Available online at www.sciencedirect.com
SCIENCE DIRECT®
Applied Soft Computing 6 (2006) 119–138
Applied Soft Computing
www.elsevier.com/locate/asoc

On learning algorithm selection for classification

Shawkat Ali *, Kate A. Smith

Faculty of Information Technology, Monash University, P.O. Box 63B, Vic. 3800, Australia

Received 24 February 2004; received in revised form 9 December 2004; accepted 13 December 2004

Abstract

This paper introduces a new method for learning algorithm evaluation and selection, with empirical results based on classification. The empirical study has been conducted among 8 algorithms/classifiers with 100 different classification problems. We evaluate the algorithms' performance in terms of a variety of accuracy and complexity measures. Consistent with the No Free Lunch theorem, we do not expect to identify the single algorithm that performs best on all datasets. Rather, we aim to determine the characteristics of datasets that lend themselves to superior modelling by certain learning algorithms. Our empirical results are used to generate rules, using the rule-based learning algorithm C5.0, to describe which types of algorithms are suited to solving which types of classification problems. Most of the rules are generated with a high confidence rating.

Keywords: Algorithm selection; Classification; No Free Lunch theorem

1. Introduction

Classification, one of the most popular and significant machine learning areas, is particularly important when a data repository contains samples that can be used as the basis for future decision making: for example, medical diagnosis or credit fraud detection. Machine learning researchers have already proposed many different types of classification algorithms, including nearest-neighbour methods, decision tree induction, error backpropagation, reinforcement learning, lazy learning, rule-based learning, and relatively new addition is statistical learning. From amongst this vast and ever increasing array of classification algorithms, it becomes important to ask the question 'which algorithm should be the first choice for my present classification problem?' Our present research seeks to find an appropriate answer to the above question by developing a new approach to learning algorithm selection. Of course, it is important to keep in mind Wolpert and Macready's well-known No Free Lunch (NFL) theorem [1]:

> If algorithm A outperforms algorithm B on some cost functions, then loosely speaking there must exist exactly as many other functions where B outperforms A.
>
> NFL theorem.

Many studies propose new classification algorithms, and attempt to produce empirical evidence of

* Corresponding author. Tel.: +61 3 9905 5800; fax: +61 3 9905 9422.
E-mail addresses: Shawkat.Ali@infotech.monash.edu.au (S. Ali), Kate.Smith@infotech.monash.edu.au (K.A. Smith).

1568-4946/$ – see front matter
doi:10.1016/j.asoc.2004.12.002

the superiority of the algorithm on a selection of datasets. The NFL theorem suggests, however, that a more useful strategy is to gain an understanding of the dataset characteristics that enable different learning algorithms to perform well, and to use this knowledge to assist learning algorithm selection based on the characteristics of the dataset.

Much of the comparative research on algorithms in the machine learning community is between decision trees and neural networks [2] and places the emphasis on the percentage of correct classifications. The STATLOG project [3] is one of the largest comparisons introduced among algorithms on a large number of datasets with statistical analysis. An algorithm's performance is measured on both the percentage of correct classifications and computational complexity. The project finds that no algorithm is uniformly most accurate over the dataset studied, consistent with the basic idea of the NFL theorem. Another large comparison recently undertaken [2] has considered 22 decision tree, 9 statistical and 2 neural network algorithms compared on 32 datasets based on the percentage of correct classification, training time, and (in the case of trees) numbers of leaves. Classification accuracy is measured by mean error rate and mean rank of error rate. The main difference in their study is that they considered the training time, more decision tree models as well as statistical algorithms. They also studied the effect of adding independent noise attributes on the classification accuracy, and examined the scalability of some of the more promising algorithms as the sample size was increased. Smith et al. [4] has introduced a methodology for choosing an algorithm for a new problem by clustering with the self-organising map (SOM). Other studies that are smaller in scale include [5–8].

The accuracy or error rate measure is the primary and the most popular way to evaluate the performance of data mining algorithms. This type of measure always assumes that the class distribution is unchangeable, that is the error costs—the cost of the majority and minority classes—have similar significance [9]. For instance, suppose we have a dataset with 1000 samples, the majority class containing 990 samples and the minority class containing 10 samples. Now if a classifier classifies them all as a majority class, the accuracy will be 99%, even though the classifier missed all minority samples due to the highly unbalanced class distribution. Therefore, this method is criticised as an impractical technique [9]. Another popular way to evaluate a classifier is the cross-validation approach. The hold-out, leave-one-out and rotation methods are different approaches to cross-validation. The main disadvantage of the cross-validation method is that all the samples are not used to construct the model when the samples are relatively small. Moreover, the hold-out method and leave-one-out method suffer from either large bias or variance [10]. To overcome the limitation of error/accuracy estimation, we propose a new classifier evaluation method called relative weighted performance measure.

Our research compares mainly decision tree algorithm C4.5, neural network trained with the backpropagation (BP) algorithm, and relatively new classifier Support Vector Machine (SVM) with some other statistical and rule-based classifiers. We organise our research in three main steps: first we compare the algorithms across a number of different measures of accuracy and computational time providing a comprehensive empirical evaluation of the performance of eight classifiers on 100 classification datasets. We then characterise the datasets using a variety of simple, statistical and information theoretical measures. Finally, the empirical results are combined with the dataset characteristic measures to generate rules describing which algorithm is best suited to which types of problems.

This study extends the previous work of the STATLOG project and Lim et al. in the following ways:

- We include one of the rapidly popular classifier based on statistical learning theory, SVM, in our study.
- We consider 100 different datasets to measure the classifier performance by 10-fold cross-validation and hold-out estimation methods.
- We introduce a new methodology to compare the classifier performance across a variety of measures rather than the common focus on percentage of correct classifications.
- We consider statistical significance test to compare the classifier performance.
- Finally, we generate rules to determine which classifier is suitable for which types of problems.

We also evaluate the rules by support and confidence measures.

In this study, we have examined 100 classification problems from the UCI Repository [11] and Knowledge Discovery Central [12] (see Appendix A), and evaluated 8 popular data mining classifiers based on a combined performance measure of average accuracy and computation time. The average accuracy is the combination of the true positive rate (TPR), the true negative rate (TNR), the percentage of correct classification (10-fold cross-validation and hold-out) and the weighted F-measure. We consider 10-fold cross-validations for those datasets with the number of samples fewer than 1000. On the other hand, for those dataset with more than 1000 samples, we consider the hold-out method, 70% for training and the rest for testing, as suggested by Henery [13]. The computational complexity considers both the model train time as well as the test set evaluation time, rather than placing emphasis on only one of these, since some of the algorithms need more time to classify the test set than training the model. The machine configuration was Pentium IV, CPU 2.66 GHz and 1 GB RAM. Some of multi-class datasets were converted into binary class. We considered all the algorithms from WEKA release 3.1.8 with default parameter settings. WEKA [14] is a Java-based machine learning tool. The algorithms performance significance test used the well-known statistical t-test. Dataset characteristics of each problem are measured following Smith et al. [4,15]. We consider simple, statistical and information theoretic measures to identify the dataset characteristics. Some of the statistical formulation is available in Matlab Statistics Toolbox [16]. By using the most co-related attributes, we generate rules to identify which algorithm is suitable for which type of problem. Finally, we examined the rules by the support and confidence measure.

The rest of the paper we organise as follows. Section 2 describes briefly the eight learning algorithms. Section 3 introduces our new methodology for classifier evaluation. The experimental results and the final observation from the comparative studies are presented in Section 4. Section 5 explains the rule generation methodology, including the measures used to characterise the datasets. The generated rules for classifier selection are presented in Section 6, together with their confidence and support. Finally, we draw conclusions from our research in Section 7.

2. Learning algorithms

This section provides a short description of all the algorithms we consider in our experimental design. All of the algorithms belong to the category of supervised learning methods, but we can further categorise them into neural, rule-based and statistical learning algorithms as described in the following sections.

2.1. Neural-based learner

In the mid-1960s, Nilsson introduced artificial intelligence for pattern recognition based on neural like threshold units called neural networks (NNs). NNs became an approach after the development of some new algorithms, such as multilayer perceptrons (MLP), radial basis function networks, SOM and BP. The MLP architecture consists of three layers of neurons, namely the input, hidden and output layers, all connected by feed forward weights. After receiving an input pattern, the NN passes the signal through the network to predict the output in the output layer. The NN then compares the predicted target value with the actual target and estimates the error to modify the weights. The scalar error function of the weights is minimised by repeating the learning procedure until the network produces the correct response to each input [17,18,37]. WEKA uses the BP algorithm to train the model. BP minimises the error function using a gradient descent method. The main disadvantage of the BP algorithm is that it is slower than some other popular machine learning techniques, and tends to become trapped in local minima of the error function [19].

2.2. Rule-based learner

Rule-based learning, especially decision trees (also called classification trees or hierarchical classifiers), is a divide-and-conquer approach or a top-down induction method that have been studied with more interest in the machine learning community. C4.5 is the advanced version decision tree algorithm of ID3 [17].

ID3 means the third series of 'interactive dichotomizer' procedures. It can classify nominal datasets only. For real value attributes, it is first binned into interval to form unordered nominal values. It does not consider any standard pruning procedure. By minimising the ID3 limitation, Quinlan [17] introduced C4.5 algorithm to solve classification problems. Like NN, C4.5 works in three main steps. First, the root node at the top node of the tree considers all samples and passes through the samples information in the second node called 'branch node'. The branch node generates rules for a group of samples based on entropy measure. In this stage, C4.5 constructs a very big tree by considering all attribute values and finalises the decision rule by pruning. It uses a heuristic approach for pruning based on statistical significance of splits. After fixing the best rule, the branch nodes send the final target value in the last node called the 'leaf node' [17,18]. OneR is a very simple, faster and one-level decision tree algorithm. It selects one-by-one attributes from a dataset and generates a different set of rules based on error rate from the training set. Finally, it chooses the attribute that offers rules with minimum error and constructs the final decision tree [20]. PART is a partial decision tree algorithm, which is the developed version of C4.5 and RIPPER algorithms. The main speciality of the PART algorithm is that it does not need to perform global optimisation like C4.5 and RIPPER to produce the appropriate rules [21]. However, decision trees are sometime more problematic due to the larger size of the tree which could be oversized and might perform badly for classification problems [22].

2.3. Statistical learner

Recently, statistical learning theory has received more attention from the pattern recognition community after the introduction of SVM by Vapnik and his group in the mid-1990s. SVM is the advanced version of the Generalized Portrait algorithm, which was developed in Russia in the late 1960s [23]. SVM works in the similar way to NN and C4.5. We can assign the three working phases for SVM, first one is input phase or transformation phase, then learning phase and final one is decision phase. NN and C4.5 do not perform any significant work in the first phase. But SVM does its most significant job, transformation of the data by using kernel mapping into a high dimensional feature space. The kernel function can be polynomial, Gaussian or many others. The high dimensional space could theoretically be infinite, where linear discrimination is almost possible. SVM starts to learn the data in the high dimensional feature space, in the learning phase, by minimising the magnitude of the weight vector constrained by the separation (optimal hyperplane based) into an unconstrained problem with the help of multiplier parameter, say Lagrange multiplier. In this stage, SVM extracts the *support vectors* only. Based on the support vectors information, SVM produces the final output function in the decision phase. Unlike NN and C4.5, SVM does not consider all samples to construct the final decision function. Moreover, SVM always obtains the unique solution for the decision function unlike iterative approaches or pruning. Another speciality of SVM is that it minimises the structural risk rather than empirical risk considered by most classical learning algorithms [18,19,38]. WEKA considers sequential minimal optimization (SMO) for SVM and polynomial kernel with degree 1 as a default setting [14]. Naive Bayes (NB) is a simple classifier based on the classical statistical theory 'Bayes theorem'. The term ''naive'' is because it calculates the maximum posterior probability, based on the assumption that the attributes on the training samples are independent and there is no hidden or latent attributes influence in the prediction procedures [24]. Kernel density (KD) is a non-parametric linear kernel-based density estimation algorithm. This algorithm does not need any prior assumption, such as normal distribution of the attributes for prediction. The discrimination capability of this algorithm is comparatively faster than some other classifiers [14]. IBK is an instance-based learning approach like the K-nearest-neighbour method. The basic principle of this algorithm is that each unseen instance is always compared with existing ones using a distance metric; most commonly Euclidean distance and the closest existing instance is used to assign the class for the test sample [14]. WEKA's default setting is $K = 1$. Compared to other algorithms, it needs more time to predict the test samples' classes.

The statistical learning algorithm, SVM, has some advantages over the well-established algorithms NN and decision tree. It considers the dot product of the feature vectors to construct the optimal hyperplane

rather than surface, clustering or interpolation as like NN or decision tree. So, there is less probability of losing important information during the modelling [25].

3. Classifier evaluation methodology

3.1. Relative weighted performance measure

We consider the two most common measures, accuracy and computational time (on training and test sets) for classifier evaluation in our method. First, we measure the algorithm performance individually for the majority and minority classes by TPR and TNR. Secondly, we measure the performance by 10-fold cross-validation for small datasets (less than 1000 samples). We consider the hold-out method for those datasets having more than 1000 samples. At the last stage, we try to minimise the effect of the imbalance between minority and majority classes' distributions by using the weighted F-measure method. F-measure considers a weighted distribution for a dataset.

We can explain the TPR and TNR measure by using the contingency Table 1.

The TPR is the ratio between the numbers of majority (positive) class samples which are correctly classified by the algorithm and the total numbers of majority class samples:

$$\mathrm{TPR} = \frac{d}{n-k} \times 100\%$$

The TNR is the ratio between the numbers of minority (negative) class samples which are correctly classified by the algorithm and the total numbers of minority class samples:

$$\mathrm{TNR} = \frac{a}{k} \times 100\%$$

Table 1
Contingency table for a binary class problem

	Model says class 1 (−ve)	Model says class 2 (+ve)	
Supervisor says class 1 (−ve)	a	b	$a+b=k$
Supervisor says class 2 (+ve)	c	d	$c+d=n-k$
	$a+c=r$	$b+d=n-r$	$a+b+c+d=n$

Now we can formulate the hold-out and cross-validation estimation following [26]. Let us consider the unlabelled sample space is X, with corresponding labels Y. The space of labelled samples is $\chi = X \times Y$ and $D = \{x_1, x_2, \ldots, x_n\}$ is a dataset, which consists of n labelled samples, where $x_i = \{v_i \in X, y_i \in Y\}$. The inducer (D, ν) will denote the label assigned to an unlabelled sample ν by the classifier built by the inducer on dataset D, i.e., $(D,\ \nu) = ((D))(\nu)$. The hold-out and cross-validation estimation consider the dataset is independent and identically distributed and equal misclassification costs using a 0/1 loss function.

The hold-out method [26] is also called the test sample estimation method. The most common construction procedure is 70% samples used for the training set and the remaining as the test set. Let us consider a hold-out set D_h be a subset of D of size h, and let D_t be $D \backslash D_h$. Now the hold-out estimation is defined as follows:

$$\mathrm{Acc}_{\mathrm{ho}} = \frac{1}{h} \sum_{\langle v_i, y_i \rangle \in D_h} \delta(\Im(D_t, v_i), y_i),$$

where $\delta(i,\ j) = 1$, if $i = j$ and 0 otherwise.

The cross-validation method [26] estimates the average percentage of correct classification for all folds. For example, in the 10-fold cross-validation method, we first split a dataset by 10-fold. Each fold contains 90% of the samples to construct a model and the remaining 10% is used to evaluate the model performance. Finally, we estimate the accuracy is the overall number of correct classification averaged across all 10-fold. Let us consider, D_i is the test set that includes sample $x_i = \langle v_i, y_i \rangle$ and the cross-validation accuracy estimation is defined as:

$$\mathrm{Acc}_{\mathrm{cv}} = \frac{1}{n} \sum_{\langle v_i, y_i \rangle \in D} \delta(\Im(D \backslash D_{(i)}, v_i), y_i),$$

where n is the number of folds.

We call the combined performance of the hold-out and cross-validation accuracy estimation for all given problems of a given classifier 'percentage of correct classification' in our relative weighted performance measure methodology.

Van Rijsbergen introduced a new measure for classifier evaluation [27,28]. The F-measure is the simplified form of E-measure and can be explained by following the same notation in Table 1:

$$\text{recall}\,(R) = \frac{\text{number of true positives}}{\text{number of false negatives} + \text{number of true positives}} = \frac{d}{c+d},$$

$$\text{precision}\,(P) = \frac{\text{number of true positives}}{\text{number of false positives} + \text{number of true positives}} = \frac{d}{b+d}.$$

Bajic introduced average score measure with the combination of TPR and TNR to choose a prediction software [34]. The E-measure [28] can be formulated by using recall and precision as follows:

$$E_\beta = 1 - \frac{1}{\alpha(1/P) + (1-\alpha)(1/R)} \Rightarrow 1 - \frac{(\beta^2+1)PR}{\beta^2 P + R},$$
$$\text{where} \quad \alpha = \frac{1}{\beta^2+1},$$

where β is the balance parameter for R and P. For example, $\beta = 0.5$ means R is half important as P. We explore the β values ranging between 0 and infinity to control the imbalance of the classes' distribution.

By replacing R with TP in the above equation, we can derive the weighted F-measure as follows:

$$F_\beta = \frac{(\beta^2+1)P\,\text{TP}}{\beta^2(P + \text{TP})}.$$

Now we calculate the ranking performance for a given algorithm based on TPR, TNR, percentage of correct classification and F-measure. The best performing algorithm on each of these measures is assigned the rank of 1 and the worst is 0. Thus, the rank of the jth algorithm on the ith dataset is calculated as:

$$R_{ij} = 1 - \frac{e_{ij} - \max(e_i)}{\min(e_i) - \max(e_i)},$$

where e_{ij}, for example, might be the percentage of correct classification for the jth algorithm on dataset i, and e_i is a vector of accuracy for dataset i. By using this equation, a detailed comparison of algorithm performance can be provided (see, for example, Appendix B, where the measure used is percentage of correct classification).

The computational complexity considers the model training time as well as the test set evaluation time. We do not singularly place importance on either training or test time, because some of the algorithms (for example, IBK), need more time to classify the test set than training the model. The total number of best and worst ranks of all classifiers based on TPR, TNR, percentage of correct classification, weighted F-measure and computational complexity is presented in Appendix C.

Next, we evaluate the performance of all the classifiers (using the total number of best and worst performances) among the 100 problems, and call this the formulated classifier performance ($\Re_i$). The total number of the best and worst ranking for TPR, TNR, percentage of correct classification, weighted F-measure and computational complexity for all the classifiers are evaluated by using the following equation:

$$\Re_i = \frac{1}{\rho}\left(\frac{s_i - f_i}{n}\right) + \frac{1}{\rho},$$

where $\rho = 2$ is the weight shifting parameter, s_i is the total number of success (best) cases for the ith classifier, f_i is the total number of failure (worst) cases for the same classifier, and n is the total number of datasets.

Finally, we measure the relative weighted performance for all the classifiers with two different weights for ranking average accuracy (ranking average performance of TPR, TNR, percentage of correct classification and weighted F-measure) and computational complexity using the following equation:

$$Z = \alpha a_i + \beta t_i,$$

where α and β are the weight parameters for ranking average accuracy against computational complexity. The average accuracy and complexity are denoted by a_i and t_i. We consider the range for α and β between 0 and 2. By changing the value of β, we observe the effect of the relative importance of accuracy and computational complexity to our perception of classifier performance.

4. Experimental results

4.1. Performance analysis

The formulated classifier performance ($\Re_i$) for TPR, TNR, percentage of correct classification,

Table 2
Formulated ranking averaged across test set classification problems based on a variety of measures (where a rank of 1 means best performing algorithm, and 0 means worst performing algorithm)

	Classifier							
	IBK	C4.5	PART	KD	NB	OneR	SVM	NN
TPR	0.595	0.595	0.595	0.61	0.495	0.365	0.565	**0.645**
TNR	0.54	0.6	0.6	0.6	0.44	0.385	0.595	**0.605**
Percentage of correct classification	0.505	0.615	0.55	0.565	0.385	0.325	**0.62**	0.615
F-measure	0.565	**0.625**	**0.625**	0.575	0.48	0.365	0.56	0.62
Average accuracy	0.551	0.609	0.593	0.588	0.45	0.36	0.585	**0.621**

Table 3
Computational performance

Classifier	Execution time
IBK	0.535
C4.5	0.535
PART	0.52
KD	0.51
NB	0.705
OneR	**0.995**
SVM	0.5
NN	0.015

weighted F-measure and computational performance are summarised in Tables 2 and 3.

We observe using the TPR and TNR measures that NN was the best performing algorithm and the second best was KD (as shown in Table 2 by bold face). For the percentage of the correct classification measure, SVM was the first choice in our experiment closely followed by NN and C4.5. C4.5/PART was the first choice for the weighted F-measure and the second was NN. On average accuracy measures, NN performed best. The OneR classifier performed worst based on various accuracy measures. However, OneR was the number one choice if we give importance only on execution time rather than classification accuracy; the second choice was NB (as shown in Table 3). The combined performance was an average combination of average accuracy and computational time for all the algorithms is presented in Fig. 1. OneR performed best, IBK, C4.5, PART, KD and SVM performed very close to each other based on the combined performance measure. So, we can choose any one as a second best algorithm among them. However, NN performed worst in our experiment based on combined performance measure as shown in Fig. 1.

The relative weighted performance (Z) calculated by considering β as independent but α fixed is shown in Fig. 2. We kept constant the average accuracy weight $\alpha = 1$ but we changed the weight for computation time β from 0.2 to 2. When t_i is less important than a_i, most of the classifiers performed similarly, but OneR was the worst. After equally

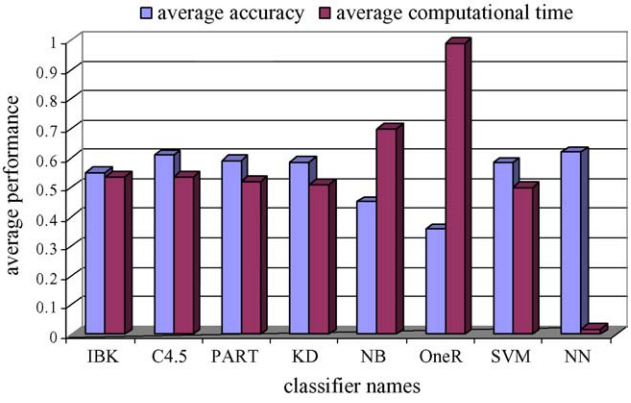

Fig. 1. Combined performance for the classifiers.

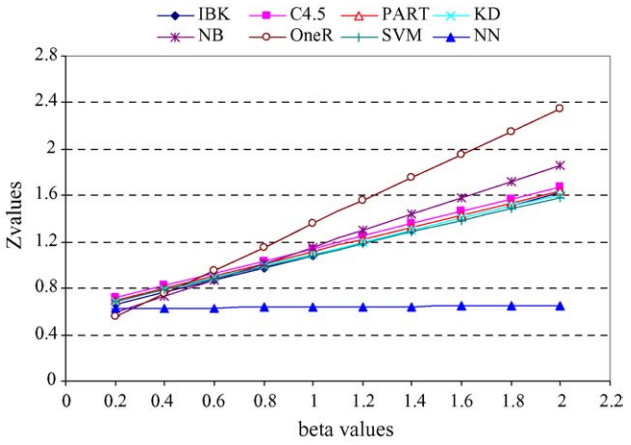

Fig. 2. Relative performance for the classifiers, when α (=1) is fixed.

weighting α and β, OneR became the first choice in our study, and NN performed worst. All the others classifier performances were close to each other. Therefore, we can conclude that the performance of the classifiers depends significantly upon our perception of the relative importance of accuracy versus computation time.

4.2. Significance test

The statistical significance t-test results are summarised in Table 4. The test input was the average performance of the combined performance measures for each algorithm.

Table 4
Results for the t-test

Algorithms	Hypothesis (H)	Significance (δ)	Confidence interval (CI)
SVM vs. IBK	0	0.8039	−0.8417 to 1.0845
SVM vs. C4.5	0	0.7513	−1.0675 to 0.7715
SVM vs. PART	0	0.5776	−1.1688 to 0.6533
SVM vs. KD	0	0.6313	−1.1356 to 0.6905
SVM vs. NB	0	0.8067	−1.0137 to 0.7896
SVM vs. OneR	0	0.9566	−0.8729 to 0.9224
SVM vs. NN	0	0.8960	−0.9522 to 0.8337

The output of $H = 0$ in the above table indicates that we may not reject the null hypothesis and that there is no statistically significant difference between the algorithms when we consider their performance averaged across the datasets, equally weighting accuracy and time considerations. The higher value of the significance level showed that there has no significant differences among SVM, IBK, C4.5, PART, KD, NB, OneR, and NN. The lower and upper CI values for all algorithms showed a closely balanced skewed position. The 95% confidence intervals of all classifiers support acceptance of the null hypothesis. So, based on the average combined performance measure, we may argue statistically that there was no significant performance difference among the classifiers.

But, in some of specific cases there is a remarkable performance difference among the algorithms. For example, the classification percentage of accuracy for ‘new-thyroid’ dataset by C4.5, SVM and NN was 93.59, 79.95 and 96.01. When we compare the performance of the algorithms based only on the average across the datasets, we tend to find that all the algorithms perform similarly on average, with some appearing to be superior depending on the chosen performance measurement and certain datasets, but with no consistently superior algorithm emerging (consistent with the NFL theorem). It is sensible therefore to use the empirical results we have generated to try to understand the conditions under which certain algorithms perform well.

5. Classifier selection methodology

5.1. Rule-based classifier selection

The trial-and-error approach is a very common procedure to select the best classifier. It is a difficult task to find the best classifier by following this procedure. If we are interested in applying these algorithms to a particular problem, then we have to consider which algorithm is more suitable for which problem. The suitability test can be done from rules with the help of dataset characteristics combined with knowledge about how the different algorithms perform on these datasets. Datasets can be characterised by using certain features such as number of attributes, their types, amount of unknown values, statistical measures or other information measures.

5.2. Datasets characteristics measurement

Each dataset can be described by a number of simple, statistical and information theoretical measures [4,15]. We average some statistical measures over all the variables and take these as a global measure of the dataset characteristics.

5.2.1. Simple measures

Some simple measures for dataset characteristics are shown in Table 5. These are the dimensions of each problem, the number of minority and majority samples and the nature of the variables.

Table 5
Simple measures for characterisation of each dataset

Measure	Notation
Number of attributes	a
Number of samples	e
Percentage of minority class	$c_{\min}$
Percentage of majority class	$c_{\max}$
Percentage of binary variables	b_{var}
Percentage of discrete variables	d_{var}
Percentage of continuous variables	c_{var}
Percentage of missing values	m_{per}

5.2.2. Statistical measures

Descriptive statistics can be used to summarise the relevant characteristics of any large dataset. The following section lists some measures provided by the Matlab Statistics Toolbox [16] and other sources [29–32] (Table 6).

Table 6
Statistical measures for characterisation of each dataset

Measure	Notation
Geometric mean	GM
Harmonic mean	HM
Trim mean	TM
Mad	M
Variance	V
Standard deviation	std
Prctile	Pr
Interquartile range	IQR
Maximum and minimum eigenvalue	$E_{\max}$ and $E_{\min}$
Canonical correlation	CC
Index of dispersion	ID
Center of gravity	CG
Kurtosis	k
Skewness	s
Correlation coefficient	r
Z-score	Z_score
Normal cumulative distribution test	p_{ncdf}
Chi-square test	p_{cst}

5.2.3. Information theoretical measures

The quality of the relationships in the data can also be assessed using information theoretical measures. We represent these measures formulation and explanation from [4,15]. A list of information theoretical measures are summarised in Table 7.

Table 7
Information theoretical measures for characterisation of each dataset

Measure	Notation
Mean entropy of variables	$H(X)$
Entropy of classes	$H(C)$
Mean mutual entropy of class and variables	$\bar{H}(C,X)$
Equivalent number of variables	ENV
Noise–signal ratio	NSR

All these data characteristics formulation is available in Appendix D.

6. Rules for classifier selection

Now that the characteristics of each dataset can be quantitatively measured, we can combine this information with the empirical evaluation of classifier

performance presented in Section 4. Based on the 100 classification problems, we can then train a rule-based classifier (C5.0) to learn the relationship between dataset characteristics and algorithm performance.

After identifying the datasets characteristics matrix with class values, we use 90% of the samples to construct the rule-based model for classifier selection. The class labels in the matrix are assigned based on the algorithm performance rank. The labels are assigned to the classifiers IBK to NN from 1 to 8 (same order as shown in Fig. 1). For example, if NN has performed with the highest accuracy for the dataset *A*, then the class membership for problem *A* is 8. We provide the emphasis to assign the best algorithm label based on the combined average performance. The supervised learning algorithm C5.0 is used to generate the rules. It has two parameters, first one is *c* called pruning confidence factor and the second one *m* is called minimum cases. The pruning factor has affect on error estimation and hence the severity of pruning the decision tree. The smaller value of *c* affects more pruning of the generated tree and higher value affects less pruning. The minimum cases *m* offers the degree to which the initial tree can fit the data. Every branch point in the tree, the value of m should be at least two. Higher value of m can offer a form of more pre-pruning decision tree [33]. For detail formulations, see [17]. After suitable tuning, the best rules are selected for the classifier. Finally, we use 10-fold cross-validation to verify the best rule. The significance of the best rule for classifier selection is measured, like for association rules, by support and confidence based on 10-fold cross-validation performance which describes the quality of a rule. The support and confidence can be summarised as follows:

$$\text{support} = \frac{\text{number of dataset that match dataset conditions and best algorithm prediction}}{\text{total number of datasets}},$$

$$\text{confidence} = \frac{\text{number of dataset that match best algorithm prediction}}{\text{number of dataset that match dataset conditions}}.$$

We generated the rules for all classifiers based on the classification performance of 100 datasets. We find the suitable pruning confidence level (*c*) between 60 and 90 and the number of minimum cases (*m*) is 2 to 8. The rules and evaluation performance based on 10-fold cross-validation are presented in Tables 8–15.

Table 8
IBK classifier selection rule evaluation averaged across 10-fold cross-validation test sets

Condition satisfied	IBK technique best	
	Yes	No
Yes	1.5	0.5
No	0	2

Support: 75%; confidence: 75%.

Table 9
C4.5 classifier selection rule evaluation averaged across 10-fold cross-validation test sets

Condition satisfied	C4.5 technique best	
	Yes	No
Yes	2.5	0.5
No	0	3

Support: 83.33%; confidence: 83.33%.

6.1. Rules for IBK

The rules for IBK are generated with $c = 85$ and $m = 4$ as follows:

```
IF (V <= 59.8893 AND r <= 0.51025 AND s > 0.92751 AND ID <= −0.88889 AND NSR <= 22.1654) OR (P_cst > 0.75051 AND ID > −0.066412) THEN we should choose IBK classifier.
```

6.2. Rules for C4.5

The rules for C4.5 are generated with $c = 80$ and $m = 2$ as follows:

```
IF (IQR <= 10.2273 AND Pr > 12.521) OR (M <= 0.72494) THEN we should choose C4.5 classifier.
```

6.3. Rules for PART

The rules for PART are generated with $c = 65$ and $m = 2$ as follows:

Table 10
PART classifier selection rule evaluation averaged across 10-fold cross-validation test sets

Condition satisfied	PART technique best	
	Yes	No
Yes	1.6	0.4
No	0	2

Support: 80%; confidence: 80%.

Table 11
KD classifier selection rule evaluation averaged across 10-fold cross-validation test sets

Condition satisfied	KD technique best	
	Yes	No
Yes	1.5	0.5
No	0	2

Support: 75%; confidence: 75%.

```
IF (GM > 0.1257 AND Z_score <= 35.7787
AND r <= 0.28951 AND s <= 0.92751) THEN
we should choose PART classifier.
```

6.4. Rules for KD

The rules for KD are generated with $c = 75$ and $m = 4$ as follows:

```
IF (NSR <= −73.9496) OR (CC <= 0.229
AND r > 0.1805) OR (NSR > 1362.221) THEN
we should choose KD classifier.
```

Table 12
NB classifier selection rule evaluation averaged across 10-fold cross-validation test sets

Condition satisfied	NB technique best	
	Yes	No
Yes	1.8	0.2
No	0	2

Support: 90%; confidence: 90%.

Table 13
OneR classifier selection rule evaluation averaged across 10-fold cross-validation test sets

Condition satisfied	OneR technique best	
	Yes	No
Yes	1.7	0.3
No	0	2

Support: 85%; confidence: 85%.

6.5. Rules for NB

The rules for NB are generated with $c = 85$ and $m = 4$ as follows:

```
IF CG > 0.2292 THEN we should choose NB
classifier.
```

6.6. Rules for OneR

The rules for OneR are generated with $c = 70$ and $m = 2$ as follows:

```
IF (e <= 1728 AND k > 19.8879) THEN we
should choose OneR classifier.
```

6.7. Rules for SVM

The rules for SVM are generated with $c = 70$ and $m = 2$ as follows:

```
IF Pr <= 364.6066 THEN we should
choose SVM classifier.
```

6.8. Rules for NN

The rules for NN are generated with $c = 80$ and $m = 2$ as follows:

```
IF std <= 9970.047 THEN we should
choose NN classifier.
```

We found the confidence levels of all the generated rules were more than 75%. The results have been tested using 10-fold cross-validation. The rules for SVM

Table 14
SVM classifier selection rule evaluation averaged across 10-fold cross-validation test sets

Condition satisfied	SVM technique best	
	Yes	No
Yes	2.8	0.2
No	0	3

Support: 93.33%; confidence: 93.33%.

Table 15
NN classifier selection rule evaluation averaged across 10-fold cross-validation test sets

Condition satisfied	NN technique best	
	Yes	No
Yes	2.7	0.3
No	0	3

Support: 90%; confidence: 90%.

selection showed higher confidence performance. These rules might be useful to determine which technique is most appropriate for a new classification task.

7. Conclusions

The relative weighted performance measures showed that there was no single classifier to solve all 100 classification problems with best performance over the experiments. The majority of classifiers showed similar efficiency based on the empirical results. Researchers have tended to focus on measuring algorithm performance based on the percentage of correct classification measure. The classifiers C4.5, NN and SVM were all very competitive as the best choices in our present study based on this measure. But the statistical *t*-test showed there were no significance differences among the classifiers when considering a range of accuracy measures and computational complexity. The most recent classifier, SVM, received the highest number of best rankings for all accuracy measures except TPR. But it did not show any best rank for computational time for any dataset. We may argue that, as suggested by the NFL theorem, there is no unique classifier that is likely to perform best for all problems. Therefore, we may be guided to choose the most suitable algorithm for a particular problem by the proposed rule-based method.

In this research, we have proposed a rule-based classifier selection approach, based on the prior knowledge on problem characteristics and the empirical results generated on 100 datasets with 8 classifier algorithms. The main aim of this research is to assist in the selection of an appropriate classification algorithm without the need for trial-and-error testing of the vast array of available algorithms. The decision tree induction algorithm C5.0 performed well to generate the rules for algorithm selection. The confidence levels of all the rules were over 83% for C4.5, NN and SVM, which had sample sizes greater than 25 datasets. The other rules' confidence levels were still more than 75% despite their limited sample size. We suggest considering more classification problems to extract better rules for IBM, PART, KD, NB and OneR classifiers. Our aim has been to determine the rules governing when certain algorithm (using the default parameter settings in WEKA only) should be recommended. Naturally, this research could be extended to consider fine-tuning of each algorithm and optimal feature selection to improve the performance of individual algorithm. This is beyond the scope of the present study however.

Appendix A. Basic properties of the 100 datasets

Dataset name	Majority instances	Minority instances	Number of attributes
abalone	2854	1323	8
adult + stretch	12	8	4
adult-stretch	12	8	4
agaricus-lepiota	4208	3916	22
ann1	3679	93	21
ann2	3355	73	21
att	454	446	9
Australian	383	307	14
bcw	458	241	9
bcw_noise	444	239	18
bio	121	68	5
bld	200	145	6
bld_noise	200	145	15
bos	598	312	13
bos_noise	339	167	25
breast-cancer-wisconsin	458	241	9
bupa	200	145	6
c	2500	2500	15

Appendix A. *(Continued)*

Dataset name	Majority instances	Minority instances	Number of attributes
car	1594	134	6
cmc	1140	333	9
cmc_noise	1140	333	15
crx	273	217	15
darp	1000	500	41
dbha	1400	700	41
dcpga	3002	1000	41
dcmpa	4202	1400	41
dero	1000	27	41
depov	1400	27	41
demoh	1000	563	41
dfgh	3002	500	41
dguv	4202	700	41
dmkot	500	27	41
dna	1051	949	60
DNA-n	2419	767	60
eaot	700	27	41
ebrop	563	500	41
ecot	700	563	41
edrop	3002	27	41
efrgo	4202	27	41
erovs	4202	563	41
emro	563	27	41
enprq	563	27	41
echocardiogram	88	43	7
flare	1171	218	10
german	700	300	24
hayes-roth	81	51	5
hco	209	122	19
h-d	164	139	13
hea	150	120	13
hea_noise	150	120	20
heartdiseas_Hungarian	279	15	13
hep	111	29	19
hepatitis	85	70	19
horse-23	232	136	22
horse-colic	244	124	27
hv84	267	168	16
hyp	2711	136	15
hypothyroid	3012	151	25
kr-vs-kp	1669	1527	36
letter-a	6404	263	16
monk1	278	278	6
monk2	395	206	6
monk3	288	266	6
mushroom	4208	3916	22
musk2	5581	1017	166
nettalk_stress	2910	2528	7
new-thyroid	150	65	5
page-blocks	5056	417	10
pendigits-8	3162	336	16
pid	355	177	7
pima	500	268	8

Appendix A. *(Continued)*

Dataset name	Majority instances	Minority instances	Number of attributes
post-operative	66	24	8
primary-tumor	224	115	17
promoter	53	53	57
sick	2870	293	25
sick-euthyroid	2870	293	25
smo	1290	565	8
smo_noise	1290	565	15
sonar	111	97	60
splice-EI	2415	762	60
t_series	35	27	2
tae	101	50	5
tae_noise	102	49	10
thy	4116	321	21
thy_noise	3488	284	35
tic-tac-toe	626	332	9
titanic	1490	711	3
tmris	51	49	3
ttt	626	332	9
vehicle	346	312	18
votes	267	168	16
wav	2400	1200	21
waveform	2000	1000	40
waveform_noise	3304	1696	21
wav_noise	3345	1655	40
wdbc	357	212	30
wine	107	71	13
wpbc	152	47	33
xaa	54	40	18
xab	51	43	18

Appendix B. Ranked algorithm performance (based on percentage of correct classifications) for the eight algorithms on each dataset

Dataset name	IBK	C4.5	PART	KD	NB	OneR	SVM	NN
abalone	0.5113	0.9315	0.8968	0.6961	0.0000	0.5553	0.8349	1.0000
adult + stretch	0.8133	1.0000	1.0000	1.0000	1.0000	0.0000	1.0000	1.0000
adult-stretch	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	1.0000	1.0000
agaricus-lepiota	1.0000	1.0000	1.0000	1.0000	0.0000	0.6457	1.0000	1.0000
ann1	0.1680	1.0000	0.9920	0.3280	0.2400	0.0000	0.0960	0.4000
ann2	0.3070	1.0000	0.8509	0.4474	0.2018	0.0000	0.1667	0.3333
att	0.2818	0.6077	0.3278	0.4788	0.7772	0.0000	0.9982	1.0000
Australian	0.3235	0.8442	0.8335	0.3377	0.0000	1.0000	0.9941	0.6423
bcw	0.6826	0.5349	0.5928	0.7066	0.8483	0.0000	1.0000	0.7066
bcw_noise	0.8609	0.5665	0.6176	0.5399	0.9100	0.0000	1.0000	0.9366
bio	0.3239	0.6007	0.5439	0.4366	1.0000	0.0000	0.8172	0.8039
bld	0.7911	0.7727	0.6698	0.6717	0.0000	0.3873	0.4813	1.0000
bld_noise	0.0252	1.0000	0.8957	0.0351	0.1987	0.0000	0.5549	0.9083
bos	1.0000	0.7440	0.6244	0.9882	0.0000	0.3181	0.1004	0.6614
bos_noise	0.0000	0.8513	0.6356	0.0964	0.1078	0.8709	0.9412	1.0000
breast-cancer-wisconsin	0.7928	0.5513	0.4708	0.7062	0.8551	0.0000	1.0000	0.8008

Appendix B. *(Continued)*

Dataset name	IBK	C4.5	PART	KD	NB	OneR	SVM	NN
bupa	0.6133	0.7910	0.7294	0.6894	0.0000	0.1217	0.2010	1.0000
c	0.8994	0.7384	0.9780	0.9141	0.7684	0.0000	0.7734	1.0000
car	0.3868	0.7113	0.8530	0.3894	0.4318	0.0000	0.7616	1.0000
cmc	0.0086	0.8333	0.6593	0.0000	0.2346	0.9815	1.0000	0.6963
cmc_noise	0.0000	1.0000	0.6032	0.0839	0.3116	0.9867	1.0000	0.9814
crx	0.6151	0.9446	0.8166	0.6514	0.0000	1.0000	1.0000	0.7697
darp	1.0000	0.8711	0.9072	0.8969	0.0000	0.4485	0.5412	0.8711
dbha	1.0000	0.8719	0.9677	0.9924	0.0000	0.8491	0.9118	0.9070
dcpga	0.9775	0.8707	0.9297	1.0000	0.0000	0.9143	0.9361	0.9691
dcmpa	1.0000	0.2667	0.5667	1.0000	0.4667	0.0000	1.0000	1.0000
dero	0.9444	0.6852	0.6111	0.9815	0.0000	0.6852	0.8889	1.0000
depov	1.0000	0.9574	1.0000	1.0000	0.0000	0.8511	1.0000	0.9574
demoh	1.0000	0.4583	0.4583	0.9250	0.4917	0.0000	0.4583	0.8417
dfgh	0.9200	0.8400	0.8400	0.9200	0.4400	0.0000	1.0000	1.0000
dguv	0.9934	0.9883	0.9914	1.0000	0.0000	0.9654	0.9771	0.9873
dmkot	1.0000	0.8555	0.9609	1.0000	0.4023	0.0000	0.9688	0.9883
dna	0.9853	0.9529	0.9912	1.0000	0.9294	0.0000	0.9912	0.9912
DNA-n	0.7083	0.6528	0.6528	1.0000	0.0000	0.8889	0.8194	0.6528
eaot	1.0000	0.0000	0.0395	1.0000	0.6184	0.2895	0.8026	1.0000
ebrop	0.9193	0.8327	0.8701	1.0000	0.0000	0.6654	0.9134	0.9567
ecot	1.0000	1.0000	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000
edrop	0.9867	0.8267	0.8867	1.0000	0.0000	0.7933	0.9467	0.9600
efrgo	0.8000	0.6500	0.5000	1.0000	0.4000	0.0000	0.6000	0.7500
erovs	0.7561	0.6341	1.0000	0.9756	0.0000	0.2927	0.9756	0.9756
emro	0.0000	1.0000	0.9411	0.3611	0.6320	0.5142	0.4223	0.5107
enprq	0.0464	0.9642	0.9272	0.0000	0.7748	0.1010	0.8187	1.0000
echocardiogram	0.0000	0.5494	0.3637	0.4718	1.0000	0.3067	0.9882	0.5793
flare	0.0000	1.0000	0.6540	0.3030	0.1641	0.9874	0.9444	0.3939
german	0.0000	0.5649	0.2830	0.3872	0.8606	0.2936	1.0000	0.4021
hayes-roth	0.6338	0.9474	1.0000	0.5478	0.2619	0.0000	0.1620	0.7146
hco	0.5855	1.0000	0.6703	0.3001	0.2611	0.6958	0.6676	0.0000
h-d	0.3816	0.6068	0.6032	0.4012	1.0000	0.0000	0.9833	0.6821
hea	0.3264	0.5309	0.4699	0.3063	1.0000	0.0000	0.9679	0.3857
hea_noise	0.3628	0.3209	0.4652	0.4488	1.0000	0.0000	0.8504	0.5707
heartdiseas_Hungarian	0.0000	0.7866	0.8534	0.4976	1.0000	0.8701	0.9380	0.8497
hep	0.3333	0.0000	0.3743	0.2935	0.7725	0.4721	1.0000	0.3333
hepatitis	0.3091	0.0000	0.4942	0.1504	0.8298	0.3008	1.0000	0.1950
horse-23	0.4052	1.0000	0.8529	0.0000	0.0049	0.6356	0.4984	0.1127
horse-colic	0.0000	0.5297	0.5427	0.6611	0.5065	1.0000	0.5909	0.6060
hv84	0.4184	0.8793	1.0000	0.4099	0.0000	0.9456	0.9541	0.8673
hyp	0.0000	1.0000	0.8197	0.1502	0.3991	0.2403	0.0858	0.3605
hypothyroid	0.0000	1.0000	0.8866	0.0773	0.2835	0.2165	0.0412	0.2474
kr-vs-kp	0.8577	1.0000	0.9911	0.8928	0.6405	0.0000	0.8868	0.9962
letter-a	1.0000	0.8323	0.8230	0.9814	0.5932	0.0000	0.7609	0.9193
monk1	0.6450	0.8871	0.8455	0.7750	0.0000	0.1886	0.0004	1.0000
monk2	0.8183	0.8915	1.0000	0.8277	0.0000	0.0060	0.0060	0.5438
monk3	0.1060	1.0000	0.9963	0.1551	0.6218	0.0000	0.0100	0.9098
mushroom	0.9989	1.0000	0.9968	1.0000	0.0000	0.8236	0.5736	0.7760
musk2	0.7177	0.7735	0.8122	0.5660	0.0000	0.3272	0.6633	1.0000
nettalk_stress	1.0000	0.4501	0.5894	0.9927	0.5052	0.0000	0.8857	0.6840
new-thyroid	1.0000	0.8187	0.7851	0.8782	0.7803	0.7233	0.0000	0.9640
page-blocks	0.7851	1.0000	0.9731	0.7672	0.2597	0.0000	0.3224	0.7940
pendigits-8	1.0000	0.9433	0.9515	0.9943	0.0000	0.4842	0.8493	0.9931

Appendix B. *(Continued)*

Dataset name	IBK	C4.5	PART	KD	NB	OneR	SVM	NN
pid	0.0000	0.7326	0.6221	0.1950	0.8635	0.5803	1.0000	0.6927
pima	0.0000	0.4853	0.4791	0.1283	0.7929	0.1917	1.0000	0.6940
post-operative	0.1077	1.0000	0.2745	0.0000	0.7352	0.8826	0.8826	0.0388
primary-tumor	0.3990	0.5859	0.8258	0.4545	0.5202	0.0202	1.0000	0.0000
promoter	0.5175	0.3818	0.6721	0.5921	0.9600	0.0000	1.0000	0.9760
sick	0.5705	1.0000	0.9899	0.6594	0.0000	0.5940	0.1913	0.7131
sick-euthyroid	0.8488	1.0000	0.9835	0.8631	0.0000	0.9045	0.7904	0.9110
smo	0.0000	0.8804	0.8016	0.0452	0.9956	0.9159	1.0000	0.9159
smo_noise	0.0492	0.3956	0.7147	0.0000	0.9815	0.7147	1.0000	0.6035
sonar	1.0000	0.4447	0.4861	0.9730	0.2186	0.0000	0.6367	0.8502
splice-EI	0.2756	1.0000	0.9570	0.4085	0.9673	0.0000	0.9308	0.9924
t_series	0.0000	1.0000	1.0000	0.6118	0.5897	0.3711	0.2001	0.4879
tae	1.0000	0.3064	0.1779	0.6264	0.0000	0.1243	0.3183	0.2834
tae_noise	0.3677	0.2973	0.0000	0.3230	1.0000	0.5756	0.6443	0.1065
thy	0.2193	1.0000	0.9697	0.2549	0.2478	0.4973	0.0000	0.5027
thy_noise	0.0000	1.0000	0.9960	0.1918	0.6902	0.8471	0.5198	0.4200
tic-tac-toe	0.6565	0.7880	0.8993	1.0000	0.1301	0.0000	0.0187	0.3123
titanic	0.9559	0.8529	1.0000	0.9559	0.0000	0.0294	0.0294	0.8015
tmris	0.0000	0.4211	0.4211	0.9474	0.3684	0.8947	1.0000	0.8421
ttt	0.9993	0.5444	0.8448	1.0000	0.0000	0.0052	0.9825	0.9491
vehicle	0.7329	0.7115	0.7239	0.7324	0.0000	0.1010	0.4508	1.0000
votes	0.3142	1.0000	0.9633	0.3110	0.0000	0.8676	0.9777	0.6667
wav	0.6567	0.6059	0.7660	0.6128	0.6798	0.0000	0.8853	1.0000
waveform	0.6870	0.6417	0.7843	0.5591	0.7130	0.0000	0.9983	1.0000
waveform_noise	0.4676	0.5464	0.7215	0.2014	0.6646	0.0000	1.0000	0.9807
wav_noise	0.4046	0.5714	0.6796	0.3064	0.6697	0.0000	0.9298	1.0000
wdbc	0.7861	0.6428	0.6319	0.8078	0.5114	0.0000	1.0000	0.9066
wine	0.8076	0.6540	0.7185	0.8076	0.8521	0.0000	1.0000	0.9479
wpbc	0.4204	0.6341	0.8085	0.4355	0.0000	0.1663	0.9607	1.0000
xaa	0.4783	0.5011	0.5995	0.4901	0.2139	0.0000	0.3370	1.0000
xab	0.7361	0.4697	0.6019	0.7516	0.3889	0.0000	0.3143	1.0000

Appendix C. Count of best and worst performances by each algorithm across the set of 100 problems, using different types of measures

1. Percentage of correct classification measure.

	Classifier							
	IBK	C4.5	PART	KD	NB	OneR	SVM	NN
Worse	16	3	1	5	32	39	2	2
Best	17	26	11	18	9	4	26	25

2. TPR measure.

	Classifier							
	IBK	C4.5	PART	KD	NB	OneR	SVM	NN
Worse	8	6	1	3	28	40	17	3
Best	27	25	20	25	27	13	30	32

Appendix C. *(Continued)*

3. TNR measure.

	Classifier							
	IBK	C4.5	PART	KD	NB	OneR	SVM	NN
Worse	12	6	2	5	38	38	14	7
Best	20	26	22	25	26	15	33	28

4. Weighted F-measure.

	Classifier							
	IBK	C4.5	PART	KD	NB	OneR	SVM	NN
Worse	16	3	1	8	24	38	18	3
Best	29	28	26	23	20	11	30	27

5. Computational complexity measure.

	Classifier							
	IBK	C4.5	PART	KD	NB	OneR	SVM	NN
Best	7	7	4	6	41	99	0	0
Worse	0	0	0	4	0	0	0	97

Appendix D. Formulation for data characteristics measurement

D.1. Statistical measures

D.1.1. Geometric mean (GM)

The geometric mean of a sequence $\{X_i\}_{i=1}^{n}$ is:

$$\mathrm{GM} = \left[\prod_{i=1}^{n} X_i\right]^{1/n}.$$

D.1.2. Harmonic mean (HM)

The harmonic mean $\mathrm{HM}(X_1, \ldots, X_n)$ of n points X_i is:

$$\mathrm{HM} = \frac{n}{\sum_{i=1}^{n} 1/X_i}.$$

D.1.3. Trim mean (TM)

The trim mean measures the arithmetic mean of a sample X excluding the specified trim fraction from the same variable. The trim fraction is user dependent parameter. We consider 20 for this parameter value over the experiment. The trimmed mean is a robust estimate of the center location of a sample. For outliers' dataset, the trimmed mean is a more appropriate estimation of the center of the dataset.

D.1.4. Mad (M)

The mad estimates the mean absolute deviation of a dataset [35].

D.1.5. Variance (V)

The variance is use to characterize the dispersion among the measures in a given population. It calculates the mean of the scores, and then measures the amount that each score deviates from the mean and then squares that deviation for a given population. Numerically, the variance equals the average of the squared deviations from the mean [36].

D.1.6. Standard deviation (std)

The std measures the spread of a set of data as a proportion of its mean:

$$\mathrm{std} = \left(\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{1/2},$$

where $\bar{x} = \frac{1}{n}\sum x_i$, n is the sequence of lengths.

D.1.7. Prctile

Prctile calculates a value for a variable that is greater than a certain percentage in the same variable. We consider the percentage value is 90 over the all datasets.

Some other simple statistical measures including mean and median are also considered to characterise the datasets, for details see [16].

D.1.8. Interquartile range (IQR)

The IQR is used as a robust measure of scale and measures the distance between the 25th and the 75th percentile [29]. The hypothesis is, if the variables are approximately normal, then IQR/$\sigma \approx 1.3$, where σ is the standard deviation of the population. Another name of 25th percentile is semi-interquartile range (siqr).

D.1.9. Maximum and minimum eigenvalues

The maximum and minimum eigenvalues are the maximum and minimum variances of a dataset. We use the sample covariance matrix to calculate the eigenvalues:

$$\Re = \frac{1}{n}\sum_{p_{i=1}}^{n} X(p_i)' X(p_i).$$

D.1.10. Canonical correlation (CC)

Correlation coefficients can be interpreted by the square root of the eigenvalues of a matrix. Because the correlations pertain to the canonical variates, so they are called *canonical correlations* for details [30].

D.1.11. Index of dispersion (ID)

The larger value of ID indicates the datasets are widely scattered, otherwise it is closely clustered [31]:

$$\mathrm{ID} = \frac{k(N^2 - \sum f^2)}{N^2(c-1)},$$

where N is the number of scores, c is the number of categories of the variables and $\sum f^2$ is the sum of the squared frequencies over the categories.

D.1.12. Center of gravity (CG)

CG measures the Euclidean norm between minority and majority classes. The minimum value indicates the closeness between groups and the maximum indicates the dispersion between groups:

$$\mathrm{CG} = \sqrt{(a_{i,j} - b_{i,j})},$$

where a and b belong to the center points of these two groups.

D.1.13. Skewness and kurtosis

The skewness (s) and kurtosis (k) [16] of a distribution are defined as:

$$s = \frac{\boldsymbol{E}(X-\mu)^3}{\sigma^3}, \qquad k = \frac{\boldsymbol{E}(X-\mu)^4}{\sigma^4},$$

where μ is the mean and σ is the standard deviation of X.

D.1.14. Correlation coefficient

The sample correlation coefficient between X and Y is denoted by r_{xy} or simply by r [32] as follows:

$$r = \frac{s_{xy}}{s_x s_y} = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{X_i - \bar{X}}{\sigma_x}\right)\left(\frac{Y_i - \bar{Y}}{\sigma_y}\right).$$

D.1.15. Z-score

The value of Z-score is greater than 3 indicates that the data distribution has outliers [32].

$$Z\text{-score} = \frac{X - \bar{X}}{\sigma}.$$

D.1.16. Normal cumulative distribution test

It computes the normal cumulative distribution function [16] of a normal distribution is:

$$p_{\mathrm{ncdf}} = F(X|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{X} \mathrm{e}^{-(t-\mu)/2\sigma^2}\,\mathrm{d}t.$$

D.1.17. Chi-square test

The cumulative probability density function [16] of chi-square test is:

$$p_{\mathrm{cst}} = F(X|\nu) = \int_0^x \frac{t^{(\nu-2)/2}\mathrm{e}^{-t/2}}{2^{\nu/2}\Gamma(\nu/2)}\,\mathrm{d}t,$$

where ν is the degree of freedom and $\Gamma(\cdot)$ is the gamma function.

D.2. Information theoretical measures

D.2.1. Mean entropy of variables

Entropy is a measure of randomness in a variable. The entropy $H(X)$ of a discrete random variable X is calculated in terms of q_i (the probability that X takes on the ith value). We average the entropy over all the

variable and take this as a global measure of the entropy of the variables:

$$H(X) = -\sum q_i \log q_i, \qquad \bar{H}(X) = p^{-1} \sum H(X_i).$$

D.2.2. Entropy of classes

This is similar to the entropy of variables, except that the randomness in class assignment is measured, where π_i is the prior probability of class A_i:

$$H(C) = -\sum_i \pi_i \log \pi_i.$$

D.2.3. Mean mutual entropy of class and variables

For a measure of common information or entropy shared between the two variables, if p_{ij} denotes the joint probability of observing class A_i and the jth value of variable X, if the marginal probability of class A_i is π_i, and if the marginal probability of variable X taking on its jth value of q_j, then the mutual information and its mean over all variables as defined as:

$$M(C,X) = \sum_{ij} p_{ij} \log\left(\frac{p_{ij}}{\pi_i q_i}\right) \quad \text{and}$$

$$\bar{M}(C,X) = p^{-1} \sum_i M(C,X_i).$$

D.2.4. Equivalent number of variables (ENV)

This is the ratio between the class entropy and the average mutual information:

$$\text{ENV} = \frac{H(C)}{\bar{M}(C,X)}.$$

D.2.5. Noise–signal ratio (NSR)

$$\text{NSR} = \frac{\bar{H}(X) - \bar{M}(C,X)}{\bar{M}(C,X)}.$$

A large NS ratio implies that a dataset contains much irrelevant information (noise) and could be condensed without affecting the performance of the model.

References

[1] D.H. Wolpert, W.G. Macready, No Free Lunch theorem for search, Technical Report SFI-TR-05-010, Santa Fe Institute, Santa Fe, NM, 1995.

[2] T.-S. Lim, W.-E. Loh, Y.-S. Shih, A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, Mach. Learn. 40 (2000) 203–229.

[3] D. Michie, D.J. Spiegelhalter, C.C. Taylor, Machine Learning, Neural and Statistical Classification, Ellis Horwood, New York, 1994.

[4] K.A. Smith, F. Woo, V. Ciesielski, R. Ibrahim, Matching data mining algorithm suitability to data characteristics using a self-organising map, in: A. Abraham, M. Koppen (Eds.), Hybrid Information Systems, Physica-Verlag, Heidelberg, 2002, pp. 169–180.

[5] C.E. Brodley, P.E. Utgoff, Multivariate versus univariate decision trees, Technical Report 92-8, Department of Computer Science, University of Massachusetts, Amherst, MA, 1992.

[6] D.E. Brown, V. Corruble, C.L. Pittard, A comparison of decision tree classifiers with backpropagation neural networks for multimodal classification problems, Pattern Recognit. 26 (1993) 953–961.

[7] S.P. Curram, J. Mingers, Neural networks, decision tree induction and discriminant analysis: an empirical comparison, J. Operational Res. Soc. 45 (1994) 440–450.

[8] J.W. Shavlik, R.J. Mooney, G.G. Towell, Symbolic and neural learning algorithms: an empirical comparison, Mach. Learn. 6 (1991) 111–144.

[9] G.M. Weiss, F. Provost, The effect of class distribution on classifier learning, Technical Report ML-TR-43, Department of Computer Science, Rutgers University, 2001.

[10] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: a review, IEEE Trans. Pattern Anal. Mach. Intell. 22 (1) (2000).

[11] C. Blake, C.J. Merz, UCI Repository of machine learning databases, University of California, Irvine, CA, 2002, http://www.ics.uci.edu/~mlearn/mlrepository.html.

[12] T.-S. Lim, Knowledge Discovery Central datasets, 2002, http://www.kdcentral.com/.

[13] R.J. Henery, Methods of comparison, in: D. Michie, D.J. Spiegelhalter, C.C. Taylor (Eds.), Machine Learning, Neural and Statistical Classification, Ellis Horwood, New York, 1994 (Chapter 7).

[14] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tool and Technique with Java Implementation, Morgan Kaufmann, San Francisco, 2000.

[15] K.A. Smith, F. Woo, V. Ciesielski, R. Ibrahim, Modelling the relationship between problem characteristics and data mining algorithm performance using neural networks, in: C. Dagli, et al. (Eds.), Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining, and Complex Systems, vol. 11, ASME Press, 2001, pp. 357–362.

[16] Statistics Toolbox User's Guide, version 3, The MathWorks Inc., USA, 2001.

[17] R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, San Mateo, CA, 1993.

[18] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., Wiley, New York, 2001.

[19] V.N. Vapnik, An overview of statistical learning theory, IEEE Trans. Neural Netw. 10 (5) (1999).

[20] R.C. Holte, Very simple classification rules perform well on most commonly used dataset, Mach. Learn. 11 (1993) 63–91.
[21] E. Frank, I.H. Witten, Generating accurate rule sets without global optimisation, in: J. Shavlik (Ed.), Machine Learning, Proceedings of the Fifteenth International Conference, Morgan Kaufmann, San Francisco, CA, 1998.
[22] R.P.W. Duin, A note on comparing classifier, Pattern Recognit. Lett. 1 (1996) 529–536.
[23] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, 1998.
[24] G.H. John, P. Langley, Estimating continuous distributions in Bayesian classifiers, in: Proceeding of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1995, pp. 338–345.
[25] H. Lodhi, J. Shawe-Taylor, N. Christianini, C. Watkins, Text classification using string kernels, in: T. Leen, T. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems, vol. 13, MIT Press, 2001.
[26] R. Kohavi, A study of cross validation and bootstrap for accuracy estimation and model selection, in: Proceedings of the 14th IJCAI, Morgan Kaufmann, San Francisco, CA, 1995, pp. 338–345.
[27] C.J. Van Rijsbergen, Information retrieval, second ed., Butterworths, London, 1979.
[28] D.D. Lewis, Evaluating and optimizing autonomous text classification systems, in: ACM-SIGIR Conference Research on Development in Information Retrieval, 1995, 246–254.
[29] W. Mandenhall, T. Sincich, Statistics for Engineering and the Sciences, fourth ed., Prentice-Hall, 1995.
[30] R. Gnanadesikan, Methods for Statistical Data Analysis of Multivariate Observations, second ed., Wiley, USA, 1997.
[31] J.L. Craft, Statistics and Data Analysis for Social Workers, second ed., F.E. Peacock Publishers, USA, 1990.
[32] A.C. Tamhane, D.D. Dunlop, Statistics and Data Analysis, Prentice-Hall, USA, 2000.
[33] R. Quinlan, See5: An Informal Tutorial, http://www.rulequest.com/see5-win.html.
[34] V.B. Bajic, Comparing the success of different prediction software in sequence analysis: a review, Brief. Bioinform. 1 (3) (2000) 214–228.
[35] L. Sachs, Applied Statistics: A Handbook of Techniques, Springer-Verlag, 1984 p. 253.
[36] D.R. Anderson, D.J. Sweeney, T.A. Williams, N.J. Harrison, J.A. Rickard, Statistics for Business and Economics, Australian ed., West Publishing Company, USA, 1992.
[37] G.D. Magoulas, V.P. Plagianakos, M.N. Vrahatis, Neural network-based colonoscopic diagnosis using on-line learning and differential evolution, Appl. Soft Comput. 4 (2004) 369– 379.
[38] F.L. Chung, W. Shitong, D. Zhaohong, H. Dewen, Fuzzy kernel hyperball perceptron, Appl. Soft Comput. 5 (2004) 67–74.