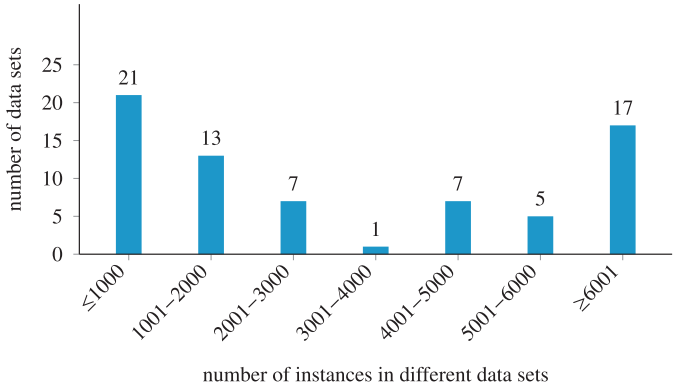

Fig. 3. Distribution of the number of instances in different data sets.

either publicly-available with source codes in Matlab or can be called from the Weka software. The only two exceptions are GBDT and DL.

For GBDT, for there is no publicly available implementation in Matlab or Weka, we adopt the GBDT implementation from the *XGBoost*[5] library (Chen & Guestrin, 2016).

For DL, we test three different methods, which are Neural Networks, DBN, implemented in Matlab using the Deep Learning Toolbox[6] (Palm, 2012), and CNN, implemented in Python using the Pylearn2[7] library (Goodfellow et al., 2013).

5.2.1. Parameter tuning

GBDT, RF, ELM, and SVM need parameter tuning to achieve the best performance.

It should be mentioned that, because it is not feasible to exhaustively try out all the possible values for each (numeric) parameter of GBDT (as well as other classifiers that need fine-tuning), moreover, the focus of this work is to compare the performance of the 11 classifiers, but not on the fine-tuning techniques for these classifiers, thus, we only try limited number of values for each parameter.

There are two crucial parameters in GBDT that should be tuned with *validation_data*: the first parameter is the learning rate that shrinks the contribution of each tree and its range is (0,1); the other parameter is the maximum depth of the individual regression estimators that limits the number of nodes in the tree, its range is {1,2,3,4}. In our experiments, with learning rate starting from 0.1 and increasing by 0.1, we test all the possible combinations between the two parameters and find the pair of parameters that can achieve the best accuracy on the *validation_data*. Thus, there are in total 40 such parameter-tuning tests on *validation_data*. The best pair of parameters on the *validation_data* will finally be used to make predictions on the *test_data*.

The two key parameters for RF are *NTrees* which is the number of individual classification trees to be trained on the *training_data* using different subsets of features to build an ensemble of classification trees, and *NVarToSample*, which denotes the number of variables to select at random for each decision split. In our experiments, we fix *NTrees* to be 100 (Khoshgoftaar et al., 2007) and vary *NVarToSample* from 1 to the total number of features, to find out the value for *NVarToSample* that can achieve the best classification accuracy on *validation_data*. Thus, the total number of tests on each *validation_data* for parameter tuning in RF is the same as the number of features in the data set. The ensemble classification model having the best value for *NVarToSample* will then be used for predicting the labels of the instances in the *test_data*.

For ELM, there are also two important parameters that should be tuned, which are the regularization coefficient and the kernel parameter. In our tuning, we first use the default value of the kernel parameter to find the best regularization coefficient value that can achieve the best classification accuracy, by varying its value from 1 until 100. We next fix the value of the regularization coefficient to its best value and find the best value for the kernel parameter with the range [1,150], using the same tuning procedure as regularization coefficient. Therefore, the total number of parameter-tuning tests in ELM is 250. We use the best values of the two parameters derived above to predict the labels of the instances in the *test_data*.

When testing SVM, we use the default type of SVM (*C-SVM*), for which we adopt the implementations from LibSVM (Chang & Lin, 2011). We have tried two kernels, RBF (Radial Basis Function) kernel and pre-computed kernel. For RBF kernel, we tune two parameters, the penalty parameter (C) and the parameter gamma (γ) in the kernel function, using the grid-search method (SVM with stochastic gradient, *SVMcg*) (Hsu, Chang, Lin et al., 2003). It has been proved that the linear kernel is essentially a special case of the RBF kernel (Keerthi & Lin, 2003).

We have not tested the polynomial and sigmoid kernels. The study in Lin and Lin (2003) proves that the sigmoid kernel is not better than the RBF kernel in general; moreover, the quality of the local minimum solution in parameters of the sigmoid kernel may not be guaranteed, it is therefore hard to select suitable parameters for the sigmoid kernel.

Regarding SVM, we test three different parameter settings, as can be seen in Table 2. Overall, the pre-computed kernel achieves the best accuracy in 40 data sets, while *SVMcg* is best on 31 data sets. Both of them significantly outperform SVM with the default parameters. Also, the precomputed kernel does not need parameter tuning, while the parameter tuning for SVMcg is time-demanding.

[5] https://github.com/dmlc/xgboost
[6] https://github.com/rasmusbergpalm/DeepLearnToolbox
[7] http://deeplearning.net/software/pylearn2/

Table 2
Accuracy results for different implementations of SVM.

Data sets	Accuracy		
	Default	'Pre-computed Kernel'	SVMcg.m
Yeast	0.5473	**0.6284**	0.3987
Wine	0.4444	**0.9444**	**0.9444**
Wall_following	0.4451	**0.9854**	0.9377
thyroid	**0.9028**	**0.9028**	**0.9028**
shuttle	0.7558	**0.9585**	0.7558
Seeds	**0.9048**	0.8571	0.8571
Satimage	0.2205	0.9037	**0.9130**
pima	0.5974	0.6494	**0.7013**
penbased	0.0545	0.9727	**0.9818**
page-blocks	0.8519	**0.9444**	0.8889
Optdigits	0.0801	0.9822	**0.9911**
new_thyroid	0.7273	**1.0000**	0.8636
...	...	...	...
Ranking	**Number of data sets**		
# of wins (ties)	8 (2)	**40 (5)**	31 (5)
# of ties		6	
# of triple ties		1	

Table 4
Accuracy results for different settings of Random Forests.

Data sets	Accuracy	
	RF-default	RF-fine-tuned
Yeast	**0.6351**	0.6216
Wine	**1**	0.9444
Wall_following	0.9963	**0.9982**
thyroid	0.9444	**1**
shuttle	**0.9954**	**0.9954**
Seeds	0.8095	**0.8571**
Satimage	0.9162	**0.9270**
pima	**0.8052**	**0.8052**
penbased	0.9636	**0.9727**
page-blocks	0.9074	**0.9259**
Optdigits	0.9787	**0.9822**
new_thyroid	**0.9546**	**0.9546**
...	...	...
Ranking	**Number of data sets**	
# of wins	25	**37**
# of ties	9	

Table 3
Accuracy results for different parameter settings of ELM.

Data sets	Accuracy		
	ELM	80%-train	90%-train
Yeast	0.5608	**0.6487**	0.6351
Wine	0.6667	**0.7222**	**0.7222**
Wall_following	0.5586	0.5916	**0.6099**
thyroid	**0.9028**	**0.9028**	**0.9028**
shuttle	**0.9908**	**0.9908**	**0.9908**
Seeds	**0.9048**	**0.9048**	**0.9048**
Satimage	0.9255	**0.9286**	0.9239
pima	0.7013	0.6623	**0.7273**
penbased	**0.9727**	**0.9727**	**0.9727**
page-blocks	**0.8889**	0.8704	0.8704
Optdigits	**0.9822**	**0.9822**	**0.9822**
new_thyroid	0.9091	**0.9546**	0.9091
...	...	...	...
Ranking	**Number of data sets**		
# of wins (ties)	31 (5)	29 (8)	**42 (13)**
# of ties		13	
# of triple ties		9	

Therefore, we finally use the precomputed kernel for our experiments.

For KNN, we try different values for K, such as 1 (King et al., 1995) and 3. We fine-tune, on *validation_data*, the best K value for KNN on each dataset.

For the rest of the algorithms (i.e. C4.5, SRC, LR, AB, NB, DL), we use their default parameters.

5.2.2. Testing setting of each classifier

We try two testing options for GBDT. The only difference between them is that the first method trains the classification model on *training_data* then fine-tunes its parameters using the *validation_data*, whereas in the second method we train a classification model using both *training_data* and *validation_data*, while still using *validation_data* for parameter tuning. That is, *validation_data* is used in both training and parameter tuning. Nevertheless, the overall performance of the two methods is found to be similar, but the first method runs faster, which is why it is selected for the benchmark experiments.

Similarly, for ELM, we also compare the above two fine-tuning methods, but find that the second method has slightly better overall prediction accuracy. In Table 3, we show the accuracy results of ELM using both the default parameters (i.e. without parameter-tuning) and the fine-tuned parameters (i.e. the second and third methods in the table).

Here, *# of wins* corresponds to the number of data sets where one method achieves the first place, regarding prediction accuracy. The values in the parentheses (i.e. *ties*) correspond to the number of data sets where two or more methods share the first place. *# of ties* (i.e. *13* in Table 3) is the number of data sets where only two methods are tied in the first place. Lastly, *# of triple ties* denotes the number of data sets where all the methods have identical accuracy results. For example, $\#\ of\ triple\ ties\ =\ 9$ in Table 3 implies that there are 9 data sets (e.g. *shuttle* and *penbased*) where the accuracy results are the same for all the three methods. Due to the ties between different methods, the total number of the data sets for *# of wins* can be larger than the number of the data sets used in the experiments, e.g. in Table 3, $31+29+42-13-9\times 2=71$.

It is clear that ELM with fine-tuned parameters (i.e. second and third methods) significantly outperforms ELM with default parameters in most cases. As the third method achieves better accuracy in more cases than the second, while also yielding better total average accuracy, it is selected as the representative ELM setting for the benchmarks.

For RF, we compare the algorithm with default parameters and the fine tuned version derived using the same tuning methodology as GBDT and ELM. We find that the fine-tuned RF has better prediction accuracy in 16.9% more data sets than the default one, as shown in Table 4. It should be noted that the *# of wins* denotes the number of data sets where an algorithm setting outperforms the other, while *# of ties* denotes the number of data sets where the two settings yield identical accuracy.

Table 5 shows the accuracy comparison of different DL implementations. It should be noted that, while Neural Networks (NN) works on every data set, Deep Belief Networks (DBN) and Pylearn2 require non-negative matrices, and Pylearn2 also requires non-negative integers, which represent the pixel values of the images, thus some data sets are not testable for them. In the 20 data sets where all these implementations are valid, they have similar performance in terms of best accuracy. NN, DBN, and *Pylearn2* achieve *top-1* accuracy on 8, 10, and 8 data sets respectively. We finally decide to use NN to compare with other classifiers. It is worth noting that DL does not show satisfactory results on most of the data sets, it has shown outstanding prediction performance on image-data though.

Table 5
Accuracy results for different implementations of Deep Learning.

Data sets	Accuracy		
	NN	DBN	Pylearn2
Yeast	0.1622	**0.3311**	N/A
Wine	**0.2778**	**0.2778**	N/A
Wall_following	**0.4542**	0.3810	0.2857
thyroid	**0.9028**	**0.9028**	N/A
shuttle	**0.7558**	N/A	0.7512
Seeds	**0.2381**	**0.2381**	N/A
Satimage	**0.2624**	0.2205	0.2562
pima	0.4026	**0.5974**	N/A
penbased	0.0818	**0.1273**	0.0727
page-blocks	**0.8704**	0.8519	N/A
...	...	...	...
Ranking	**Number of data sets**		
# of wins (ties)	**47 (10)**	25 (11)	10 (1)
# of ties		11	
# of triple ties		0	

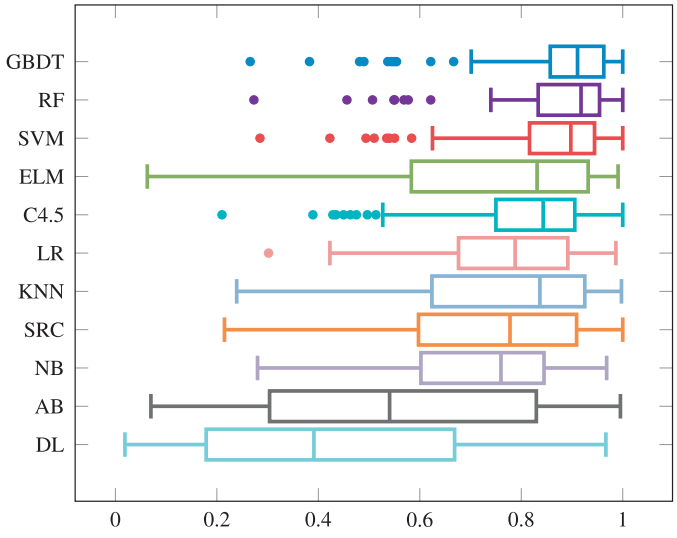

Fig. 4. Box plots for Accuracy (raw values).

6. Results and analysis

6.1. Accuracy comparisons of individual classifiers

In Table 6, we compare the classification accuracy achieved by every classifier on the 71 data sets. In Fig. 4, we depict the averaged accuracy performance of each classifier on all 71 data sets using a box plot (Tukey, 1977). We see that RF and GBDT perform the best, followed by SVM and ELM. Moreover, we observe that the interquartile range of RF, GBDT, and SVM are the smallest, showing that these three algorithms generally perform well, in terms of prediction accuracy, regardless of the data sets. In Fig. 5, we summarise the distributions of average ranks of the accuracy performance for each classifier on all the data sets using a box plot. By average ranks, we mean that we rank the algorithms in terms of accuracy, but in cases where two or more algorithms have identical accuracy values (ties) on the same data set, these algorithms receive a rank equal to the average of the ranks they span, instead of the lower rank. For example, in Table 6, GBDT and ELM are best performers on the data set *Ecoli_reduction*, having accuracy value of 0.8788, using the average rank measure, both algorithms will get a rank value of 1.5, instead of 1. It is clear that GBDT and RF rank high in the majority of the data sets, followed by SVM. Moreover, RF ranks consistently high, as it can be deduced by the compact length of both the interquartile range and the whiskers. The

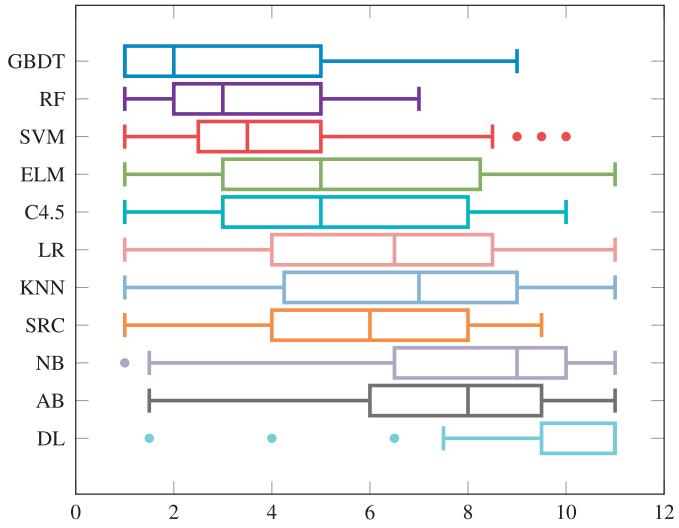

Fig. 5. Box plots for Accuracy (mean ranks).

low ranking of SVM in 3 data sets (*shuttle*, *ionosphere*, and *biodeg*) could be interpreted as outliers. It is easy to see that AB and DL are in general the worst performers, with only a few cases/data sets being the exceptions to the rule.

In Table 7, we summarise Table 6 by ranking the classifiers based on the number of data sets on which a classifier achieves the *top-1*, *top-2*, etc., prediction accuracy. It should be noted that for DL, on each data set we pick the best accuracy out of the three DL implementations (because the accuracy of DL on these 71 data sets is very low), but for any other classifier we pick the version that corresponds to the parameter set or implementation that yields the best overall accuracy.

From Table 7 we can observe that GBDT ranks as the best classifier in 42.25% of the data sets and within *top-2* in 63.38% of all the data sets. Similarly, it ranks *top-3* with a probability of 67.61%, *top-4* with 77.46%, and *top-5* with 83.10%. Comparing this with the remaining classifiers, we can see that GBDT is significantly better, in terms of accuracy, than the other classifiers in *top-1* and *top-2* cases, while still being the second best in *top-3*, *top-4*, and *top-5* cases.

RF ranks second at *top-1* and *top-2*, but it is the best performer in *top-3*, *top-4*, *top-5* cases. The commonly used SVM ranks 3rd in all cases, while ELM ranks overall as 4th, it achieves the best accuracy in 11 out of the 71 datasets. C4.5 and SRC generally rank as the fifth and sixth most accurate classifiers. The rest of the classifiers demonstrate unremarkable results.

Referring only to the *top-1* accuracy for each data set, with a tolerance of 0.2%, 0.5% and 0.8%, respectively, each classifier is ranked based on their accuracy, taking into account the above tolerance values, as can be seen in Table 8. We see that, when the variance tolerance value is 0.5% or smaller, GBDT is *top-1* on accuracy in approximately 50% of all the data sets. In comparison, RF ranks *top-1* in roughly 1/3 of all the data sets, with the same tolerance value; but ELM and SVM rank *top-1* in only 20% of all the data sets.

We check the accuracy results in Van Rijn et al. (2013) in order to validate our experiments. For example, for the *mushroom* data set, our accuracy results in this work for RF, SVM, C4.5, NB, and AB are close to those reported in Van Rijn et al. (2013), being *1, 0.9950, 1, 0.9507, 0.9991* respectively in Van Rijn et al. (2013), while their according values are *0.9951, 1, 1, 0.9570, 0.9668* in this work. For the *Mfeat-kar* data set and for the same algorithms, the accuracy results in Van Rijn et al. (2013) are *0.9511, 0.9213, 0.7685, 0.9299, 0.1940*, while in this work the corresponding values are *0.9500, 0.9750, 0.7800, 0.9400, 0.1800*. These results demon-

Table 6
Accuracy results for different classification algorithms on 71 data sets.

Accuracy	Classifiers										
Data sets	GBDT	RF	ELM	SVM	C4.5	SRC	KNN	LR	AB	NB	DL
1 Yeast	0.6216	0.6216	**0.6487**	0.6284	0.5135	0.5743	0.5473	0.6216	0.4122	0.5946	0.3311
2 Wine	**1**	0.9444	0.7222	0.9444	**1**	0.9444	0.8333	0.8889	0.8889	0.9444	0.2778
3 thyroid	**1**	**1**	0.9028	0.9028	0.9861	0.9028	0.9028	0.9306	0.9306	0.9028	0.9028
4 shuttle	**0.9954**	**0.9954**	0.9908	0.9585	**0.9954**	0.9816	0.9862	0.9724	**0.9954**	0.9401	0.7558
5 Seeds	0.8571	0.8571	0.9048	0.8571	0.7143	0.9048	**0.9524**	0.7619	0.8095	0.8095	0.2381
6 pima	0.7013	0.8052	0.6623	0.6494	0.7662	0.5974	0.6104	0.8052	**0.8312**	0.7533	0.5974
7 penbased	0.9545	0.9727	0.9727	0.9727	0.8818	**0.9818**	0.9636	0.8909	0.1364	0.8727	0.1273
8 page-blocks	0.8889	0.9259	0.8704	**0.9444**	0.9259	0.9074	0.9074	0.9259	0.8889	0.9074	0.8704
9 new_thyroid	0.9545	0.9546	0.9546	**1**	0.9546	0.9091	0.9546	0.9546	0.9546	0.9546	0.7273
10 image_segmentation	**0.9870**	0.9827	0.9827	0.9437	0.9697	0.9827	0.9178	0.9697	0.2597	0.8139	0.0606
11 IIF_Intensity	0.6667	0.7500	0.2833	**0.9500**	0.6667	0.4500	0.8667	0.6500	0.5000	0.6000	0.3667
12 glass_reduction	0.7619	0.8095	**0.9048**	0.8095	0.4286	0.6667	0.7619	0.7143	0.4286	0.3810	0.4286
13 FER	0.8977	0.9432	0.9773	0.8977	0.6591	**1**	0.9773	0.7841	0.3409	0.3636	0.1705
14 Ecoli_reduction	**0.8788**	0.8182	**0.8788**	0.8485	0.8485	0.7576	0.8182	0.7879	0.6667	0.7273	0.3636
15 dermatology	0.9730	0.9460	0.9460	**1**	0.9460	0.9730	0.9189	0.9730	0.5405	0.9460	0.3243
16 contraceptive	**0.5541**	0.5068	0.5338	0.5405	0.5270	0.5405	0.4932	0.5270	0.4122	0.5000	0.4122
17 Cardiotocography	**0.9108**	0.8967	0.7465	0.8545	0.8639	0.7371	0.7183	0.8685	0.3897	0.7136	0.0188
18 car	**1**	0.9711	0.9480	0.9191	0.9538	0.8613	0.8555	0.6763	0.6705	0.7861	0.6705
19 balance	0.9683	0.9524	0.9524	0.9206	0.8571	**1**	0.9524	0.9365	0.8095	0.9683	0.4603
20 autos	**0.8750**	0.8125	0.3750	0.6250	**0.8750**	0.6875	0.3125	0.5625	0.3125	0.3750	0.3125
21 abalone	0.2657	0.2730	0.2874	0.2850	0.2101	0.2150	0.2391	**0.3019**	0.2295	0.2802	0.1377
22 phoneme	0.8669	0.8946	0.8799	0.7745	0.8466	**0.8983**	0.8928	0.7449	0.7708	0.7338	0.2847
23 winequality-white	0.5490	0.5490	0.4694	0.5347	0.4347	0.4327	0.4592	0.5633	**0.5694**	0.3959	**0.5694**
24 winequality-red	0.4813	0.5688	0.4500	0.4938	0.4750	0.5438	0.4313	**0.5875**	0.5750	0.5313	0.4313
25 twonorm-5an-nn	0.9459	0.9432	**0.9487**	**0.9487**	0.8257	0.5243	0.9405	0.9460	0.8500	0.9460	0.9189
26 segment-5an-nn	**0.9784**	0.9740	0.8312	0.8355	0.9437	0.8831	0.7836	0.8918	0.2511	0.8182	0.1732
27 page-blocks-5an-nn	0.9708	**0.9745**	0.9197	0.9380	0.9635	0.9434	0.9398	0.9307	0.9179	0.7664	0.8869
28 spambase-10an-nn	**0.9109**	0.9044	0.7304	0.9044	0.8609	0.7087	0.7044	0.6870	0.8783	0.6000	0.6130
29 segment-10an-nn	**0.9610**	0.9481	0.8095	0.8615	0.8745	0.7879	0.6926	0.7922	0.2771	0.7792	0.1385
30 spambase-15an-nn	0.8935	**0.9000**	0.6761	0.8630	0.8196	0.6739	0.6500	0.6609	0.8522	0.6109	0.3913
31 yeast-20an-nn	0.3826	0.4564	**0.5168**	0.5101	0.3893	0.3222	0.3423	0.4765	0.4698	0.3960	0.3960
32 vowel_context	0.8485	0.9394	**0.9899**	0.9697	0.7879	0.9798	0.9495	0.6970	0.1616	0.6364	0.1111
33 Ionosphere	0.9167	0.9167	0.8889	0.8056	**0.9444**	**0.9444**	0.8889	0.8889	0.9167	0.8056	0.7222
34 german_credit	**0.7600**	0.7400	0.7100	0.7200	0.7400	0.6900	0.7200	0.7200	0.7100	**0.7600**	0.7400
35 splice1	0.9436	0.9342	0.9248	0.9248	**0.9718**	0.8809	0.7743	0.8966	0.8840	0.8715	0.8746
36 Mush	0.8958	0.8333	0.5833	**0.9583**	0.7500	0.8958	0.8750	0.8542	0.7292	0.7708	0.7708
37 Wall_following	**1**	0.9982	0.5916	0.9854	**1**	0.9084	0.8865	0.7070	0.7491	0.5202	0.4542
38 Satimage	0.9270	0.9270	**0.9286**	0.9037	0.8789	0.6615	0.8463	0.8603	0.4550	0.8012	0.2624
39 Optdigits	0.9786	0.9822	0.9822	0.9822	0.8968	**0.9875**	0.9840	0.9448	0.1868	0.9235	0.0996
40 hayes_roth	**0.7857**	**0.7857**	**0.7857**	**0.7857**	**0.7857**	0.6429	0.5000	0.6429	0.2143	**0.7857**	0.5714
41 BioCells	0.7959	0.7959	0.7959	0.8163	**0.8367**	0.7959	**0.8367**	0.7959	0.7755	0.7347	0.7143
42 magic	0.8654	**0.8696**	0.8092	0.8307	0.8507	0.7124	0.7745	0.7944	0.7844	0.7376	0.6583
43 yeast-5an-nn	0.5369	**0.5503**	0.5436	**0.5503**	0.4631	0.3960	0.4430	0.5168	0.4362	0.4430	0.3557
44 spambase-5an-nn	**0.9196**	0.9130	0.7087	0.9152	0.8609	0.7283	0.6891	0.7478	0.8848	0.6413	0.6087
45 satimage-5an-nn	0.8773	**0.8913**	0.6056	0.8665	0.8432	0.6273	0.6444	0.7578	0.4922	0.6320	0.3121
46 penbased-5an-nn	**0.9809**	0.9782	0.9718	0.9536	0.9055	0.9582	0.9682	0.8436	0.2009	0.7836	0.1527
47 yeast-10an-nn	0.5436	0.5772	0.5503	**0.5839**	0.4966	0.4362	0.5034	0.5571	0.4765	0.5369	0.3960
48 twonorm-10an-nn	0.9486	0.9460	0.9419	**0.9500**	0.8216	0.5162	0.9216	0.9297	0.8297	0.9324	0.8757
49 satimage-10an-nn	**0.8975**	0.8913	0.5093	0.8820	0.7857	0.6289	0.5776	0.6848	0.4565	0.6149	0.1817
50 penbased-10an-nn	**0.9627**	**0.9627**	0.9318	0.9091	0.8600	0.9064	0.9300	0.8064	0.1864	0.7591	0.1518
51 page-blocks-10an-nn	**0.9745**	0.9708	0.9069	0.9234	0.9580	0.9343	0.9252	0.8978	0.9215	0.8102	0.8923
52 yeast-15an-nn	**0.4899**	0.4564	0.4161	0.4228	0.4497	0.4564	0.4094	0.4228	0.4094	0.4094	0.3557
53 twonorm-15an-nn	**0.9243**	**0.9243**	0.9027	0.9189	0.7716	0.4946	0.8865	0.8946	0.8189	0.8960	0.8527
54 segment-15an-nn	0.9307	**0.9351**	0.6450	0.8268	0.8442	0.6970	0.5584	0.7359	0.2814	0.6926	0.1342
55 satimage-15an-nn	**0.8820**	0.8758	0.4099	0.8649	0.7438	0.5776	0.4534	0.6304	0.4565	0.5761	0.3121
56 penbased-15an-nn	0.9355	**0.9427**	0.8891	0.8973	0.8109	0.8791	0.8909	0.7518	0.1909	0.7600	0.1455
57 twonorm-20an-nn	0.9095	0.9027	0.8960	**0.9135**	0.7865	0.5216	0.8554	0.8703	0.8014	0.8770	0.7987
58 spambase-20an-nn	**0.9087**	0.9022	0.6478	0.8739	0.8413	0.6587	0.6239	0.6587	0.8609	0.6022	0.3913
59 segment-20an-nn	0.9177	**0.9178**	0.5498	0.8355	0.8225	0.6191	0.5498	0.7792	0.2684	0.7143	0.1429
60 satimage-20an-nn	**0.8820**	0.8680	0.4146	0.8339	0.7143	0.5575	0.4301	0.5932	0.3168	0.5901	0.2019
61 penbased-20an-nn	**0.9200**	0.9064	0.8464	0.8300	0.7655	0.7882	0.8455	0.6909	0.1764	0.6982	0.1345
62 Letter	0.9470	0.9600	**0.9725**	0.9280	0.8720	0.7795	0.9495	0.7715	0.0700	0.6140	0.0425
63 gesture_phase	**0.9771**	0.9543	0.0629	**0.9771**	0.9371	0.4114	0.8971	0.7771	0.6800	0.7257	0.4114
64 mushroom	**1**	0.9951	0.9779	**1**	**1**	**1**	0.9975	0.9865	0.9668	0.9570	0.9668
65 Waveform	0.8520	0.8580	**0.8640**	0.8340	0.7340	0.7560	0.7960	0.8460	0.6120	0.7880	0.3140
66 biodeg	**0.8868**	0.8679	0.8491	0.7547	0.8396	0.8208	0.8019	0.8679	0.8208	0.7830	0.6415
67 Sonar	0.9048	**0.9524**	0.6191	0.9048	0.7619	0.8571	0.7143	0.6667	0.8571	0.7619	0.6667
68 Mfeat-kar	0.9250	0.9500	**0.9800**	0.9750	0.7800	0.9750	0.9700	0.8800	0.1850	0.9400	0.4350
69 Gas	**0.9778**	**0.9778**	0.0889	0.9111	0.9333	0.7778	0.6889	0.9556	0.3778	0.7778	0.3111
70 Mfeat-fac	0.9700	0.9750	0.5450	0.9800	0.8750	0.9750	0.9600	**0.9850**	0.1650	0.9350	0.1250
71 usps	0.9292	0.9317	0.9582	0.9447	0.8680	**0.9656**	0.9467	0.8914	0.3039	0.7678	0.1789

Table 7
Accuracy ranking of different classification algorithms.

Algorithms	Accuracy ranking				
	Top-1	Top-2	Top-3	Top-4	Top-5
GBDT	**42.25%**	**63.38%**	67.61%	77.46%	83.10%
RF	21.13%	56.34%	**77.46%**	**84.51%**	**92.96%**
SVM	18.31%	30.99%	52.11%	64.79%	78.87%
ELM	15.49%	23.94%	30.99%	47.89%	56.34%
C4.5	12.68%	15.49%	28.17%	43.66%	56.34%
SRC	11.27%	22.54%	23.94%	33.80%	38.03%
LR	4.23%	11.27%	19.72%	26.76%	40.85%
AB	4.23%	7.04%	8.45%	19.72%	18.31%
KNN	2.82%	8.45%	12.68%	26.76%	36.62%
NB	2.82%	5.63%	7.04%	9.86%	19.72%
DL	1.41%	1.41%	2.82%	4.23%	4.23%

Table 8
Best accuracy ranking with variance (in percentage).

Algorithms	Accuracy ranking			
	no var	0.2% var	0.5% var	0.8% var
GBDT	42.25%	46.48%	53.52%	56.34%
RF	21.13%	23.94%	33.80%	45.07%
SVM	18.31%	18.31%	19.72%	26.76%
ELM	15.49%	15.49%	18.31%	21.13%
C45	12.68%	12.68%	12.68%	12.68%
SRC	11.27%	11.27%	12.68%	14.08%
LR	4.23%	4.23%	5.63%	5.63%
AB	4.23%	4.23%	4.23%	4.23%
KNN	2.82%	2.82%	5.63%	7.04%
NB	2.82%	2.82%	4.23%	4.23%
DL	1.41%	1.41%	1.41%	1.41%

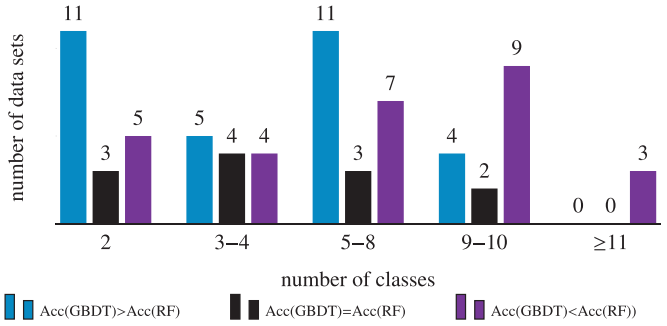

Fig. 6. Accuracy comparison between GBDT and RF grouped by the number of classes.

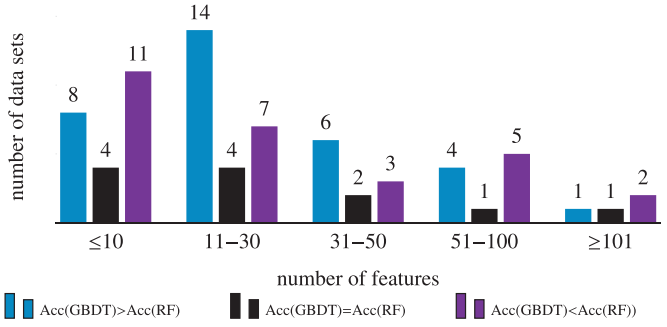

Fig. 7. Accuracy comparison between GBDT and RF grouped by the number of features.

strate that our results are in general consistent with those reported in Van Rijn et al. (2013). It should be noted that, the experimental results for GBDT, ELM, and SRC are not available in Van Rijn et al. (2013).

However, we have also observed inconsistent results, such as the *page-blocks* data set, where the accuracy results for the above-mentioned algorithms, as reported in Van Rijn et al. (2013), are *0.9762, 0.9123, 0.9700, 0.8990, 0.9313*, while in this work the corresponding values are *0.9259, 0.9444, 0.9259, 0.9074, 0.8889*. This inconsistency occurs due to the different parameters tuned/adopted, and the specific instances selected for training/testing can also be different.

In order to demonstrate the difference in prediction accuracy between classifiers, we make pairwise comparisons. However, comparison between classifiers over multiple data sets using just the average accuracy values is not statistically safe, because these values are not commensurable (Demšar, 2006). Nevertheless, the mean values give a rough estimate for the performance of the classifiers and summary results can be used as a guide. Demšar (2006) and Garcia and Herrera (2009) recommend pair significance testing for comparative studies. Moreover, because the accuracy values over all data sets for each classifier do not follow the normal distribution, significance tests should be non-parametric. For all the pair-wise comparisons, we apply the Friedman non-parametric test for multiple groups. This test either retains or rejects the null-hypothesis test of whether these accuracy values come from one distribution or not. In our tests, we set the confidence level at 95%. Then, we follow with post-hoc analysis using Dunn's test, as explained in Section 4.1. For significance level $\alpha = 0.05$, $k = 11$ classifiers, and $n = 71$ matched data points (corresponding to 71 data sets), we find that the corresponding critical value of a two-tailed hypothesis test is $z = 3.3173$ and the comparison between a pair of classifiers is statistically significant if the absolute difference of their sum ranks is larger than 127.5476. The results of Dunn's test are listed in Table 9. We observe that, the difference between GBDT and RF/SVM is not statistically significant, while the difference between GBDT/RF and ELM is significant. With a confidence level of 95%, only the differences between ELM and AB/ND/DL are significant. At the same confidence level, SVM sum ranking is statistically better than SRC, KNN, LR, AB, NB, and DL, but we can not make this claim in pair-wise comparisons with GBDT, RF, ELM, and C4.5.

6.1.1. GBDT vs RF

In Fig. 6 we compare the influence of the number of classes on the prediction accuracy between GBDT and RF. We can observe that when the number of classes is 8 or less GBDT outperforms RF, while RF has better prediction accuracy when the number of classes is 9 or more.

From Table 6 we can also observe that in 5 data sets, *pima, IIF_Intensity, winequality-red, yeast-20an-nn*, and *vowel_context*, the accuracy difference between RF and GBDT is atypically large in favour of RF, over 10%. However, except these 5 data sets, GBDT outperforms RF in mean overall accuracy.

Regarding the influence of the number of features on the accuracy performance, RF performs better when the number of features is 10 or less, but GBDT is exceptional when the number of features is between 11 and 50, as it can be seen in Fig. 7. When the number of features is greater than 50, both classifiers have similar ranking performance.

Using non-parametric test, with a confidence level of 95%, as can be seen in Table 9, we find that their ranking differences are statistically insignificant.

6.1.2. GBDT vs SVM

In Fig. 8 we compare the influence of the number of classes on the prediction accuracy between GBDT and SVM. We can observe that when the number of classes is 8 or less GBDT significantly outperforms SVM, while SVM has better prediction accuracy when the number of classes is 9 or more.

Table 9
Dunn's Multiple Comparisons Test for 11 classifiers on 71 data sets (confidence level: 95%).

Classifiers	Rank sum differences									
	RF	ELM	SVM	C45	SRC	KNN	LR	AB	NB	DL
GBDT	**−1.0**	180.0	**65.5**	177.0	216.5	250.0	197.5	364.0	323.5	504.0
RF		181.0	**66.5**	178.0	217.5	251.0	198.5	365.0	324.5	505.0
ELM			**−114.5**	**−3.0**	**36.5**	**70.0**	**17.5**	184.0	143.5	324.0
SVM				**111.5**	151.0	184.5	132.0	298.5	258.0	438.5
C4.5					**39.5**	**73.0**	**20.5**	187.0	146.5	327.0
SRC						**33.5**	**−19.0**	147.5	**107.0**	287.5
KNN							**−52.5**	**114.0**	**73.5**	254.0
LR								166.5	**126.0**	306.5
AB									**−40.5**	140.0
NB										180.5

Bold values correspond to non statistically significant comparisons (absolute rank sum differences that are < 127.5476).

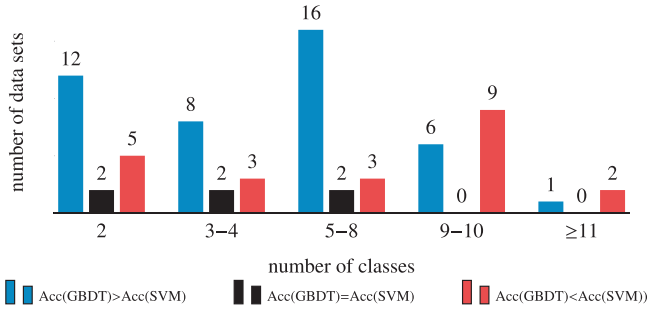

Fig. 8. Accuracy comparison between GBDT and SVM grouped by the number of classes.

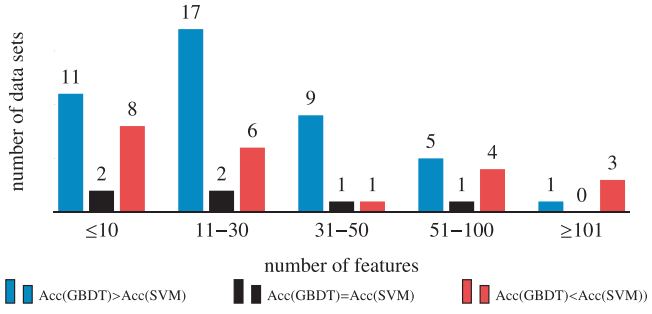

Fig. 9. Accuracy comparison between GBDT and SVM grouped by the number of features.

Regarding the influence of the number of features on the accuracy performance, GBDT performs significantly better when the number of features is 50 or less but SVM is exceptional when the number of features is more than 100, as it can be seen in Fig. 9. Both classifiers show similar performance when the number of features is between 50 and 100.

We observe that for the *IIF_Intensity, winequality-red*, and *yeast-20an-nn* data sets, SVM has unexpected superior accuracies than GBDT, but the former performs very poorly in the *autos* data set.

Using non-parametric test, with a confidence level of 95%, as can be seen in Table 9, we find that these differences are statistically significant.

6.1.3. GBDT vs ELM

From Table 6, it can be seen that GBDT clearly outperforms ELM. The total average accuracy for GBDT over 71 data sets is 86.03% while for DL it is 74.47% over 71 data sets. In particular, in 17 data sets out of 71, the difference is extremely large, over 30%.

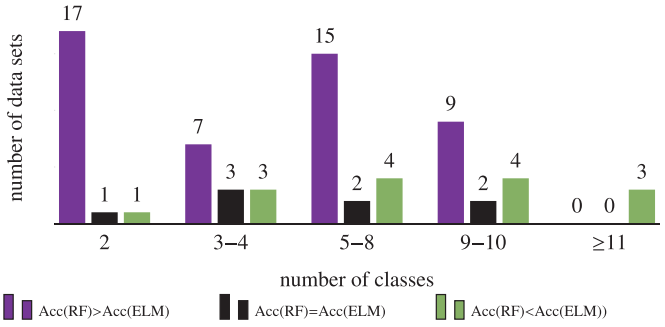

Fig. 10. Accuracy comparison between RF and ELM grouped by the number of classes.

Using Friedman's non-parametric test, with a confidence level of 95%, as can be seen in Table 9, we find that these differences are statistically significant.

6.1.4. GBDT vs DL

From Table 6, it can be seen that GBDT clearly outperforms DL. The total average accuracy of GBDT over 71 data sets is 86.03% while for Deep Learning it is 43.42%. There are only 2 exceptions; DL has better accuracy in *winequality-white* and *yeast-20an-nn* with an accuracy increase of 1.4% and 2%.

Using Friedman non-parametric test, with a confidence level of 95%, as can be seen in Table 9, we find that these differences are statistically significant.

These findings reveal that although DL is a very hot approach for computer vision and speech recognition, its performance is far beyond GBDT for classification tasks on small numerical data sets.

6.1.5. ELM vs RF and C4.5

Using Friedman's non-parametric test, with a confidence level of 95%, as can be seen in Table 9, we find that the differences between ELM and RF are statistically significant. In terms of mean accuracy and number of winning data sets, RF is much better than ELM.

In Fig. 10 we compare the influence of the number of classes on the prediction accuracy between RF and ELM. We can observe that when the number of classes is 10 or less RF outperforms ELM, while ELM has better prediction accuracy when the number of classes is 11 or more.

In Fig. 11, we observe that RF always performs significantly better, regardless of the number of features, especially when this number is between 11 and 30, and 51 and 100.

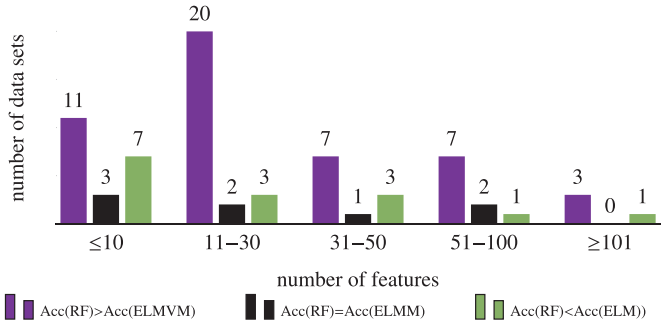

Fig. 11. Accuracy comparison between RF and ELM grouped by the number of features.

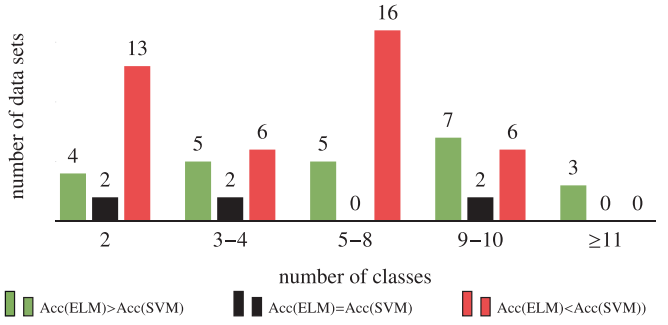

Fig. 12. Accuracy comparison between ELM and SVM grouped by the number of classes.

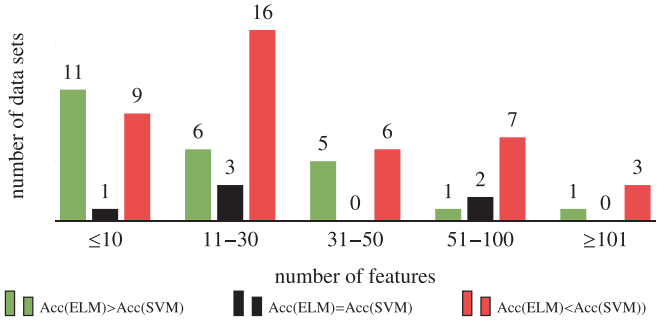

Fig. 13. Accuracy comparison between ELM and SVM grouped by the number of features.

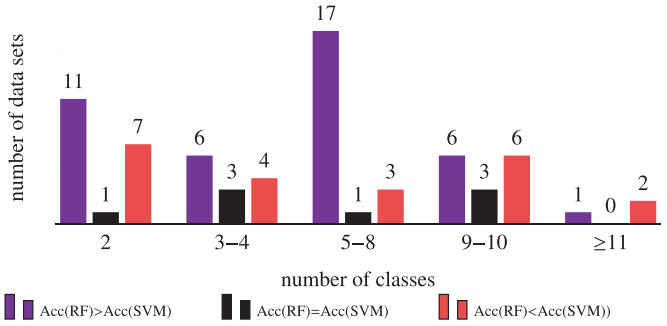

Fig. 14. Accuracy comparison between RF and SVM grouped by the number of classes.

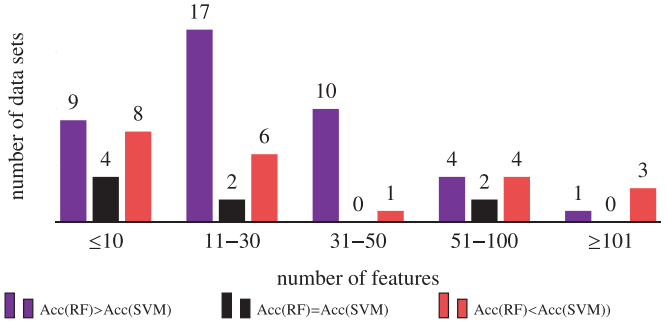

Fig. 15. Accuracy comparison between RF and SVM grouped by the number of features.

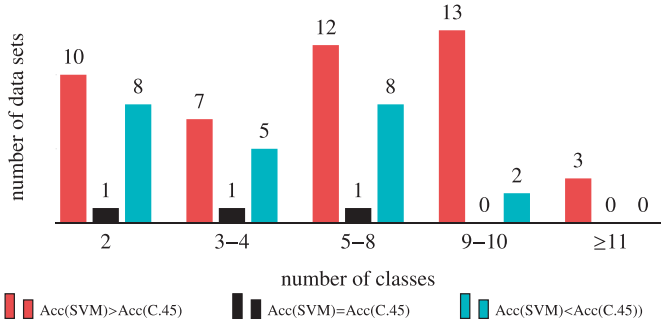

Fig. 16. Accuracy comparison between SVM and C4.5 grouped by the number of classes.

If we exclude *IIF_Intensity* and *gesture_phase*, ELM performs slightly better than C4.5, but this difference is not significant according to Friedman test, as can be seen in Table 9.

6.1.6. ELM vs SVM

In Fig. 12, we compare the influence of the number of classes on the prediction accuracy between ELM and SVM. We can observe that when the number of classes is 8 or less SVM outperforms ELM, while the latter has better prediction accuracy when the number of classes is 9 or above.

In Fig. 13, we observe that SVM has significantly better performance when the number of features is between 11 and 30 or greater than 51. For all other cases the two classifiers show similar performance.

6.1.7. RF vs SVM

In Fig. 14, we compare the influence of the number of classes on the prediction accuracy between RF and SVM. We can observe that RF ranks higher in more data sets with less than 8 classes, while the two algorithms rank similarly when the number of classes is 9 or above. The difference in performance is wider when the number of classes is between 5 and 8.

In Fig. 15, we observe that RF ranks significantly better than SVM when the number of features is between 11 and 50. When the number of features is 10 or less or between 51 and 100 the two algorithms demonstrate similar ranking performance, while SVM is better when the number of features exceeds 101.

6.1.8. SVM vs C4.5

In Fig. 16 we compare the influence of the number of classes on the prediction accuracy between SVM and C4.5. We can observe that SVM consistently ranks better, with the gap in ranking performance being significant when the number of classes exceeds 9.

In Fig. 17 we observe that SVM outperforms C4.5 in all cases, especially when the number of features is greater than 31.

6.2. Accuracy comparisons among groups of classifiers

Table 10 demonstrates the grouped accuracy results of different groups of classifiers. For example, when GBDT and RF are consid-

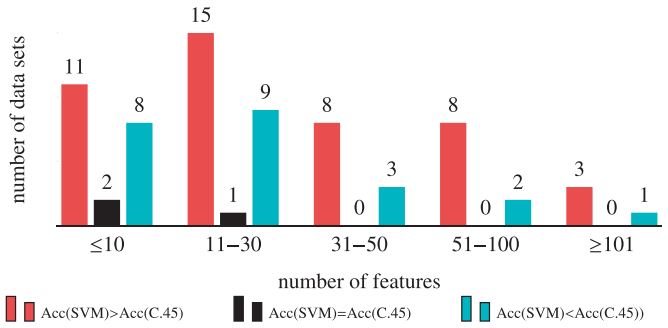

Fig. 17. Accuracy comparison between SVM and C4.5 grouped by the number of features.

Table 10
Accuracy ranking of different classification algorithms groups.

Algorithms grouped	Accuracy ranking	
	Top-1	Top-2
GBDT & RF	***54.93***%	77.46%
GBDT & SVM	**56.34**%	***80.28***%
GBDT & ELM	***54.93***%	**83.10**%
GBDT & C4.5	46.48%	69.01%
RF & SVM	36.62%	76.06%
RF & ELM	35.21%	70.42%
RF & C4.5	30.99%	66.20%
SVM & ELM	30.99%	46.48%
SVM & C4.5	28.17%	42.25%
ELM & C4.5	26.76%	38.03%
GBDT & RF & SVM	**67.61**%	85.92%
GBDT & RF & ELM	**67.61**%	**90.14**%
GBDT & RF & C4.5	59.15%	80.28%
GBDT & SVM & ELM	**67.61**%	**90.14**%
GBDT & SVM & C4.5	***60.56***%	83.10%
GBDT & ELM & C4.5	59.15%	***87.32***%
RF & SVM & ELM	49.30%	84.51%
RF & SVM & C4.5	45.07%	84.51%
RF & ELM & C4.5	45.07%	80.28%
SVM & ELM & C4.5	40.85%	56.34%
GBDT & RF & SVM & ELM	**78.87**%	**92.96**%
GBDT & RF & SVM & C4.5	***71.83***%	88.73%
GBDT & RF & ELM & C4.5	***71.83***%	**92.96**%
GBDT & SVM & ELM & C4.5	***71.83***%	**92.96**%
RF & SVM & ELM & C4.5	57.75%	**92.96**%
GBDT & RF & SVM & ELM & C4.5	***83.10***%	**95.77**%
GBDT & RF & SVM & ELM & SRC	**88.73**%	**95.77**%
GBDT & RF & SVM & ELM & KNN	81.69%	**95.77**%
GBDT & RF & SVM & ELM & C4.5 & KNN	***84.51***%	**97.18**%
GBDT & RF & SVM & ELM & SRC & KNN	**91.55**%	**97.18**%
GBDT & RF & SVM & ELM & C4.5 & SRC	**91.55**%	**97.18**%

ered together, the probability when either of them ranks *top-1* in Table 6 is 54.93%, while the probability is 77.46% if any of them ranks *top-2*. We examine all valid groups of two, three, four, five, and six classifiers. It should be noted that the table does not show the groups of classifiers where the combined performance is insignificant.

In Table 10, we observe that, quite surprisingly, for the group of two classifiers, the group with GBDT and SVM is the best in terms of accuracy at the *top-1* case and the group with GBDT and ELM is the best at the *top-2* case, despite the fact that RF has demonstrated better rankings than both SVM and ELM in *top-1* and *top-2* cases as individual classifier, which can be seen in Table 7. It can be seen from Table 10 that the group of GBDT & SVM has 56.34% at *top-1* and 80.28% at *top-2* accuracy performance.

As for the groups of three or more classifiers we see that the best groups always contain both GBDT and ELM. The two best groups of 3 classifiers, which have GBDT, ELM, and either RF or SVM show the best results: 67.61% at *top-1* and 90.14% at *top-2*.

The best performing groups of 4 or more classifiers always contain GBDT, RF, SVM, and ELM, which are the best 4 individual classifiers in terms of accuracy. For groups of 4, GBDT & RF & SVM & ELM performs best, with 78.87% at *top-1* and 92.96% at *top-2*.

For groups of 5, the one having GBDT & RF & SVM & ELM & SRC performs the best, with 88.73% at *top-1* and 95.77% at *top-2*. Finally, for groups of 6, the two groups having GBDT & RF & SVM & ELM & SRC and either C4.5 or KNN perform best, with 91.55% at *top-1* and 97.18% at top-2.

To summarise, for the selected 71 data sets and the 11 algorithms under test, with high probability, it is possible to achieve the most accurate prediction by considering just a small subset of the 11 algorithms, instead of exhaustively checking/testing all of them. For instance, the group of GBDT & RF & ELM & SVM has a probability of 78.87% to contain the best classifier. Therefore, this study can guide the practitioners, engineers and researchers to promptly find the most accurate classifier for their specific applications/data.

6.3. Influence of the number of classes and features on the prediction performance

In Fig. 18 we present, for each classifier, the number of data sets where this classifier has the best prediction accuracy, with respect to the number of classes in the data sets. We observe that in general GBDT has the best accuracy, independent of the number of classes, while the performance of ELM is exceptional when the data set has more than 10 classes. RF comes third in most cases, while SVM is exceptional only when the number of classes is 3 or less, but still not as good as GBDT. For the remaining classifiers, C4.5 performs better when the number of classes is 5 or less. In general, for two-classes data sets, GBDT, SVM, RF, ELM and C4.5 have the best classification accuracies. When the number of classes reaches 10 and above, the first 4 classifiers remain the most accurate ones, but C4.5 is no longer among them. Generally speaking, GBDT is the classification algorithm that has the largest number of data sets having the best accuracies. It should be noted that, when the number of classes is 10 and above, the accuracy of ELM is as good as that of GBDT.

In Fig. 19, we investigate the influence of the number of features to the classification performance. We see that, when the number of features is smaller than 60, GBDT is the best classifier in terms of accuracy. More importantly, it is significantly better than the other classifiers when the number of features is less than 40. Specifically, when the number of features is below 20. GBDT, SVM, RF, and ELM have the best classification accuracies (it should be noted that the best classification accuracies of different classification algorithms may be the same). When the number of features is between 20 and 40, GBDT remains the best classifier, whereas SVM, RF, ELM and C4.5 are generally among the next best classification algorithms. When the number of features is less than 40, GBDT is the best classifier, having the *top-1* accuracies on 25 data sets, i.e. 45.4% of all the data sets in this range. RF, SVM, and ELM are the second best classification algorithms at this range, showing *top-1* accuracies on 21.82%, 20.00%, and 18.18% of the data sets, respectively. When the number of features is greater than 40, the number of *top-1* accuracy results for GBDT, SVM, RF, and C4.5 are approximately the same. But ELM shows outstanding classification performance when the number of features is over 60.

6.4. AUC Comparisons of individual classifiers

In Table 11, we show the AUC values for the 11 classification algorithms that we examine in this work in the first 36 data sets. These values are depicted using box plots in Fig. 20. We observe that GBDT, RF, and SVM remain the top 3 performers in AUC, while

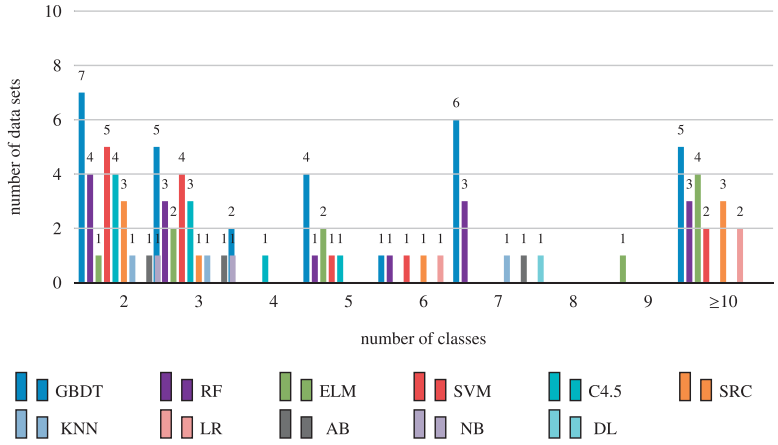

Fig. 18. The number of data sets on which each classifier achieves the best accuracy, grouped by the number of classes.

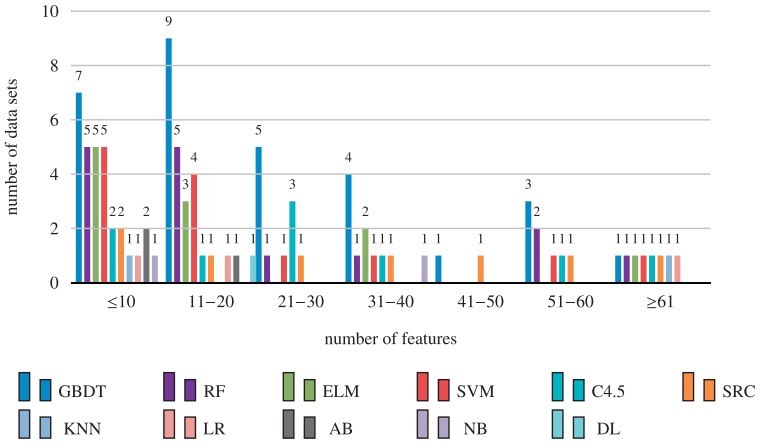

Fig. 19. The number of data sets on which each classifier achieves the best accuracy, grouped by the number of features.

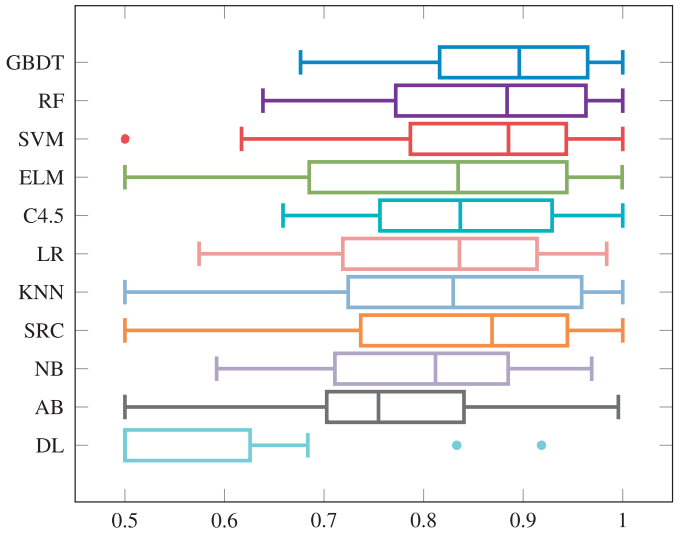

Fig. 20. Box plots for AUC (raw values).

DL, AB, and NB are still among the worst performers. It is noticeable that, in contrast with ACC in Fig. 4, SRC performs better in AUC than ELM, LR, and KNN. We can detect an outlier for SVM that corresponds to *thyroid*, a data set that has both categorical and ordinal variables. While SVM yields 0.9028 accuracy on that data set (see Table 6), the corresponding mean rank ACC value is also an outlier in Fig. 4. On the other hand, DL demonstrates uncharacteristically good performance on the *page–blocks* and *twonorm–5an–nn* data sets, as it can be deduced from Table 11 and Fig. 20.

In Fig. 21, we depict the absolute difference between ACC and AUC mean ranks, for the first 36 data sets. We can visually verify that, with a few exceptions, both AUC and ACC metrics rank these 11 classifiers consistently. Overall, with a variance of 2 in terms of mean rank, the two metrics agree on 81.31% of the $36 \times 11 = 396$ cases.

In the case of DL, we observe that it ranks similarly in both measures but we notice a high discrepancy in *Pageblocks* data set, where DL ranks 1st with regard to AUC while it was ranked 10th in ACC. *Pageblocks* is a highly skewed data set with one majority class and two minority classes. We notice that the corresponding AUC values for each pair of classes for DL are 0.98913, 0.51087, 1, while for SVM (which ranks 1st in ACC but 7th in AUC) the corresponding values are 1, 0.75, 0.5, and for GBDT (which ranks 8th in ACC and 5th in AUC) the values are 0.98913, 0.57428, 0.83333, thus, after averaging over all 3 values, DL ranks first.

This can be attributed to the fact that ACC ignores probability estimations of classification in favour of class labels (Ling, Huang, & Zhang, 2003), that is, the percentage of the correctly predicted instances for all classes is calculated collectively. Therefore, in imbalance datasets, the number of correctly predicted instances that belong to the majority class can dominate the final ACC value. While for AUC, by averaging the pair-wise AUC values, the influence of

Table 11
AUC results for different classification algorithms on 36 data sets.

AUC	Classifiers										
Data sets	GBDT	RF	ELM	SVM	C4.5	SRC	KNN	LR	AB	NB	DL
1 Yeast	**0.8472**	0.8370	0.8296	0.8248	0.8249	0.7992	0.8227	0.8359	0.6754	0.8382	0.5000
2 Wine	**1**	0.9667	0.7192	0.9000	**1**	0.9667	0.7900	0.8733	0.9167	0.9667	0.5000
3 thyroid	**1**	**1**	0.5000	0.5000	0.9167	0.5000	0.5000	0.6667	0.8333	0.6667	0.5000
4 shuttle	**0.9956**	**0.9956**	0.9626	0.9841	**0.9956**	0.9825	0.9796	0.9737	0.9956	0.9175	0.5000
5 Seeds	0.8939	0.8788	0.9394	0.8939	0.7273	0.9394	**0.9545**	0.7455	0.8485	0.8788	0.5000
6 pima	0.6764	0.7468	0.6280	0.6171	0.7307	0.5631	0.5792	0.7738	**0.8061**	0.7251	0.5000
7 penbased	0.9915	**0.9984**	0.9888	0.9888	0.9427	0.9896	0.9834	0.9392	0.7570	0.9386	0.5000
8 page-blocks	0.7989	0.7778	0.6667	0.7500	0.7446	0.8056	0.8056	0.7391	**0.8333**	0.7029	**0.8333**
9 new_thyroid	0.8750	0.8750	0.9688	**1**	0.8750	0.9375	0.9688	0.9688	0.8750	0.9688	0.5000
10 image_segmentation	0.9892	0.9882	**0.9928**	0.9697	0.9858	0.9900	0.9730	0.9839	0.7735	0.9269	0.6504
11 IIF_Intensity	0.7251	0.7561	0.5273	**0.9495**	0.7520	0.6417	0.9454	0.7483	0.7381	0.6508	0.5238
12 glass_reduction	0.7191	0.7477	**0.9815**	0.7901	0.8171	0.8750	0.9630	0.7269	0.8056	0.8148	0.5000
13 FER	0.9027	0.9111	0.9885	0.8659	0.7820	**1**	**1**	0.8409	0.5994	0.6577	0.5000
14 Ecoli_reduction	0.8747	0.9033	0.9058	0.8950	**0.9075**	0.8917	0.8921	0.9058	0.7633	0.8883	0.5000
15 dermatology	0.9800	0.9800	0.9600	**1**	0.9208	0.9800	0.9800	0.9800	0.7524	0.9600	0.6837
16 contraceptive	**0.6933**	0.6386	0.6910	0.6909	0.6661	0.6597	0.6267	0.6663	0.5000	0.6509	0.5000
17 Cardiotocography	0.8824	0.8739	0.8233	0.8621	0.8439	0.8313	0.8239	**0.9220**	0.7356	0.8212	0.5000
18 car	**1**	0.9508	0.9299	0.8970	0.9289	0.8776	0.8356	0.5745	0.5000	0.8657	0.5089
19 balance	0.8333	0.8333	0.8333	0.8667	0.7813	**1**	0.9838	0.9558	0.7238	0.8333	0.5000
20 autos	0.9389	0.8472	0.6792	0.8444	**0.9775**	0.7986	0.7450	0.8361	0.8000	0.7986	0.5000
21 abalone	0.7319	0.7390	0.7116	0.7355	0.6588	0.7276	**0.7598**	0.7108	0.6928	0.7056	0.6296
22 phoneme	0.8438	0.8586	0.8417	0.6508	0.7989	**0.8625**	0.8561	0.6399	0.6717	0.7064	0.5000
23 winequality-white	0.8526	0.7452	0.8360	**0.8795**	0.8298	0.7843	0.7035	0.8343	0.5000	0.8154	0.5000
24 winequality-red	0.6842	0.7662	0.7389	0.7833	0.7603	**0.7889**	0.6882	0.7557	0.7229	0.7750	0.5000
25 twonorm-5an-nn	0.9457	**0.9540**	0.9483	0.9484	0.8258	0.5245	0.9309	0.9457	0.8501	0.9457	0.9184
26 segment-5an-nn	0.9653	0.9687	0.9153	0.8956	**0.9733**	0.9494	0.8941	0.9476	0.7586	0.8934	0.6548
27 page-blocks-5an-nn	0.9642	0.8968	0.7341	0.8910	**0.9670**	0.7536	0.7552	0.7399	0.6714	0.7479	0.5000
28 spambase-10an-nn	0.9027	**0.9594**	0.7024	0.8971	0.8523	0.6702	0.6646	0.6442	0.8717	0.5940	0.5000
29 segment-10an-nn	**0.9706**	0.8889	0.8773	0.9382	0.9292	0.8979	0.8106	0.8825	0.7226	0.8816	0.5274
30 spambase-15an-nn	**0.9024**	0.6702	0.6466	0.8558	0.8171	0.6280	0.6482	0.6163	0.8478	0.5921	0.5000
31 yeast-20an-nn	0.7401	**0.9956**	0.6607	0.7380	0.6910	0.7461	0.6902	0.6370	0.7282	0.7154	0.5000
32 vowel_context	0.9141	0.9345	0.9994	0.9978	0.9144	0.9970	**1**	0.8533	0.7126	0.7906	0.5762
33 Ionosphere	0.8891	0.6443	0.8436	0.8345	**0.9091**	**0.9091**	0.8636	0.8691	0.8891	0.8091	0.6218
34 german_credit	0.6994	**0.9391**	0.6544	0.6452	0.6847	0.6232	0.6939	0.6535	0.6131	0.7408	0.5000
35 splice1	0.9471	0.8765	0.9287	0.9259	**0.9730**	0.8868	0.8092	0.8994	0.8874	0.8736	0.5000
36 Mush	0.8983	**0.9925**	0.5652	0.9583	0.7461	0.8948	0.8930	0.8600	0.7209	0.7696	0.5070

Table 12
AUC ranking of different classification algorithms.

Algorithms	AUC ranking				
	Top-1	Top-2	Top-3	Top-4	Top-5
GBDT	13.89%	**38.89%**	**58.33%**	**69.44%**	**75.00%**
RF	**16.67%**	33.33%	50.00%	66.67%	**75.00%**
SVM	11.11%	25.00%	36.11%	52.78%	66.67%
ELM	5.56%	11.11%	30.56%	33.33%	41.67%
C4.5	13.89%	22.22%	27.78%	41.67%	61.11%
SRC	8.33%	19.44%	27.78%	47.22%	50.00%
LR	8.33%	19.44%	22.22%	33.33%	38.89%
AB	2.78%	5.56%	11.11%	25.00%	38.89%
KN	2.78%	5.56%	8.33%	19.44%	22.22%
NB	0.00%	5.56%	8.33%	13.89%	19.44%
DL	2.78%	2.78%	2.78%	2.78%	2.78%

the majority class instances is limited only to the pairs that contain the majority class(es), but for the pairs that do not contain the majority class, the AUC values can be large, as in the case of DL on *Pageblocks*. After the AUC values are averaged over all the pairs, the final AUC rankings can differ significantly from ACC.

Similarly, we observe on *Abalone* that KNN ranks significantly higher in AUC (1st) than in ACC (10th), while on *winequality-white* AB ranks significantly lower in AUC than in ACC (1st in ACC but 10th in AUC).

In Table 12, we present the top algorithms in AUC ranking, where the top two algorithms, in terms of *top-x* ranking, remain GBDT and RF. However, RF achieves better *top-1* performance than GBDT. Also, C4.5 yields significantly better results, in terms of AUC, than the corresponding ACC results, matching GBDT and outperforming both SVM and ELM in *top-1* AUC performance, which is inconsistent with the findings we draw in Section 6.1, where ELM and SVM rank better than C4.5. But for *top-x*, where $x = 2..5$, C4.5 still ranks lower than GBDT and SVM in AUC, as in the case of ACC.

In Fig. 22, we illustrate the mean rank performance of the 11 algorithms, in terms of AUC. We observe that RF ranks first in AUC mean rank, followed by GBDT and SVM. The next algorithms are ELM and C4.5. Being consistent with the ACC results in Fig. 5, DL, AB, and NB are still the worst performers.

6.5. Running time comparisons between classifiers

In Figs. 23–25, we show each classifier's training time per instance upon the first 36 data sets using box and whisker plots. These classifiers can be divided into three groups with regard to the training time. We see that the classifiers in the first group, i.e. in Fig. 23, are the most efficient algorithms regarding the median of the training times. NB is the fastest classifier in training, followed by KNN, AB, and C4.5. The next fastest classifiers in training are SVM, ELM, and LR, whereas SRC, GBDT, DL and RF are the slowest in training. We also see in Fig. 24 that the efficiency of SVM and ELM are very similar and they show fair running time performance. Finally, the classifiers in the last group (RF, SRC, GBDT), depicted in Fig. 25, are the slowest performers in classifier training (we exclude SRC since it is more than 20 times slower than RF and GBDT in the same group).

We also observe in Table 13 that NB is the fastest classifier in training. The next fastest classifiers in training are KNN, SVM and ELM, whereas SRC, GBDT, DL and RF are the slowest in training.

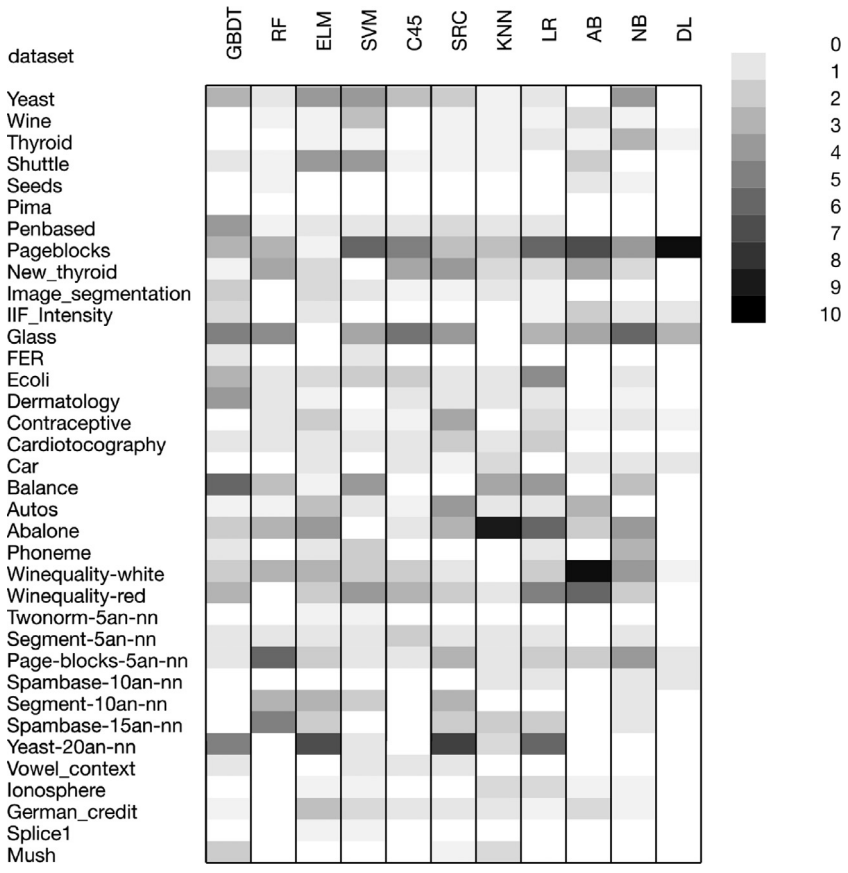

Fig. 21. Cell plot of absolute mean rank difference between ACC and AUC for 36 data sets.

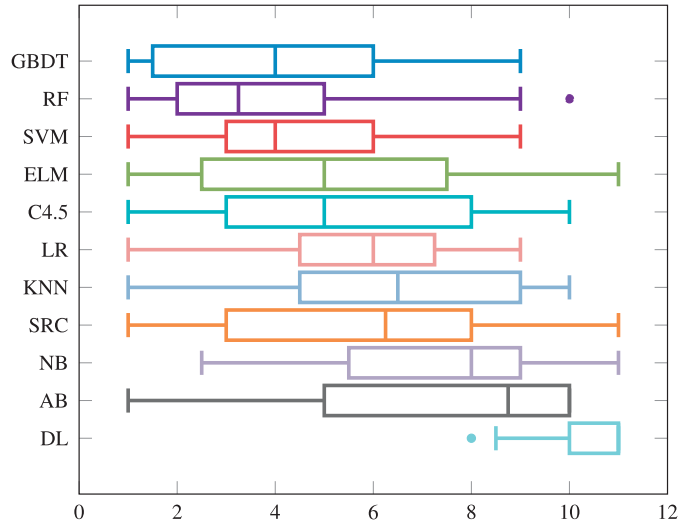

Fig. 22. Box plots for AUC (mean ranks).

However, we see in Table 14 that *GBDT is the fastest classifier in testing*, it is among the slowest classifiers in training though. The next fastest classifiers in testing are DL, SVM and ELM, whereas LR, SRC, NB and RF are the slowest in testing. *It is quite interesting to find NB to be the slowest classifier in testing, although it is the fastest one in training.*

In Figs. 26 and 27, we show each classifier's testing time per instance upon the 36 data sets in seconds using box and whisker plots. These classifiers can be divided into two groups with regard to the testing time. We see that the classifiers in the first group,

Table 13
Training efficiency ranking of different classification algorithms.

Algorithms	Training efficiency ranking				
	Top-1	Top-2	Top-3	Top-4	Top-5
NB	63.89%	86.11%	97.22%	100.00%	100.00%
KNN	11.11%	44.44%	63.89%	72.22%	80.56%
ELM	11.11%	22.22%	33.33%	41.67%	52.78%
SVM	11.11%	19.44%	25.00%	38.89%	55.56%
AB	2.78%	19.44%	50.00%	61.11%	86.11%
C4.5	0.00%	8.33%	19.44%	69.44%	100.00%
LR	0.00%	0.00%	13.89%	16.67%	25.00%
GBDT	0.00%	0.00%	0.00%	0.00%	0.00%
DL	0.00%	0.00%	0.00%	0.00%	0.00%
RF	0.00%	0.00%	0.00%	0.00%	0.00%
SRC	0.00%	0.00%	0.00%	0.00%	0.00%

Table 14
Testing efficiency ranking of different classification algorithms.

Algorithms	Testing efficiency ranking				
	Top-1	Top-2	Top-3	Top-4	Top-5
GBDT	61.11%	94.44%	100.00%	100.00%	100.00%
DL	25.00%	77.78%	83.33%	100.00%	100.00%
SVM	11.11%	22.22%	44.44%	61.11%	83.33%
ELM	2.78%	5.56%	52.78%	69.44%	86.11%
KNN	0.00%	0.00%	19.44%	44.44%	69.44%
C4.5	0.00%	0.00%	5.56%	16.67%	25.00%
AB	0.00%	0.00%	0.00%	5.56%	19.44%
LR	0.00%	0.00%	0.00%	2.78%	2.78%
SRC	0.00%	0.00%	0.00%	0.00%	11.11%
NB	0.00%	0.00%	0.00%	0.00%	2.78%
RF	0.00%	0.00%	0.00%	0.00%	0.00%

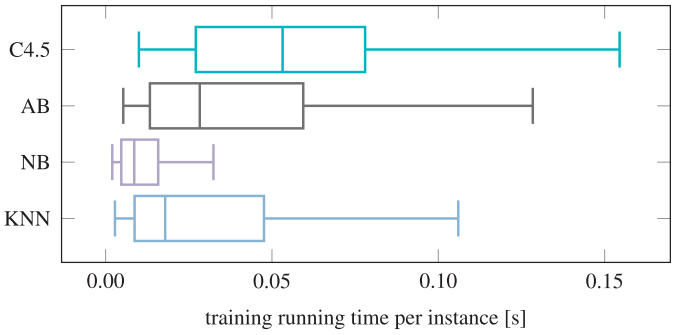

Fig. 23. Box plots for training time efficiency of KNN, NB, AB andC4.5.

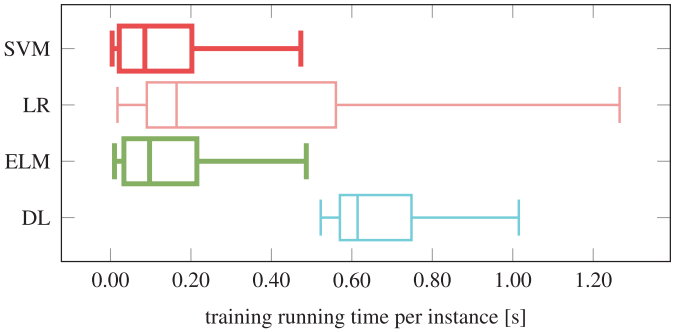

Fig. 24. Box plots for training time efficiency of SVM, LR, ELM and DL.

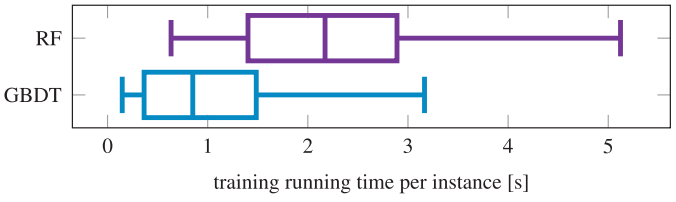

Fig. 25. Box plots for training time efficiency of RF and GBDT.

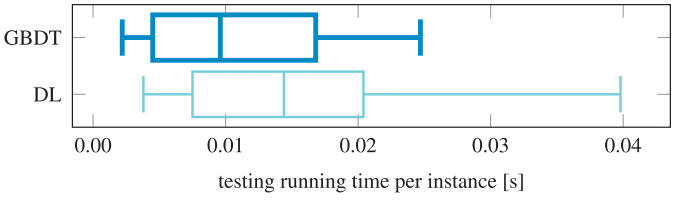

Fig. 26. Box plots for testing time efficiency of GBDT and DL.

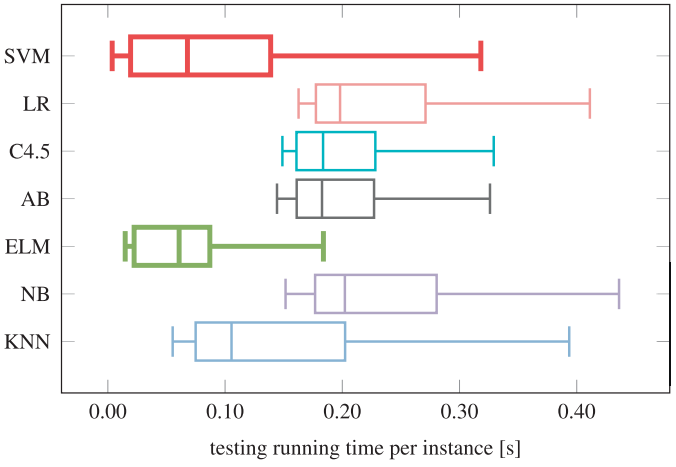

Fig. 27. Box plots for testing time efficiency of different classifiers.

i.e. in Fig. 26, are the most efficient algorithms at the median of the testing times. GBDT is the fastest classifier in testing, followed by DL. We also see in Fig. 27 that the testing efficiencies of SVM and ELM are very similar, followed by KNN, LR, C4.5, AB, and NB. It should be noted that we exclude RF and SRC since both are significantly slower than GBDT and DL in testing time.

In summary, GBDT is among the slowest classifiers in training but it is the fastest classifier in testing. NB is the fastest classifier in training, but it is the slowest classifier in testing. SVM and ELM are the only two classifiers that are fast in both training and testing.

The training, testing, and overall running time, spent by different classifiers on the first 36 data sets are given in Tables 15–17, respectively.

It is worth noted that, in order to achieve the best classification accuracy, ELM needs parameter-tuning, which is very time-demanding. As can be seen in Tables 15 and 16, an average of 98%, 93%, and 99% of the respective running time for GBDT, RF and ELM is spent on parameter-tuning, in order to find the best parameters that achieve the best classification accuracies.

7. Conclusion

In this paper, we provide an update-to-date empirical comparison on the classification prediction performance and time efficiency of 11 state-of-the-art classification algorithms, using publicly available data sets from UCI, KEEL, and LibSVM repositories. The list of classifiers tested in this work includes widely adopted classifiers, in specific Support Vector Machines (SVM) and Random Forests (RF), other well-established classifiers, such as C4.5, AdaBoost (AB), K Nearest Neighbours classifier (KNN), Logistic Regression (LR), and Stochastic Gradient Boosting Trees (GBDT), and algorithms that have been proposed in the recent years, i.e. Extreme Learning Machine (ELM), Sparse Representation based Classification (SRC), and Deep Learning (DL).

While ensemble and boosting methods have been reported to obtain good predictive performance in supervised learning, GBDT is generally less popular than RF and AB, it is typically missing from large-scale comparative studies in the literature and, consequently, tends to be underutilised in machine learning applications. In our experiments, we show that GBDT and RF show both best total average classification accuracy and best mean rank across all 71 data sets, followed by SVM. ELM also yields good accuracy results but this performance varies widely across all data sets.

Using group-wise comparisons, we find that a group of 3 classifiers that includes GBDT, and two out of RF, SVM, and ELM ranks *top-1*, in terms of classification accuracy, in 67.61% of the data sets. Moreover, the group of 4 classifiers that includes GBDT, RF, ELM, and SVM ranks *top-1* in accuracy in 78.87% of the data sets. This is an attractive property, since it is possible to yield high accuracy prediction (i.e. pick the most accurate classifier) by considering only a couple of classifiers instead of exhaustively checking/testing all of them.

We also report the AUC performance for the first 36 data sets. For the AUC metric, we find that GBDT is the best performer, followed by RF and SVM. Generally, both ACC and AUC measures agree on the ranking of the 11 classification algorithms, with GBDT, RF, SVM being the top three performers, followed by ELM and C4.5, while DL, AB, NB rank last.

Unsurprisingly, the top accuracy performers, i.e. GBDT and RF, show average or slow training time efficiency. But GBDT is found to be the fastest algorithm in testing/prediction. DL is the second fastest in prediction efficiency but it is the worst performer in terms of accuracy. SRC shows generally good accuracy performance but it is the slowest classifier in both training and testing. NB is the fastest classifier in training. KNN and SVM are generally the most efficient classifiers in terms of overall running time.

Table 15
Training time (in seconds) efficiency results for different classification algorithms.

Time	Classifiers										
Data sets	GBDT	RF	ELM	SVM	C4.5	SRC	KNN	LR	AB	NB	DL
1 Yeast	1.0521	1.8815	0.1477	0.1671	0.2237	112.6729	0.8284	0.8035	0.0116	0.0759	0.8501
2 Wine	0.1583	0.8334	0.0026	0.0028	0.0059	0.5819	0.0185	0.0230	0.0198	0.0058	0.1198
3 thyroid	0.2260	1.0934	0.0223	0.0143	0.0085	11.4662	0.0171	0.1432	0.0325	0.0063	0.3707
4 shuttle	0.3495	1.2362	0.3174	0.0420	0.0220	332.8577	0.0173	0.2083	0.0694	0.0065	1.0193
5 Seeds	0.1660	0.9347	0.0022	0.0012	0.0054	0.6242	0.0176	0.0304	0.0113	0.0037	0.1007
6 pima	0.1011	1.0754	0.0298	0.1311	0.0137	11.8717	0.0168	0.0150	0.0228	0.0084	0.4486
7 penbased	1.5309	3.3691	0.0874	0.0516	0.0404	26.0356	0.0169	0.4962	0.0109	0.0094	0.5661
8 page-blocks	0.2371	1.1161	0.0147	0.0040	0.0082	5.5503	0.0171	0.0535	0.0261	0.0044	0.2698
9 new_thyroid	0.1470	0.8170	0.0024	0.0008	0.0049	0.5036	0.0171	0.0131	0.0089	0.0034	0.1191
10 image_segmentation	2.4053	5.3743	0.3382	0.0874	0.0990	338.3258	0.0185	5.8898	0.0206	0.0111	1.2510
11 IIF_Intensity	0.8108	1.2178	0.0235	0.1257	0.1022	5.4495	0.0167	31.1400	0.0622	0.0085	0.5025
12 glass_reduction	0.2729	1.0080	0.0021	0.0029	0.0080	0.3505	0.0285	0.0316	0.0054	0.0055	0.1866
13 FER	7.7745	1.3781	0.0403	0.2911	0.2582	29.5491	0.0178	0.8230	0.0349	0.0109	0.5828
14 Ecoli_reduction	0.2636	0.8610	0.0062	0.0038	0.0062	1.0913	0.0173	0.0271	0.0063	0.0039	0.1593
15 dermatology	0.4385	0.8113	0.0059	0.0087	0.0091	1.2382	0.0173	0.1253	0.0064	0.0067	0.2847
16 contraceptive	0.2928	1.1075	0.1522	0.1934	0.0675	92.7528	0.0172	0.0509	0.0091	0.0052	0.7924
17 Cardiotocography	3.0740	6.6743	0.3006	0.2500	0.1313	307.1321	0.0179	11.4475	0.0142	0.0113	1.1344
18 car	0.8128	1.2143	0.2086	0.0545	0.0156	83.2559	0.0176	0.1160	0.0376	0.0047	0.9490
19 balance	0.1812	0.8561	0.0194	0.0065	0.0091	4.8639	0.0173	0.0253	0.0083	0.0039	0.3078
20 autos	0.3018	0.8433	0.0014	0.0066	0.0078	0.4576	0.0173	0.1866	0.0061	0.0039	0.0966
21 abalone	6.5284	9.4116	0.9965	1.1806	0.5153	2146.1000	0.0191	4.0051	0.0199	0.0104	2.2184
22 phoneme	0.7393	4.1837	1.8870	0.3044	0.1052	3529.8000	0.0186	0.0848	0.1326	0.0095	2.7510
23 winequality-white	5.7383	12.0639	1.5286	3.9913	0.4731	3145.4000	0.0183	1.3534	0.0235	0.0171	2.7646
24 winequality-red	0.7510	1.9693	0.1914	0.3144	0.0866	129.8226	0.0175	0.2414	0.0194	0.0067	0.9118
25 twonorm-5an-nn	2.3468	6.6080	3.8522	1.3126	0.6078	5977.4000	0.0187	0.3970	0.6513	0.0314	3.9248
26 segment-5an-nn	2.0823	4.4454	0.4346	0.1936	0.1302	338.0312	0.0181	1.2555	0.0269	0.0107	1.2745
27 page-blocks-5an-nn	3.4631	6.2812	2.7946	0.7769	0.2371	3852.6000	0.0251	0.7576	0.0838	0.0125	2.6745
28 spambase-10an-nn	3.4054	5.0384	2.0552	3.4599	0.4122	2394.1000	0.0208	0.3508	0.5006	0.0385	3.1951
29 segment-10an-nn	4.4911	4.4949	0.5277	0.1619	0.1522	324.8562	0.0180	0.8963	0.0300	0.0132	1.1634
30 spambase-15an-nn	1.8127	6.0469	1.9452	5.0882	0.4850	2428.8000	0.0199	0.2272	0.5932	0.0349	3.1004
31 yeast-20an-nn	2.3236	2.1041	0.1416	0.1646	0.0829	87.7835	0.0174	0.6620	0.0105	0.0058	0.7540
32 vowel_context	2.5268	1.9564	0.0528	0.0679	0.0656	20.2383	0.0173	0.5825	0.0147	0.0058	0.6775
33 Ionosphere	0.1164	0.9094	0.0057	0.0059	0.0230	0.9816	0.0169	0.0293	0.0405	0.0050	0.3344
34 german_credit	0.1593	1.9635	0.0520	0.1618	0.0589	20.6455	0.0169	0.0931	0.0533	0.0071	0.6932
35 splice1	0.8042	2.5517	0.6661	1.6345	0.1397	962.0511	0.0179	0.3171	0.2853	0.0252	2.1243
36 Mush	0.6141	5.2488	0.0172	0.1621	0.1579	8.0781	0.0197	0.2339	0.2244	0.0147	0.5309

Table 16
Testing time (in seconds) efficiency results for different classification algorithms.

Time	Classifiers										
Data sets	GBDT	RF	ELM	SVM	C4.5	SRC	KNN	LR	AB	NB	DL
1 Yeast	0.0014	0.5573	0.0096	0.0167	0.0353	0.1607	0.1205	0.0346	0.0257	0.1204	0.0073
2 Wine	0.0003	0.4468	0.0003	0.0002	0.0056	0.0049	0.0048	0.0064	0.0078	0.0072	0.0004
3 thyroid	0.0004	0.4552	0.0014	0.0014	0.0140	0.0137	0.0055	0.0297	0.0141	0.0167	0.0010
4 shuttle	0.0007	0.4488	0.0164	0.0042	0.0344	0.0542	0.0136	0.0379	0.0356	0.0368	0.0012
5 Seeds	0.0003	0.4507	0.0003	0.0001	0.0060	0.0023	0.0040	0.0065	0.0060	0.0062	0.0007
6 pima	0.0003	0.4421	0.0016	0.0020	0.0141	0.0079	0.0049	0.0152	0.0146	0.0291	0.0010
7 penbased	0.0020	0.4763	0.0040	0.0080	0.0202	0.0475	0.0069	0.0214	0.0190	0.0319	0.0011
8 page-blocks	0.0003	0.4256	0.0009	0.0003	0.0115	0.0059	0.0045	0.0120	0.0121	0.0121	0.0010
9 new_thyroid	0.0003	0.4299	0.0004	0.0001	0.0062	0.0020	0.0039	0.0065	0.0063	0.0061	0.0007
10 image_segmentation	0.0024	0.4719	0.0193	0.0116	0.0428	0.1689	0.0190	0.0426	0.0373	0.0458	0.0028
11 IIF_Intensity	0.0004	0.4719	0.0019	0.0083	0.0126	0.0111	0.0068	0.0148	0.0120	0.0152	0.0012
12 glass_reduction	0.0004	0.5964	0.0003	0.0003	0.0061	0.0028	0.0059	0.0064	0.0059	0.0079	0.0006
13 FER	0.0021	0.4497	0.0031	0.0202	0.0166	0.0302	0.0095	0.0210	0.0193	0.0212	0.0013
14 Ecoli_reduction	0.0004	0.4846	0.0008	0.0004	0.0079	0.0049	0.0043	0.0085	0.0078	0.0082	0.0008
15 dermatology	0.0005	0.4200	0.0009	0.0013	0.0085	0.0076	0.0046	0.0103	0.0085	0.0168	0.0007
16 contraceptive	0.0006	0.4304	0.0110	0.0093	0.0255	0.0341	0.0085	0.0258	0.0248	0.0251	0.0049
17 Cardiotocography	0.0040	0.5068	0.0178	0.0350	0.0355	0.1444	0.0187	0.0374	0.0394	0.0492	0.0053
18 car	0.0028	0.4638	0.0124	0.0036	0.0279	0.0680	0.0102	0.0307	0.0288	0.0289	0.0017
19 balance	0.0004	0.4342	0.0011	0.0004	0.0126	0.0110	0.0044	0.0144	0.0129	0.0122	0.0010
20 autos	0.0003	0.4226	0.0003	0.0004	0.0053	0.0024	0.0040	0.0060	0.0053	0.0058	0.0003
21 abalone	0.0072	0.8125	0.0594	0.0981	0.0668	0.6031	0.0367	0.0733	0.0651	0.0789	0.0079
22 phoneme	0.0017	0.5028	0.0984	0.0272	0.0814	0.3340	0.0554	0.1072	0.0841	0.0819	0.0021
23 winequality-white	0.0106	0.7279	0.0812	0.1493	0.0749	0.4451	0.0546	0.0955	0.0768	0.0876	0.0037
24 winequality-red	0.0010	0.4699	0.0114	0.0225	0.0279	0.0530	0.0094	0.0277	0.0272	0.0281	0.0014
25 twonorm-5an-nn	0.0016	0.5565	0.1779	0.0733	0.1113	0.6202	0.1762	0.1419	0.1068	0.1180	0.0042
26 segment-5an-nn	0.0019	0.5064	0.0204	0.0240	0.0368	0.2009	0.0182	0.0429	0.0359	0.0448	0.0017
27 page-blocks-5an-nn	0.0052	0.6109	0.0966	0.0312	0.0815	0.3743	0.0880	0.0974	0.0827	0.0866	0.0021
28 spambase-10an-nn	0.0017	0.5130	0.0795	0.1915	0.0697	0.6797	0.1609	0.1117	0.0733	0.0782	0.0024
29 segment-10an-nn	0.0038	0.4715	0.0200	0.0188	0.0377	0.1422	0.0188	0.0398	0.0366	0.0456	0.0016
30 spambase-15an-nn	0.0011	0.5254	0.0753	0.2414	0.0699	0.6753	0.1522	0.0748	0.0919	0.0776	0.0019

(*continued on next page*)

Table 16 (*continued*)

Time	Classifiers										
Data sets	GBDT	RF	ELM	SVM	C4.5	SRC	KNN	LR	AB	NB	DL
31 yeast-20an-nn	0.0037	0.4841	0.0098	0.0126	0.0255	0.1024	0.0082	0.0261	0.0300	0.0267	0.0026
32 vowel_context	0.0023	0.4632	0.0036	0.0075	0.0177	0.0686	0.0063	0.0190	0.0175	0.0204	0.0022
33 Ionosphere	0.0002	0.4270	0.0008	0.0007	0.0087	0.0050	0.0044	0.0095	0.0089	0.0086	0.0007
34 german_credit	0.0003	0.4485	0.0036	0.0091	0.0219	0.0168	0.0093	0.0272	0.0180	0.0192	0.0014
35 splice1	0.0009	0.5083	0.0429	0.0545	0.0486	0.2120	0.0937	0.0560	0.0510	0.0565	0.0023
36 Mush	0.0002	0.4321	0.0027	0.0114	0.0108	0.0200	0.0120	0.0182	0.0122	0.0133	0.0006

Table 17
Total running time (in seconds) efficiency results for different classification algorithms.

Time	Classifiers										
Data sets	GBDT	RF	ELM	SVM	C4.5	SRC	KNN	LR	AB	NB	DL
1 Yeast	93.8526	25.9565	29.3568	0.1838	0.2590	112.8336	0.9489	0.8381	0.0373	0.1963	0.8574
2 Wine	8.5725	19.7928	0.5683	0.0030	0.0115	0.5868	0.0233	0.0294	0.0276	0.0130	0.1202
3 thyroid	11.7767	32.6109	6.6639	0.0157	0.0225	11.4799	0.0226	0.1729	0.0466	0.0230	0.3717
4 shuttle	19.0177	18.8908	66.7429	0.0462	0.0564	332.9119	0.0309	0.2462	0.1050	0.0433	1.0205
5 Seeds	9.1109	10.8117	0.4771	0.0013	0.0114	0.6265	0.0216	0.0369	0.0173	0.0099	0.1014
6 pima	7.1861	16.5359	7.5792	0.1331	0.0278	11.8796	0.0217	0.0302	0.0374	0.0375	0.4496
7 penbased	70.5040	51.5714	20.7990	0.0596	0.0606	26.0831	0.0238	0.5176	0.0299	0.0413	0.5672
8 page-blocks	11.7569	16.8439	4.1260	0.0043	0.0197	5.5562	0.0216	0.0655	0.0382	0.0165	0.2708
9 new_thyroid	8.2787	8.1473	0.5775	0.0009	0.0111	0.5056	0.0210	0.0196	0.0152	0.0095	0.1198
10 image_segmentation	103.8036	88.5705	88.5963	0.0990	0.1418	338.4947	0.0375	5.9324	0.0579	0.0569	1.2538
11 IIF_Intensity	77.3140	252.9774	4.7348	0.1340	0.1148	5.4606	0.0235	31.1548	0.0742	0.0237	0.5037
12 glass_reduction	15.6069	15.3914	0.5244	0.0032	0.0141	0.3533	0.0344	0.0380	0.0113	0.0134	0.1872
13 FER	171.9570	377.9565	9.5469	0.3113	0.2748	29.5793	0.0273	0.8440	0.0542	0.0321	0.5841
14 Ecoli_reduction	16.4886	10.9204	1.2079	0.0042	0.0141	1.0962	0.0216	0.0356	0.0141	0.0121	0.1601
15 dermatology	20.7021	48.2502	1.5432	0.0100	0.0176	1.2458	0.0219	0.1356	0.0149	0.0235	0.2854
16 contraceptive	28.6780	22.9369	29.0171	0.2027	0.0930	92.7869	0.0257	0.0767	0.0339	0.0303	0.7973
17 Cardiotocography	190.7999	109.6142	64.7671	0.2850	0.1668	307.2765	0.0366	11.4849	0.0536	0.0605	1.1397
18 car	34.7088	12.7300	38.9322	0.0581	0.0435	83.3239	0.0278	0.1467	0.0664	0.0336	0.9507
19 balance	15.4721	7.4807	4.1102	0.0069	0.0217	4.8749	0.0217	0.0397	0.0212	0.0161	0.3088
20 autos	15.3653	40.8648	0.3305	0.0070	0.0131	0.4600	0.0213	0.1926	0.0114	0.0097	0.0969
21 abalone	676.5194	103.6932	214.5246	1.2787	0.5821	2146.7031	0.0558	4.0784	0.0850	0.0893	2.2263
22 phoneme	26.6357	34.5028	388.4505	0.3316	0.1866	3530.1340	0.0740	0.1920	0.2167	0.0914	2.7531
23 winequality-white	189.5612	124.8552	321.3620	4.1406	0.5480	3145.8451	0.0729	1.4489	0.1003	0.1047	2.7683
24 winequality-red	66.4235	43.7430	34.5401	0.3369	0.1145	129.8756	0.0269	0.2691	0.0466	0.0348	0.9132
25 twonorm-5an-nn	165.0550	425.6094	792.3773	1.3859	0.7191	5978.0202	0.1949	0.5389	0.7581	0.1494	3.9290
26 segment-5an-nn	133.6708	98.9874	99.9592	0.2176	0.1670	338.2321	0.0363	1.2984	0.0628	0.0555	1.2762
27 page-blocks-5an-nn	153.7433	68.0199	542.6178	0.8081	0.3186	3852.9743	0.1131	0.8550	0.1665	0.0991	2.6766
28 spambase-10an-nn	110.5440	727.0735	391.6938	3.6514	0.4819	2394.7797	0.1817	0.4625	0.5739	0.1167	3.1975
29 segment-10an-nn	149.3835	109.1441	103.3961	0.1807	0.1899	324.9984	0.0368	0.9361	0.0666	0.0588	1.1650
30 spambase-15an-nn	120.5493	739.1174	380.8176	5.3296	0.5549	2429.4753	0.1721	0.3020	0.6851	0.1125	3.1023
31 yeast-20an-nn	122.1776	28.8895	30.9767	0.1772	0.1084	87.8859	0.0256	0.6881	0.0405	0.0325	0.7566
32 vowel_context	112.3140	43.6930	11.1296	0.0754	0.0833	20.3069	0.0236	0.6015	0.0322	0.0262	0.6797
33 Ionosphere	7.2201	60.1375	1.2047	0.0066	0.0317	0.9866	0.0213	0.0388	0.0494	0.0136	0.3351
34 german_credit	13.2927	89.6951	11.9188	0.1709	0.0808	20.6623	0.0262	0.1203	0.0713	0.0263	0.6946
35 splice1	48.2881	271.2588	133.5522	1.6890	0.1883	962.2631	0.1116	0.3731	0.3363	0.0817	2.1266
36 Mush	35.3123	938.1919	3.8523	0.1735	0.1687	8.0981	0.0317	0.2521	0.2366	0.0280	0.5315

In this work, by considering a broad range of data sets, we give a domain-agnostic evaluation report for established and more recently introduced classifiers, which could be useful as a guideline for researchers or practitioners that need to select a classifier for a specific need or would consider constructing ensembles using any of the examined classification algorithms as a base classifier. Some of the classifiers included in this report, such as DL, ELM, and SRC, have not been thoroughly examined in the literature. We report both on individual and group-wise prediction performance and make statistical comparisons between the classification algorithms. In order to offer an insight of the influence of the data properties (number of classes and features), we make pair-wise comparisons between the top-performing algorithms, splitting the list of data sets accordingly. The included time efficiency report could also reveal further interesting results.

An important conclusion of this work is that GBDT, while not as widely utilised in classification systems as SVM and RF, performs well consistently across different data sets, even with a non-exhaustive hyper-parameter tuning, and even in some cases where SVM or RF do not perform well, which points that it can be included in heterogeneous ensembles. In our experimental findings, GBDT is the fastest algorithm in testing/prediction, which is a desirable characteristic when considering practical applications for real-time and dynamic systems. In general, our empirical assessment is in accordance with what has been reported in the recent literature (Ayaki, Yanagimoto, & Yoshioka, 2016; Xia, Liu, Li, & Liu, 2017).

Nevertheless, our selection of UCI and KEEL data sets has not been done in a systematic manner, so the distributions of the data sets properties (see Figs. 1–3) should be taken into account when drawing conclusions and making decisions, since the robustness of the utilised performance metrics has not been examined. We use cross-validation to prevent the overfitting problem but more metrics should be considered for generalisations to arbitrary problems if the goal is to evaluate general-purpose classifiers.

In the future work, we will further investigate the performance of the 11 classifiers in specific application domains and with different feature selection methods.

Acknowledgments

This work is partially funded by the National Science Foundation of China (NSFC) under Grant no. 41401466 and 61300215, as well as Henan Science and Technology Project under Grant no. 132102210188. It is also supported by Henan University under Grant no. xxjc20140005 and 2013YBZR014. The authors acknowledge the help of Ms. Jingjun Bi on reorganising the experimental results.

References

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., & Herrera, F. (2010). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing, 17*(2–3), 255–287.

Alcalá-Fdez, J., Sánchez, L., García, S., del Jesus, M., Ventura, S., Garrell, J., … Herrera, F. (2009). Keel: A software tool to assess evolutionary algorithms for data mining problems. *Soft Computing, 13*(3), 307–318.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician, 46*(3), 175–185.

Arauzo-Azofra, A., Aznarte, J. L., & Benitez, J. M. (2011). Empirical study of feature selection methods based on individual feature evaluation for classification problems. *Expert Systems with Applications, 38*(7), 8170–8177.

Ayaki, T., Yanagimoto, H., & Yoshioka, M. (2016). Recommendation from access logs with ensemble learning. *Artificial Life and Robotics*, 1–5.

Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., & Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society, 54*(6), 627–635.

Ballings, M., den Poel, D. V., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications, 42*(20), 7046–7056.

Bengio, Y. (2009). Learning deep architectures for ai. *Foundation and Trends in Machine Learning, 2*(1), 1–127.

Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition, 30*(7), 1145–1159.

Breiman, L. (2001). Random forests. *Mach. Learn., 45*, 5–32.

Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications, 39*(3), 3446–3453.

Carletta, J. (1996). Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics, 22*(2), 249–254.

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.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology, 2*, 27:1–27:27.

Chapelle, O., & Chang, Y. (2011). Yahoo! learning to rank challenge overview. . In O. Chapelle, Y. Chang, & T.-Y. Liu (Eds.), *Yahoo! learning to rank challenge*. In *JMLR Proceedings: 14* (pp. 1–24). JMLR.org.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. In *KDD '16* (pp. 785–794).

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning, 20*, 273–297.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 215–242.

Daniel, W. W. (2000). *Applied nonparametric statistics* (2nd ed.). Cengage Learning.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis, 1*.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research, 7*, 1–30.

Deng, L., & Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing, 7*(3–4), 197–387.

Duda, R. O., Stork, D. G., & Hart, P. E. (2000). *Pattern classification*. Wiley.

Dunn, O. J. (1964). Multiple comparisons using rank sums. *Technometrics, 6*(3), 241–252.

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research, 15*(1), 3133–3181.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *International conference on machine learning* (pp. 148–156).

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis, 38*(4), 367–378.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*, 675–701.

Garcia, S., & Herrera, F. (2009). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research, 9*, 2677–2694.

Giudici, P. (2005). *Applied data mining: Statistical methods for business and industry*. John Wiley & Sons.

Goodfellow, I. J., Warde-Farley, D., Lamblin, P., Dumoulin, V., Mirza, M., Pascanu, R., Bergstra, J., Bastien, F., & Bengio, Y. (2013). Pylearn2: A machine learning research library. arXiv:1308.4214.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182.

Hand, D. J. (2009). Measuring classifier performance: A coherent alternative to the area under the roc curve. *Machine Learning, 77*(1), 103–123.

Hand, D. J., & Till, R. J. (2001). A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning, 45*(2), 171–186.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*(9), 1263–1284.

Hernández-Orallo, J., Flach, P., & Ferri, C. (2012). A unified view of performance metrics: Translating threshold choice into expected classification loss. *Journal of Machine Learning Research, 13*(Oct), 2813–2869.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 1527–1554.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2003). A practical guide to support vector classification. *Technical Report*. Department of Computer Science, National Taiwan University.

Hu, H., Li, J., Plank, A. W., Wang, H., & Daggard, G. (2006). A comparative study of classification methods for microarray data analysis. In P. Christen, P. J. Kennedy, J. Li, S. J. Simoff, & G. J. Williams (Eds.), *AusDM: 61* (pp. 33–37). Australian Computer Society.

Huang, G.-B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 42*(2), 513–529.

Jones, S., Johnstone, D., & Wilson, R. (2015). An empirical evaluation of the performance of binary classifiers in the prediction of credit ratings changes. *Journal of Banking & Finance, 56*(C), 72–85.

Keerthi, S. S., & Lin, C.-J. (2003). Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation, 15*(7), 1667–1689.

Khoshgoftaar, T. M., Golawala, M., & Hulse, J. V. (2007). An empirical study of learning from imbalanced data using random forest.. In *ICTAI (2)* (pp. 310–317). IEEE Computer Society.

King, R. D., Feng, C., & Sutherland, A. (1995). Statlog: Comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence, 9*, 289–333.

Kononenko, I., & Bratko, I. (1991). Information-based evaluation criterion for classifier's performance. *Machine Learning, 6*(1), 67–80.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*.

Kuncheva, L. I. (2002). Switching between selection and fusion in combining classifiers: An experiment. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 32*(2), 146–156.

Kuncheva, L. I., & Rodríguez, J. J. (2010). Classifier ensembles for fmri data analysis: An experiment. *Magnetic Resonance Imaging, 28*(4), 583–593.

Landgrebe, T. C. W., & Duin, R. P. W. (2007). Approximating the multiclass roc by pairwise analysis. *Pattern Recognition Letters, 28*(13), 1747–1758.

Le Cessie, S., & Van Houwelingen, J. C. (1992). Ridge estimators in logistic regression. *Applied Statistics, 41*(1), 191–201.

Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136.

Li, T., Zhang, C., & Ogihara, M. (2004). A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics, 20*, 2429–2437.

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*(3), 203–228.

Lin, H.-T., & Lin, C.-J. (2003). A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. *Neural Computation*, 1–32. submitted to

Ling, C. X., Huang, J., & Zhang, H. (2003). Auc: a better measure than accuracy in comparing learning algorithms. In *Conference of the canadian society for computational studies of intelligence* (pp. 329–341).

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering.. *IEEE Transactions on Knowledge and Data Engineering, 17*, 491–502.

Liu, Y. (2004). A comparative study on feature selection methods for drug discovery. *Journal of Chemical Information and Modeling, 44*(5), 1823–1828.

Liu, Y., Loh, H. T., & Tor, S. B. (2005). Comparison of extreme learning machine with support vector machine for text classification. In M. Ali, & F. Esposito (Eds.), *IEA/AIEaie*. In *Lecture Notes in Computer Science: 3533* (pp. 390–399). Springer.

Lorena, A. C., Jacintho, L. F. O., de Siqueira, M. F., Giovanni, R. D., Lohmann, L. G., de Carvalho, A. C. P. L. F., & Yamamoto, M. (2011). Comparing machine learning classifiers in potential distribution modelling. *Expert Systems with Applications, 38*(5), 5268–5275.

Macia, N., & Bernado-Mansilla, E. (2014). Towards uci+: A mindful repository design.. *Information Science, 261*, 237–262.

Nanni, L., Brahnam, S., Ghidoni, S., & Lumini, A. (2015a). Toward a general-purpose heterogeneous ensemble for pattern classification. *Computational Intelligence and Neuroscience, 2015*, 85.

Nanni, L., Brahnam, S., & Lumini, A. (2010). High performance set of pseaac and sequence based descriptors for protein classification. *Journal of Theoretical Biology, 266*(1), 1–10.

Nanni, L., Brahnam, S., & Lumini, A. (2012). Matrix representation in pattern classification. *Expert Systems with Applications, 39*(3), 3031–3036.
Nanni, L., Fantozzi, C., & Lazzarini, N. (2015b). Coupling different methods for overcoming the class imbalance problem. *Neurocomputing, 158*(C), 48–61.
Ngai, E., Xiu, L., & Chau, D. (2009). Application of data mining techniques in customer relationship management: A literature review and classification. *Expert Systems with Applications, 36*(2, Part 2), 2592–2602.
Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing, 26*(1), 217–222.
Palm, R. B. (2012). *Prediction as a candidate for learning deep hierarchical models of data*: 5. Technical University of Denmark.
Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..
Rijsbergen, C. J. V. (1979). *Information retrieval* (2nd). Butterworth-Heinemann.
Saeys, Y., Abeel, T., & de Peer, Y. V. (2008). Robust feature selection using ensemble feature selection techniques. In *Machine learning and knowledge discovery in databases, european conference, ECML/PKDD 2008* (pp. 313–325).
Saeys, Y., Inza, I. n., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics, 23*(19), 2507–2517.
Tukey, J. W. (1977). *Exploratory data analysis*. Addison-Wesley.
Tuszynski, J. (2008). Catools: Tools: Moving window statistics, gif, base64, roc auc, etc. R package version, 1.
Van Rijn, J. N., Bischl, B., Torgo, L., Gao, B., Umaashankar, V., Fischer, S., … Vanschoren, J. (2013). Openml: A collaborative science platform. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 645–649). Springer.
Vanschoren, J., Blockeel, H., Pfahringer, B., & Holmes, G. (2012). Experiment databases - a new way to share, organize and learn from experiments.. *Machine Learning, 87*(2), 127–158.
Wei, B., Yang, M., Shen, Y., Rana, R. K., Chou, C. T., & Hu, W. (2013). Real-time classification via sparse representation in acoustic sensor networks. In *The 11th ACM conference on embedded network sensor systems, SenSys '13, Roma, Italy* (pp. 1–14).
Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin, 1*(6), 80–83.
Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computing, 8*(7), 1341–1390.
Wright, J., Yang, A. Y., Ganesh, A., Sastry, S. S., & Ma, Y. (2009). Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(2), 210–227.
Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., … Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems, 14*(1), 1–37.
Xia, Y., Liu, C., Li, Y., & Liu, N. (2017). A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring. *Expert Systems with Applications, 78*, 225–241.
Ye, J., Chow, J.-H., Chen, J., & Zheng, Z. (2009). Stochastic gradient boosted distributed decision trees. . In D. W.-L. Cheung, I.-Y. Song, W. W. Chu, X. Hu, & J. J. Lin (Eds.), *CIKM* (pp. 2061–2064). ACM.
Yule, G. U. (1900). On the association of attributes in statistics: With illustrations from the material of the childhood society, &c. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character, 194*, 257–319.
Zheng, Z. (1993). A benchmark for classifier learning. *Technical Report*. University of Sydney.
Zhu, X., Wu, X., & Yang, Y. (2004). Error detection and impact-sensitive instance ranking in noisy datasets. In *AAAI* (pp. 378–384).