Comparison of 14 different families of classification algorithms on 115 binary datasets

Jacques Wainer
email: wainer@ic.unicamp.br
Computing Institute
University of Campinas
Campinas, SP, 13083-852, Brazil

June 6, 2016

Abstract

We tested 14 very different classification algorithms (random forest, gradient boosting machines, SVM - linear, polynomial, and RBF - 1-hidden-layer neural nets, extreme learning machines, k-nearest neighbors and a bagging of knn, naive Bayes, learning vector quantization, elastic net logistic regression, sparse linear discriminant analysis, and a boosting of linear classifiers) on 115 real life binary datasets. We followed the Demsar analysis and found that the three best classifiers (random forest, gbm and RBF SVM) are not significantly different from each other. We also discuss that a change of less then 0.0112 in the error rate should be considered as an irrelevant change, and used a Bayesian ANOVA analysis to conclude that with high probability the differences between these three classifiers is not of practical consequence. We also verified the execution time of "standard implementations" of these algorithms and concluded that RBF SVM is the fastest (significantly so) both in training time and in training plus testing time.

keywords: Classification algorithms; comparison; binary problems; Demsar procedure; Bayesian analysis

1 Introduction

Fernández-Delgado et al. (2014) evaluated 179 different implementations of classification algorithms (from 17 different families of algoritms) on 121 public datasets. We believe that such highly empirical research are very important both for researchers in machine learning and specially for practitioners.

For researchers, this form of research allows them to focus their efforts on more likely useful endeavors. For example, if a researcher is interested in developing algorithms for very large classification problems, it is probably more useful to develop a big-data random forest (which is the family of classification algoritms with best performance according to Fernández-Delgado et al. (2014)) than to do it for Bayesian networks (mainly naive Bayes) or even Nearest Neighbors methods, which perform worse than random forests.

For practitioners, this form of research is even more important. Practitioners in machine learning will have limited resources, time, and expertise to test many different classification algorithms on their problem, and this form of research will allow them to focus on the most likely useful algorithms.

Despite its importance and breath, we believe that Fernández-Delgado et al. (2014) had some "imperfections" which we address in this research. The "imperfections" are discussed below as the extensions we carried in this paper:

- We used the same set of datasets, but we transform them so that the problems are all binary. Many classification algorithms are defined only to binary problems, for example the SVM family, logistic regression, among others. Of course there are meta-extensions of such algorithms to multi-class problems, for example, one-vs-one, one-vs-all, error correcting output coding (ECOC) (Dietterich and Bakiri, 1995), stacked generalization (Wolpert, 1992), pairwise coupling (Hastie et al., 1998), among others. Also, there are alternative formulations for specific binary classifiers to deal with multiclass, for example, Franc et al. (2002) for SVM, Engel (1988) for logistic regression, and so on.

  For the algorithms that are intrinsically binary, the application to multiclass problems poses two problems. The first is that one has to decide on which meta-extension to use, or if one should use the specific multiclass formulation of the algorithm. In some cases, one meta-extension is implemented by default, and the user must be aware that this decision was already made for him/her. For example, the libSVM default approach is one-vs-one. But a second, more subtle problem is the search for hyperparameters: it is very common that each combination of hyperparameters are tested only once for all classifiers in the one-vs-one solution. That is, all $n(n-1)/2$ classifiers in a one-vs-one solution has the same set of hyperparameters, and that may cause a decrease in accuracy in comparison to the case in which classifier is allow to choose its one set of hyperparameter. Thus, on multiclass

problems, those intrinsically binary algorithms could be facing many disadvantages.

On the issue of binary classifiers, Fernández-Delgado et al. (2014) did not include in their comparisons the gradient boosting machine (`gbm`) algorithm, considered a very competitive algorithm for classification problems because, as reported in Zajac (2016), the implementation did not work in multiclass problems. We included `gbm` in our comparison.

- We reduced the number of classifiers to only a few classes/algorithms and not different implementations of the same algorithm. Fernández-Delgado et al. (2014) compared an impressing 179 different classification programs, but it was unclear how many were just different implementations of the same algorithm, and how many were variations within the same "family" of algorithms. We believe that for practitioner and research communities, it is more useful to have an understanding of how different families of algorithms rank in relation to each other. For the practitioner, which should have more limited access to the different algorithms, this knowledge allow them to order which algorithms should be applied first to their particular problem.

  In fact, Fernández-Delgado et al. (2014) also perform an analysis of their results based on the algorithm's "family", but they have difficulty of extracting useful information from this analysis, since in most cases, different "implementations" in the same family have widely different results. In one analysis, they rank the families by the worse performing member, which does not provide useful information. But in the end, their conclusions are mainly centered on the families of classifiers, because that is the most useful level of knowledge. From the abstract of the paper:

  > A few models are clearly better than the remaining ones: random forest, SVM with Gaussian and polynomial kernels, extreme learning machine with Gaussian kernel, C5.0 and avNNet (a committee of multi-layer perceptrons implemented in R with the caret package). The random forest is clearly the best family of classifiers (3 out of 5 bests classifiers are RF), followed by SVM (4 classifiers in the top-10), neural networks and boosting ensembles (5 and 3 members in the top-20, respectively).

- We performed a more careful search for hyperparameters for each clas-

sification algorithm. Given Fernández-Delgado et al. (2014) daunting task of testing 179 programs, they had to rely on default values for the hyperparameters which may lead to suboptimal results. Since we are working with significantly fewer algorithms, we could spend more time selecting reasonable ranges of values for the hyperparameters. In particular we tried to limit the number of training steps for the hyperparameter search in 24 so that no algorithm would have an advantage of having more degrees of freedom to adjust to the data (but in section 2.4 we discuss that some algorithms may allow testing many values of the hyperparameter with just one training and testing step).

- Besides computing when two algorithms are significantly different in their error rates, in the standard null-hypothesis significance tests (NHST), we also use a Bayesian analysis to discover when the differences of two algorithms has no "practical consequences." As far as we know, this is the first time a Bayesian ANOVA is used to compare classifiers, which we believe is an important direction in analysis of results in empirical machine learning. But more significantly is the use of "limits of practical significance", that is, the definition of thresholds below which the differences are irrelevant from practical purposes, which goes beyond an more standard "significance test" analysis currently used in the machine learning literature.

  Fernández-Delgado et al. (2014) follow the standard null hypothesis significant test in analyzing their result, but even within this framework, their analysis is not as rigorous as it should have been. The NIST standard for comparing many classifiers across different datasets was proposed by Demsar (2006) and it is discussed in section 3.1. In particular, when testing the different algorithms for statistical significant differences, the Demsar procedure requires one to use the Nemenyi test, which is a nonparametric test that performs the appropriate multiple comparison p-value correction. But Fernández-Delgado et al. (2014) used a paired t-test (a parametric test) between the first ranked and the following 9 top ranked algorithms, apparently without multiple comparions corrections. Very likely, given the large number of comparisons need to contrast all 179 algorithms, very few, if any of the pairwise comparisons would have been flagged as significant by the Nemenyi test.

  In this paper we followed the full Demsar procedure to analyse the results, but we also used the Bayesian ANOVA to verify when the

differences between the algoritms is not only "statistically significant", but also "of practical significance".

Fernández-Delgado et al. (2014) first two ranked programs are two different implementation of the same algorithm - parallel implementation of random forest (first ranked) and a non-parallel implementation (second ranked). The authors state that in some sense, the difference between the two results should not be "important" (they use the term "significant") and indeed the statistical significance analysis shows that the difference was not statistically significant, but neither was the next 6 ranked algorithms (in comparison to the top performing).

- We studied the computational costs of running a "standard" implementation of the different algorithms. With information regarding the computational cost a practitioner may want to balance execution time and expected accuracy. Furthermore, this information may encourage the practitioner choose other implementations that the ones tested, and researcher to develop faster implementations of the best ranked algorithms.

2 Data and Methods

2.1 Experimental procedure

In general terms the experimental procedure followed by this research is the following.

Each dataset D_i (the datasets are discussed in section 2.2) is divided in half into two subsets S_{1i} and S_{2i}, each with the same proportion of classes. For each subset S_{ji}, we used a a 5-fold cross validation procedure to select the best set of hyperparameters for the procedure a, $(\hat{\theta}_a)$. Then we trained the whole subset S_{ji} using the procedure a with hyperparameters $\hat{\theta}_j$ and computed the error rate on the subset $S_{\hat{j}i}$, (where $\hat{1} = 2$ and $\hat{2} = 1$). We call the error rate of algorithm a when learning on the subset S_{ji}, with hyperparameters $\hat{\theta}_a$, when testing on the subset $S_{\hat{j}i}$ as $\epsilon(i, \hat{j}|a, \hat{\theta}, j)$.

The expected for procedure a error on (whole) dataset i is $\epsilon(i|a)$ and it is computed as the average

$$\epsilon(i|a) = \frac{\epsilon(i, 1|a, \hat{\theta}, i, 2) + \epsilon(i, 2|a, \hat{\theta}, i, 1)}{2} \tag{1}$$

We then performed the analyses described in sections 3.1 and 3.2 on the sets $\{\epsilon(i|a)\}$ for all datasets i (described in section 2.2 and for all classification algorithms a (section 2.3).

2.2 Datasets

We started with the 121 datasets collected from the UCI site, processed, and converted by the authors of Fernández-Delgado et al. (2014) into a unified format. The datasets is derived from the 165 available at UCI website in March 2013, where 56 were excluded by the authors of Fernández-Delgado et al. (2014). For the remaining 121, Fernández-Delgado et al. (2014) converted all categorical variables to numerical data, and each feature was standardized to zero mean and standard deviation equal to 1.

We downloaded the datasets preprocessed by the authors of Fernández-Delgado et al. (2014) in November 2014. We performed the following transformations:

- 65 of the datasets were multiclass problems. We converted them to a binary classification problem by ordering the classes by their names, and alternatively assigning the original class to the positive and negative classes. The datasets have different proportions between the positive and negative classes, approximately normally distributed with mean 0.6 and standard deviation of 0.17.

- 19 of the datasets were separated into different training and test sets and on the November 2014 data, the test set was not standardized. We standardized the test set (separately from the train set) and joined both sets to create a single dataset.

- we removed the 6 datasets with less than 100 datapoints,

- the 9 datasets with more than 10.000 datapoints (more data points for subset) we searched the hyperparameters on a subset of 5000 datapoints. The final training was done with the whole subset, and the testing with the other full subset.

Thus, in this research we tested the algorithms in 121 - 6 (very small datasets removed) = 115 datasets.

Details of each dataset are described in A.

2.3 Classification algorithms

We used 14 classification algorithms from very different families. As discussed in the Introduction, we argued that one of the possible criticisms to the Fernández-Delgado et al. (2014) paper is that the authors do not distinguish different algorithms from different implementations of that algorithm.

Although we do not have a clear or formal definition of what are "families of algorithms" we believe that we have a sufficiently diverse collection of algorithms.

We chose not to use algorithms that do not require hyperparameters, such as Linear Discriminant Analysis (LDA) and logistic regression. Hyperparameters allow the algorithm to better adjust to the data details, and so we believe LDA and logistic regressions would be in disadvantages to the other algorithm. We added regularized versions of these algorithms, which contain at least one hyperparameters.

Thus, `glmnet` (L1 and L2 regularized logistic regression (Zou and Hastie, 2005)) and `sda` a L1-regularized LDA are two mainly linear algorithms. We would also add `bst` a boosting of linear models.

Among the distance based classifiers, `knn` is the usual k-nearest neighbor, and `rknn` is a bagging of knn, where each base learner is a `knn` on a random sample of the original features. `lvq`, or learning vector quantization (Kohonen, 1995) is a cluster plus distance, or dictionary based classification: a set of "prototypes," or clusters, or "codebook" is discovered in the data, many for each class, and new data is classified based on the distance to these prototypes.

Neural network based classifiers include `nnet` a common 1-hidden layer logistic network, and `elm` or extreme learning machines (Huang et al., 2006). Extreme learning machines are a 1-hidden layer neural network, where the weights from the input to the hidden layer are set at random, and only the second set of weights are learned (usually via least square optimization).

We only tested the naive Bayes `nb` as a Bayesian network based algorithm.

We divided the SVM family into the linear SVM (`svmLinear`), the polynomial SVM (`svmPoly`) and the RBF SVM (`svmRadial`).

Finally we included one implementation of random forests (**rf**) and one implementation of gradient boosting machine classifiers (`gbm`) (Friedman, 2001).

The implementation information of the algorithms are listed below.

bst Boosting of linear classifiers. We used the R implementation in the package *bst* (Wang, 2014)

elm Extreme learning machines Implementation: package *elmNN* (Gosso, 2012))

gbm Gradient boosting machines. Implementation: package *gbm* (Ridgeway, 2013)

glmnet Elastic net logistic regression classifier. Implementation : package *glmnet* (Friedman et al., 2010))

knn k-nearest neighbors classifier. Implementation: package *class* (Venables and Ripley, 2002).

lvq Learning vector quantization. Implementation: package *class* (Venables and Ripley, 2002))

nb Naive Bayes classifier: package *klaR* (Weihs et al., 2005)

nnet A 1-hidden layer neural network with sigmoid transfer function. Implementation: package *nnet* (Venables and Ripley, 2002)

rf Random forest. Implementation: package *randomForest* (Liaw and Wiener, 2002)

rknn A bagging of knn classifiers on a random subset of the original features. Implementation: package *rknn* (Li, 2015)

sda A L1 regularized linear discriminant classifier. Implementation: package *sparseLDA* (Clemmensen, 2012)

svmLinear A SVM with linear kernel. package *e1071* (Meyer et al., 2014))

svmPoly A SVM with polynomial kernel. package *e1071* (Meyer et al., 2014))

svmRadial A SVM with RBF kernel. package *e1071* (Meyer et al., 2014))

2.4 Hyperparameters ranges

The RandomForest classifier is a particularly convenient algorithm to discuss the grid search on hyperparameters. Most implementation of `rf` use two or three hyperparameters: the `mtry`, the number of trees, and possible some limit of complexity of the trees, either the minimum size of a terminal node, or the maximum profundity, or a maximum number of terminal nodes. We did not set a range of possible values for the hyperparameters that limit the

complexity of the trees. Mtry is the number of random features that will be used to construct a particular tree. There is a general suggestion (we do not know the source or other papers that tested this suggestion) that this number should be the square root of the number of features of the dataset. If nfeat is the number of features of the dataset, we set the possible values to $\{0.5 \times \sqrt{\text{nfeat}}, 1 \times \sqrt{\text{nfeat}}, 2 \times \sqrt{\text{nfeat}}, 3 \times \sqrt{\text{nfeat}}, 4 \times \sqrt{\text{nfeat}}, 5 \times \sqrt{\text{nfeat}}\}$. Also, the range is limited to at most nfeat-1.

The number of trees is what we will call a **free hyperparameter**, that is, an hyper-parameter can be tested for many values but it only need one training step. One can train a random forest with n trees, but at testing time, some implementations can return the classification of each tree on the test data. Thus one can test the accuracy of a random forest with $m < n$ trees, just by computing the more frequent classification of the first m trees (or any random sample of m trees). Thus, to select this hyperparameter, it is not necessary to create a grid with multiple training or testing. So, for the random forest, `ntree` is a free hyperparameter, that will be tested from 500 to 3000 by 500. But we also put another limit on the number of trees, half of the number data points in the subset.

The number of repetitions or boosts in a boosting procedure is also a free parameter. The last class of free hyperparameter refer to an implementation of classification algorithms that calculate all the possible values of a parameter that causes changes in the learned function. The only relevant algorithm for this research is the elastic-net regularized logistic regression implemented by the package glmnet (Friedman et al., 2010)), which computes all the values (or the *path* as it is called) of the regularization parameter λ. Hastie et al. (2004) discuss a complete path algorithm for SVM (for the C hyperparameter) but we did not use this implementation in this paper.

We list the range of hyperparameters for each of the algorithms tested, where nfeat is the number of features in the data and ndat is the number of data points.

bst Hyperparameters: shrinkage = $\{0.05, 0.1\}$. Free hyperparameter, number of boosts, from 100 to 3000 by 200, at most ndat.

elm Hyperparameter: number of hidden units = at most 24 values equally spaced between 20 and ndat/2

gbm Hyperparameters: interaction-depth = 1..5, shrinkage=$\{0.05, 0.1\}$. number of boosts is a free hyperparameter, tested from 50 to 500 with increments of 20 to at most ndat.

glmnet Hyperparameters α, 8 values equally spaced between 0 and 1. Free hyperparameter λ, 20 values geometrically spaced between 10^{-5} to 10^3.

knn Hyperparameter k:= 1, and at most 23 random values from 3 to ndat/4.

lvq Hyperparameter: size of the codebook = at most 8 values equally spaced between 2 and 2× nfeat

nb Hyperparameters: usekernel = { true, false }, fL = $\{0, 0.1, 1, 2\}$

nnet Hyperparameter: number of hidden units = at most 8 values equally spaced between 3 and nfeat/2, decay = $\{0, 0.01, 0.1\}$.

rf Hyperparameter mtry = $\{0.5, 1, 2, 3, 4, 5\}\sqrt{nfeat}$ up to a value of nfeat/2. Number of trees is a free hyperparameters, tested from 500 to 3000 by 500 up to ndat/2.

rknn Hyperparameters: mtry = 4 values equally spaced between 2 and nfeat−2, k= 1, and at most 23 random values from 3 to ndat/4. The number of classifiers is a free hyperparameter from 5 to 100, in steps of 20.

sda Hyperparameter: $\lambda = 8$ values geometrically spaced from 10^{-8} to 10^3

svmLinear . Hyperparameter: C = $2^{-5}, 2^0, 2^5, 2^{10}, 2^{15}$.

svmPoly Hyperparameter C as in the linear kernel and degree from 2 to 5.

svmRadial Hyperparameters C as in the linear SVM and $\gamma = 2^{-15}, 2^{-10.5}, 2^{-6}, 2^{-1.5}, 2^3$.

2.5 Reproducibility

The data described above, the R programs that tested the 14 algorithms, the results of running there algorithms, the R programs used to analyse these results and generate the tables and figures in this paper, and the saved interactions of the MCMC algorithm are available at `https://figshare.com/s/d0b30e4ee58b3c9323fb`.

3 Statistical procedures

3.1 Demsar procedure

We will follow the procedure proposed by Demsar (2006). The procedure suggests one should first apply a Friedman test (which can be seen as a

non-parametric version of the repeated measure ANOVA test) to determine if there is sufficient evidence that the error rate measures for each procedure are not samples from the same distribution. If the p-value is below 0.05 (for a 95% confidence) than one can claim that it is unlikely that the error rates are all "the same" and one can proceed to compare each algorithm to the others. When comparing all algorithms among themselves, which is the case here, Demsar proposed the Nemenyi test, which will compute the p-value of all pairwise comparisons. Again, a p-value below 0.05 indicates that that comparison is statistically significant, that is, it is unlikely that two sets fo error rates are samples from the same distribution.

3.2 Bayesian comparison of multiple groups

A standard null hypothesis significant test assumes the null hypothesis, usually that the two samples came form the same population and computes the probability (p-value) of two samples from the same population having as large a difference in mean (or median) as the one encountered in the data. If the p-value is not low, one **cannot claim** that the null hypothesis is true and that the two data sets came from the same population (and thus all observed differences are due to "luck"). Failing to disprove the null hypothesis because the p-value is too high is not the same as proving the null hypothesis.

Equivalence tests are a form of "proving" a weaker version of the usual null hypothesis. Equivalence tests assume as the null hypothesis that the difference between the mean (or median) of the two sets is above a certain threshold, and if the p-value is low enough, one can claim that this null hypothesis is false, and thus that the difference between the means is smaller than the threshold. Equivalence tests are useful when that threshold is a limit of **practical irrelevance**, that is, a limit below which changes in the mean of two groups are of no practical consequence. Of course, this limit of irrelevance is very problem dependent. Section 3.3 will discuss our proposal for such a limit.

A different approach to prove that the differences are not important is to use Bayesian methods. Bayesian methods will compute the (posterior) distribution of probability for some set of measures, given the prior hypothesis on those measures. Thus a Bayesian method can compute the posterior distribution of the difference of two means. The area of the distribution of the difference that falls within the limits of irrelevance is the probability that the difference is of no practical importance. In Bayesian statistics, the limit of irrelevance is called Region of Practical Equivalence (ROPE).

The standard Bayesian ANOVA, as described in Kruschke (2014) is based on normal distributions. We are interested in a 2-factor ANOVA, where one of the factors is the classification algorithm , and the other factor is the dataset. We assume that there is no interaction component, that is, we are not interested in determining particularities of each algorithm on each dataset - we are making a claim that the datasets used in this research are a sample of real world problems, and we would like to make general statements about the algorithms.

Let us denote y_{ad} as the error rate for the algorithm a on dataset d, then the usual hierarchical model is (Kruschke, 2014):

$$
\begin{aligned}
y_{ad} &\sim \mathbf{N}(\nu_{ad}, \sigma_0) && \text{(2a)}\\
\nu_{ad} &= \beta + \alpha_a + \delta_d && \text{(2b)}\\
\sigma_0 &\sim \mathbf{U}(ySD/100, ySD * 10) && \text{(2c)}\\
\beta &\sim \mathbf{N}(yMean, ySD * 5) && \text{(2d)}\\
\alpha_a &\sim \mathbf{N}(0, \sigma_a) && \text{(2e)}\\
\delta_d &\sim \mathbf{N}(0, \sigma_d) && \text{(2f)}\\
\sigma_a &\sim \text{Gamma}(ySD/2, ySD * 2) && \text{(2g)}\\
\sigma_d &\sim \text{Gamma}(ySD/2, ySD * 2) && \text{(2h)}
\end{aligned}
$$

where $\mathbf{U}(L, H)$ is the uniform distribution with L and H as the low and high limits; $\mathbf{N}(\mu, \sigma)$ is the normal distribution with mean μ and standard deviation σ; and $\text{Gamma}(m, \sigma)$ is the Gamma distribution with mode m and standard deviation σ - note that this is not the usual parametrization of the Gamma distribution. ySD is the standard deviation of the y_{ad} data, and $yMean$, the mean of that data.

We are interested in the joint posterior probability $P(\alpha_1, \alpha_2, \ldots, \alpha_A | \{y_{ad}\})$. From that one can compute the relevant pairwise differences $P(\alpha_i - \alpha_j | \{y_{ad}\})$ and in particular, how much of the probability mass fall within the region of irrelevant differences. More specifically, if $\vec{\alpha} = \langle \alpha_1, \alpha_2, \ldots, \alpha_A \rangle$, then the simulation approach we will follow generates a set $\{\vec{\alpha}_j | \vec{\alpha}_j \sim P(\vec{\alpha} | \{y_{ad}\})\}$ for $j = 1 \ldots N$ where N is the number of chains in the MCMC simulation. We compute characteristics of distribution of $P(\alpha_i - \alpha_j | \{y_{ad}\})$ from the $\vec{\alpha}_j$.

Finally, one commonly used *robust* variation of the model above is to substitute the normal distribution in Equation 2a for a student-t distribution, with low degree of freedom, that is:

$$
\begin{aligned}
y_{ad} &\sim \mathbf{t}(\nu_{ad}, \sigma_0, df)\\
df &\sim \text{Exp}(1/30)
\end{aligned}
$$

where Exp(λ) is the exponential distribution with rate λ. We also run the robust version of the model, as discussed in E.

3.3 Threshold of irrelevance

We propose two different forms of defining the threshold of irrelevance for differences in error rates. We will compute the two thresholds and use the lowest of the two measures as the threshold of irrelevance.

The first proposal is to compare the two measures of error for each dataset and for each classification algorithm. In Equation 1 they were the two terms $\epsilon(i, 1|a, \hat{\theta}_1, i, 1)$ and $\epsilon(i, 2|a, \hat{\theta}_2, i, 2)$, that is, the error of classifier a when learning on training subset S_{2i} and tested on the subset S_{1i}, and the dual of that. The difference

$$\delta_{ai} = |\epsilon(i, 2|a, \hat{\theta}_1, i, 1) - \epsilon(i, 1|a, \hat{\theta}_2, i, 2)| \tag{3}$$

can be seen as the change on the error rate that one would expect from applying the classifier a on two different samples (S_{1i} and S_{2i}) from the same population (D_i). We will then compute the median of δ_{ai} for all datasets i and for all classification algorithms a that are among the three best (as calculated by $\epsilon(i|a)$) for each dataset. That is, we are considering as a threshold of irrelevance, the median of the change one should expect from using a good classifier (among the top three for that dataset) on two different samples of the same dataset.

The second proposal compares the estimate of the error $\epsilon(i, 2|a, \hat{\theta}, i, 1)$ computed from the 5-fold CV procedure on the subset S_{1i} which selects the best hyperparameters with the measure of the error itself. We have not made explicit the steps in the 5-fold CV, but intuitively, for each combination of hyperparameters values θ, it computes the average of the error of training in 4 folds and testing on the remaining one. Let us call it $\epsilon_{cv}(i, 1|a, \theta, i, 1)$ - that is the CV error computed within the S_{1i} subset itself. We select the combination of hyperparameters that have the lowest $\epsilon_{cv}(i, 1|a, \theta, i, 1)$. But this CV error is an estimate of the future error of the classifier (trained with that combination of hyperparameters). On the other hand $\epsilon(i, 2|a, \hat{\theta}, i, 1)$ is a "future" error of the classifier - the training set is slightly different from the 5-fold CV since it includes the whole subset S_{1i} while for each fold it included only 4/5 of S_{1i}, and the testing set S_{2i} is totally new. The difference

$$\delta^{cv}_{ai} = |\epsilon(i, 2|a, \hat{\theta}, i, 1) - \epsilon_{cv}(i, 1|a, \theta, i, 1)| \tag{4}$$

can be seen as the change in error one would expect from applying the classifier a on a slightly larger training set and testing it on completely

new data. We will compute the median of δ_{ai}^{cv} for all datasets i and for all classification algorithms a that are among the three best (as calculated by $\epsilon(i|a)$) for each dataset.

In both these proposals, we are defining the threshold of irrelevance based on a "futility" point of view, and not based on theoretical considerations. One knows that given a different sample from the same distribution or given new data to test, all algorithms will have different error rates.

3.4 Computational costs

We compute to measures of computational costs to run the algorithms. The first one is the **1-train-test**, which measures the total time to train the algorithm a on a subset S_{ji} and to test it in the subset $S_{\hat{j}i}$. Thus, that is the time to train the algorithm and to run it on two equally sized data.

But all algorithms must search for the appropriate hyperparameters. Thus also compute the total time to search for the correct hyperparameter (using, as discusses above a 5-fold CV). But different algorithms may have a different gird size of tentative hyperparameters (since as discussed for some algorithms some of range of hyperparameter may depend on characteristics of the dataset). Thus we divide the total time to search for the best hyperparameter by the number of hyperparameter combinations tested. We call it the **per hyperparameter** time.

Since the execution time varies greatly on different datasets, we will use the mean rank of each execution time to rank the algorithms, and use the Demsar procedure to determine which execution times are significantly different than the others (from a statistical sense). We have no intuition on what could be considered an irrelevant change in either execution times, so we will not perform the Bayesian ANOVA analysis.

The program ran on different cores of a cluster of different machines, so there is no guarantee that the machines have the same hardware specification. But we ran all the 14 algorithms for a particular subset on the same program. Thus the unit of distribution is all the 14 algorithms searching for the best combination of hyperparameters on a 5-fold CV of half of a dataset, then training on the full half dataset with the best selection of the hyperparameters and finally testing the classifier on the other half of the dataset.

Therefore, for the timing analysis we do not average the two measures from the subsets to obtain the measure per dataset. Instead we perform the Demsar statistical analysis using the subsets as subject indicator.

4 Results

4.1 Error rates of the different algorithms

Table 1 list the mean ranking of all algorithms, the number of times each algorithm was among the top for a dataset. The best performing algorithm, in terms of mean rank across all subsets was the random forest, followed by SVM with Gaussian kernels, followed by gradient boosting machines. The three worst performing algorithms in terms of mean rank were boosting of linear classifiers naive Bayes and L1-regularized LDA. We make no claim that these three algorithms are intrinsically "bad". It is possible that our choice of hyperparameters was outside the range of more useful values, or the implementation used was particularly not robust. In particular both `nb` and `sda` did not run at all for 20 subsets, which may explain partially their poor performance.

The ranking of the algorithm is dense, that is, all best performing algorithm receive rank 1, all second best algorithms receive rank 2, and so on. Also for the ranking, we rounded the error rates to 3 significant digits, so two algorithms have the same rank if their error rates have a difference of less than 0.0005. We do not use the rounding for the calculations of the irrelevance threshold.

alg	mean rank	count
rf	3.04	36
svmRadial	3.36	24
gbm	3.41	25
nnet	4.88	13
rknn	5.04	10
svmPoly	5.14	11
knn	5.32	11
svmLinear	6.15	13
glmnet	6.16	15
elm	6.55	5
lvq	6.96	8
sda	7.05	5
nb	8.23	7
bst	9.08	4

Table 1: The mean rank and the number of times the algorithm was among the top performer for each of the algorithms.

Figure 1 displays the heat map of the rank distribution.

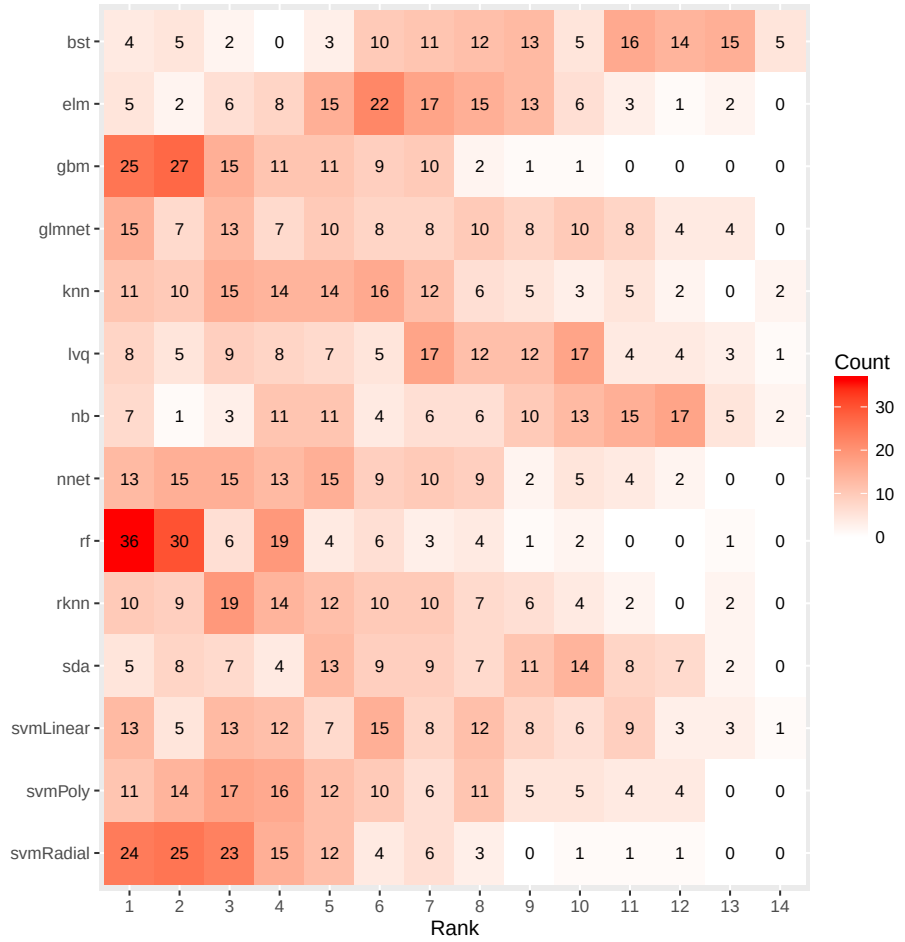

Figure 1: The heat map with the distribution of the number of times each classifier achieved a particular rank.

4.2 Demsar procedure

Table 2 lists the p-value of the pairwise comparisons for all classifiers. The p-values below 0.05 indicate that the differences between the corresponding classifiers is statistically significant.

The Demsar analysis shows that there is no statistical significance differ-

	rf	svmRadial	gbm	nnet	rknn	svmPoly	knn	svmLinear	glmnet	elm	lvq	sda	nb
svmRadial	1.00												
gbm	1.00	1.00											
nnet	**0.00**	**0.01**	**0.04**										
rknn	**0.00**	**0.00**	**0.00**	1.00									
svmPoly	**0.00**	**0.00**	**0.02**	1.00	1.00								
knn	**0.00**	**0.00**	**0.01**	1.00	1.00	1.00							
svmLinear	**0.00**	**0.00**	**0.00**	0.75	1.00	0.86	0.93						
glmnet	**0.00**	**0.00**	**0.00**	0.73	1.00	0.84	0.92	1.00					
elm	**0.00**	**0.00**	**0.00**	0.07	0.54	0.12	0.19	1.00	1.00				
lvq	**0.00**	**0.00**	**0.00**	**0.00**	0.09	**0.01**	**0.01**	0.72	0.75	1.00			
sda	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.15	0.17	0.90	1.00		
nb	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.05	0.40	0.94	
bst	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.03**	0.82

Table 2: The p-values of the Nemenyi pairwise comparison procedure. P-values below 0.05 are in bold and denote that the difference in error rate of the two corresponding classifiers is statistically significant

ence between the top 3 classification algorithms – the p-values of the pairwise comparisons among `rf`, `svmRadial`, and `gbm` are all well above 0.05. `nnet`, the next algorithm in the rank, is statistically significantly different from each of the first three. As discussed, the failure to prove that the three top algorithms are not statistically different, does not prove that they are equivalent. We will need the Bayesian ANOVA for that.

4.3 Irrelevance thresholds

The median δ_{ia} and δ_{ia}^{cv} for all datasets, and for the three best algorithms for each dataset are:

$$\text{median}(\delta_{ia}) = 0.0112$$
$$\text{median}(\delta_{ia}^{cv}) = 0.0134$$

Thus, the two measures are comparable, a little over 1%, and we will use the lower among the two as out threshold of irrelevance (0.0112). The fact that the median of δ_{ia} is smaller than the other is somewhat surprising, since δ_{ia} measures the error of learning (and testing) with two different samples of the data population, while δ_{ia}^{cv} is the difference of learning on basically the same subset, but testing in two different samples of different sizes from the same data population.

4.4 Bayesian ANOVA analysis

Table 3 displays the pairwise probability that the differences of the classifiers' error rates is within our limits of irrelevance (from -0.0112 o 0.0112).

The comparison between the top three algorithms shows that the differences are with high probability (0.83 to 0.64) within our range of irrelevance,

	rf	svmRadial	gbm	nnet	rknn	svmPoly	knn	svmLinear	glmnet	elm	lvq	sda	nb
svmRadial	0.83												
gbm	0.74	0.64											
nnet	0.08	0.04	0.26										
rknn	0.01	0.00	0.04	0.62									
svmPoly	0.44	0.32	0.72	0.56	0.17								
knn	0.00	0.00	0.01	0.42	0.80	0.07							
svmLinear	0.00	0.00	0.00	0.20	0.61	0.02	0.78						
glmnet	0.00	0.00	0.00	0.02	0.20	0.00	0.36	0.61					
elm	0.00	0.00	0.00	0.06	0.33	0.00	0.52	0.74	0.82				
lvq	0.00	0.00	0.00	0.16	0.55	0.01	0.73	0.85	0.67	0.79			
sda	0.00	0.00	0.00	0.01	0.12	0.00	0.24	0.47	0.82	0.73	0.53		
nb	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.03	
bst	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 3: The probability that the difference between the error rate is within the limits of irrelevance (from -0.0112 to 0.0112) for all pairs of algorithms

that is, there is no "practical difference" in the error rates of the three algorithms. In particular there is a stronger argument to be made that the best algorithm, `rf` is equivalent for practical purposed to the `svmRadial`. The claim of equivalence between `gbm` and the other two is less strong, in particular, to our surprise, in relation to the second best `svmRadial`.

4.5 Computational costs of the algorithms

alg	mean rank 1-t-t	alg	mean rank p/h
svmLinear	2.67	knn	1.72
elm	2.68	lvq	4.30
bst	3.57	**svmRadial**	4.43
knn	3.72	sda	4.56
sda	4.21	glmnet	4.96
glmnet	6.17	svmPoly	6.03
svmPoly	7.16	**gbm**	6.65
lvq	7.17	nnet	7.88
svmRadial	7.73	nb	9.06
gbm	8.88	**rf**	9.83
nnet	9.54	rknn	10.70
rf	10.94	svmLinear	11.25
rknn	11.13	elm	11.67
nb	12.19	bst	11.94

Table 4: The mean rank for the 1-train-test measure (column 2: 1-t-t) and the per hyperparameter measure (column 4: p/h)

Table 4 shows the mean rank for the 1-train-test and per hyperparameter

times. The top 6 algorithms in term of accuracy are indeed computationally costly in terms of training and testing, and they correspond to the 7 worse 1-train-test times. The result for the `nb` is surprising. Also surprising, to the author at least, is how more costly is `rf` in comparison to the two other "equivalent" classifiers, `svmRadial` and `gbm`. Table 5 is the Demsar procedure p-values for the pairwise comparisons of the 1-train-test cost of the top 6 algorithms (B contains the full p-value table). Notice that the difference between `svmRadial` and `gbm` is not significant; the difference between `rf` and `svmRadial` and `gbm` are significant.

Table 6 lists the p-values of the pairwise comparisons of the top 6 algorithms. B contains the full p-value table. `svmRadial` is significantly faster than all other algorithm in the per hyperparameter measure.

	rf	svmRadial	gbm	nnet	rknn
svmRadial	**0.00**				
gbm	**0.01**	0.69			
nnet	0.37	**0.06**	1.00		
rknn	1.00	**0.00**	**0.00**	0.13	
svmPoly	**0.00**	1.00	0.08	**0.00**	**0.00**

Table 5: The pairwise comparison p-value table for the top 6 algorithm of the 1-train-test computational cost of the algorithms.

	rf	svmRadial	gbm	nnet	rknn
svmRadial	**0.00**				
gbm	**0.00**	**0.00**			
nnet	**0.03**	**0.00**	0.61		
rknn	0.94	**0.00**	**0.00**	**0.00**	
svmPoly	**0.00**	0.17	1.00	**0.05**	**0.00**

Table 6: The pairwise comparison p-value table for the top 6 algorithm of the per hyperparameter cost of the algorithms.

The per hyperparameter cost is surprising low for the `svmRadial`, which is a welcome result since the hyperparameter grid for the `svmRadial` does not depend on characteristics of the dataset, and therefore one will need to test many combinations, regardless of the dataset.

On the other hand the large per hyperparameter cost of `rf` could be a problem if one cannot use the free hyperparameter "trick" we described.

Not all implementations of `rf` allow access to each tree decision[1], and if one has to explicitly train for different number of trees, the total number of hyperparameters tested times the high per hyperparameter cost may make the random forest solution less viable.

5 Discussion

An important part of these results is the definition of an irrelevance threshold, a difference in error rates between to algorithms which is likely to be irrelevant from practical purposes. We claim that 0.0112 is median change of error rate one should expect when a classifier is trained and tested on a fully different set of data from the same population. This is a very strong claim, for which we have no theoretical backing, but which is empirically derived from the 115 datasets used in this research. And if one agrees that these datasets are a random sample of real life datasets that a practitioner will "find in the wild" (more on this issue below), that one must accept that this conclusion is likely generalizable for all data.

In general terms, one should not loose too much sleep trying to improve a classifier by less than 1% since it is likely that one will loose (or gain) that much when new data is used.

We can provide some independent evidence that the change in error rate with new data is around 1%. Kaggle competitions are evaluated against two hold out test sets, the public and the private. Usually the public dataset is around 25 to 30% of the private dataset (so the private dataset is not totally new data, but 70 to 75% new data. Thus the difference of private and public accuracy scores are an independent lower bound estimate of the change in accuracy for new data. Unfortunately very few of the Kaggle competitions are measured using accuracy, but one of them the "cats and dogs[2]" did. The median of the difference between the public and private test scores, for the top 50 entries on the public score is 0.0068, which is 60% of our threshold of irrelevance.

If one is reluctant to accept our 1% as a limit of irrelevance when comparing classifiers, Table 8 shows the probability that the difference between the accuracy rate is below 0.0056, that is half of our irrelevance threshold, for the six best algorithms. Even with this extremely rigorous limit of irrelevance, there is a some probability that `rf`, `svmRadial`, and `gbm` would

[1] The `predict` method of random forest in sklearn, for example, does not allow access to each three individual decision.

[2] `https://www.kaggle.com/c/dogs-vs-cats`

perform at similar levels. In particular, there is still a 50% probability that `rf` and `svmRadial` will be equivalent for practical purposes. .

	rf	svmRadial	gbm	nnet	rknn
svmRadial	0.51				
gbm	0.41	0.33			
nnet	0.02	0.01	0.08		
rknn	0.00	0.00	0.01	0.31	
svmPoly	0.18	0.11	0.40	0.26	0.05

Table 7: Probability that the difference in error rates is within -0.0056 and 0.0056, for the six best ranking algorithms.

Another important point of discussion is the Bayesian analysis itself. We followed a more traditional modeling of the Bayesian ANOVA, where the priors for the means follow a Gaussian distribution (Equations 2e and 2f). But error rates does not follow a Gaussian distribution, first because they are limited to a 0-1 range. Second, the whole point of the Demsar procedure to compare classifiers on different datasets is to use a non-parametric test, one that does not rely on the assumption of normality of the data. We are not making an assumption that error rates are normally distributed, but we are making an assumption that the priors for the coefficients of the dependency on the dataset (given an algorithm) and on the algorithm (given a dataset) are drawn from a Gaussian distribution. C discusses that those assumptions are somewhat reasonable, but in the future, the community should probably define a better model for the Bayesian analysis of error rates. D shows that there was convergence for the Monte Carlo simulations that generated the results in Table 3. Finally, we also ran the robust form of the model. Table 8 shows the probability that the differences between the top six algorithms are within the limit of practical relevance. Notice that one can be 100% sure of the equivalence of the top three algorithms using the robust model. Also notice that even `nnet` which was significantly different from the top three, is with high probability equivalent to at least `svmRadial` and `gbm`. There is no contradiction in both statements: the NHST claims that the difference is "real" while the Bayesian ANOVA claims that that difference (although real) it is not important. But nevertheless, the robust model seems to make much stronger claims of equivalence than the non-robust model, and as a precaution, we preferred the latter model. The stronger results of the robust model can be explained by shrinkage (Kruschke, 2014); since it allows more outliers, the mean of each distribution of the algorithms' coefficient would

be "free" to move closer together, and thus the higher probabilities that the differences are below the threshold of irrelevance.

	rf	svmRadial	gbm	nnet	rknn
svmRadial	1.00				
gbm	1.00	1.00			
nnet	0.64	0.93	0.95		
rknn	0.74	0.96	0.97	1.00	
svmPoly	0.50	0.85	0.88	1.00	1.00

Table 8: Probability that the difference in error rates is within -0.0112 and 0.0112, for the six best ranking algorithms using the robust Bayesian model.

E displays the full table for the pairwise probabilities that the differences between the algorithms are within the limit of irrelevance, and also discusses the convergency and model checking of the robust model.

Our results are in general compatible with those in Fernández-Delgado et al. (2014). Random forest is the best ranking algorithm in both experiments; gradient boosting machines which was not included in Fernández-Delgado et al. (2014) performs well, as reported in various blogs, RBF SVM also performs well. The divergence starts with the next best classifications algorithms: Fernández-Delgado et al. (2014) lists polynomial kernel SVM `svmPoly` and extreme learning machines `elm` as the next best families, but in our case `svmPoly` was equivalent to 1-hidden layer neural nets `nnet`, and a bagging of knn `rknn`. The differences between these algorithms was also below our practical limit of relevance. `elm` did perform worse than `nnet` and `rknn`.

5.1 Limits on this research

One limit of this research is that its conclusions should only be generalized to datasets "similar" to the ones used. In particular, our datasets did not include very large, or very sparse, or datasets with much more features than datapoints (as it is common in some bioinformatics applications). Furthermore, our experiments were performed on binary classification tasks. Given these restrictions on the data, if one can assume that the datasets in UCI repository are samples of "real life" problems, our results should be generalizable.

A second limit of this research is based on our decisions regarding the hyperparameter search for each algorithm. There is very little research on the range of possible or useful values of hyperparameters for any of the

algorithms discussed, so our decisions are debatable. And if our choices were particularly bad, an algorithm may have been unfairly ranked in the comparisons.

The reader should be aware of the limited usefulness for the timing results. We used "standard" implementations of the algorithms in R. All three implementations were written in some compiled language and linked to R, but it may be that only libSvm, which is the base of the SVM R implementation, has been under current development, and the execution time of the SVM may be due to that. There has been more recent implementations of random forest (Wright and Ziegler, 2015) and gradient boosting machines (Distributed (Deep) Machine Learning Community, 2016) which claim both a faster execution time and shorter memory footprint. On the other hand, incremental and online solvers for SVM (Bordes et al., 2005; Shalev-Shwartz et al., 2011) may further tip the scale in favor of SVM.

5.2 Future research

Given the result that `rf`, `svmRadial`, and `gbm` are likely the best performing algorithms, it is interesting to explore further variations and different implementations of these algorithms. We discussed that more modern implementations the algorithms may alter the timing results. But variations of the algorithms may also alter the ranking and the apparent equivalence of the three algorithms. Variations of Random Forest, such as Rotation Forest (Rodriguez et al., 2006), Extremely Randomized Forest (Geurts et al., 2006), random forest of conditional inference trees (Hothorn et al., 2006; Strobl et al., 2007), should be compared with the standard algorithm. Least square SVM (Suykens and Vandewalle, 1999) should be compared with the standard SVM, and different models of boosting such as AdaBoost, LogiBoost, BrownBoost, should be compared with `gbm`.

There has been a large number of published research on different methods and algorithms for hyperparameter selection in RBF SVM, but almost no research in hyperparameter selection for Random Forests and Gradient Boosting Machines. Hyperparameter selection is a very important and computationally expensive step in selecting an algorithm for a particular problem, and further understanding of how to improve this process is needed, specially for those two algorithms.

6 Conclusion

We have shown that random forests, RBF SVM, and gradient boosting machines are classification algorithm that most likely will result in the highest accuracy and that it is likely that there will be no important difference in error rate among these three algorithms, specially between random forest and RBF SVM. In terms of training and testing execution times, SVM with a RBF kernel is faster then the two other competitors.

We believe that this paper also makes important methodological contributions for machine learning research, mainly in the definition of a threshold of irrelevance, below which, changes in error rate should be considered as of no practical significance. We argued that this threshold should be 0.0112. Another important methodological contribution is the use of a Bayesian Analysis of Variance method to verify that the differences among the three top performing algorithms, was very likely smaller than the 0.0112 threshold of practical irrelevance, and we showed that the Bayesian model used is appropriate to be used in comparisons of error rates among different algorithms.

References

Bordes, A., Ertekin, S., Weston, J., Bottou, L., 2005. Fast kernel classifiers with online and active learning. The Journal of Machine Learning Research 6, 1579–1619.

Brooks, S., Gelman, A., 1998. General methods for monitoring convergence of iterative simulations. Journal of Computational and Graphical Statistics 7 (4), 434–455.

Clemmensen, L., 2012. sparseLDA: Sparse Discriminant Analysis. R package version 0.1-6.
URL `http://CRAN.R-project.org/package=sparseLDA`

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

Dietterich, T., Bakiri, G., 1995. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research, 263–286.

Distributed (Deep) Machine Learning Community, 2016. xgboost: Scalable, Portable and Distributed Gradient Boosting (GBDT, GBRT or GBM)

Library, for Python, R, Java, Scala, C++ and more. https://github.com/dmlc/xgboost.

Engel, J., 1988. Polytomous logistic regression. Statistica Neerlandica 42 (4), 233–252.

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, 3133–3181.

Franc, V., et al., 2002. Multi-class support vector machine. In: 16th International Conference on Pattern Recognition. Vol. 2. pp. 236–239.

Friedman, J., Hastie, T., Tibshirani, R., 2010. Regularization paths for generalized linear models via coordinate descent. Journal of statistical software 33 (1), 1.

Friedman, J. H., 2001. Greedy function approximation: a gradient boosting machine. Annals of Statistics, 1189–1232.

Gelman, A., Meng, X.-L., Stern, H., 1996. Posterior predictive assessment of model fitness via realized discrepancies. Statistica Sinica, 733–760.

Geurts, P., Ernst, D., Wehenkel, L., 2006. Extremely randomized trees. Machine Learning 63 (1), 3–42.

Gosso, A., 2012. elmNN: Implementation of ELM (Extreme Learning Machine) algorithm for SLFN (Single Hidden Layer Feedforward Neural Networks). R package version 1.0.
URL http://CRAN.R-project.org/package=elmNN

Hastie, T., Rosset, S., Tibshirani, R., Zhu, J., 2004. The entire regularization path for the support vector machine. The Journal of Machine Learning Research 5, 1391–1415.

Hastie, T., Tibshirani, R., et al., 1998. Classification by pairwise coupling. The Annals of Statistics 26 (2), 451–471.

Hothorn, T., Hornik, K., Zeileis, A., 2006. Unbiased recursive partitioning: A conditional inference framework. Journal of Computational and Graphical Statistics 15 (3), 651–674.

Huang, G.-B., Zhu, Q.-Y., Siew, C.-K., 2006. Extreme learning machine: theory and applications. Neurocomputing 70 (1), 489–501.

Kohonen, T., 1995. Learning Vector Quantization. Springer.

Kruschke, J., 2014. Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan. Academic Press.

Li, S., 2015. rknn: Random KNN Classification and Regression. R package version 1.2-1.
URL https://cran.r-project.org/web/packages/rknn/index.html

Liaw, A., Wiener, M., 2002. Classification and regression by randomforest. R News 2 (3), 18–22.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., 2014. e1071: Misc Functions of the Department of Statistics (e1071), TU Wien. R package version 1.6-4.
URL http://CRAN.R-project.org/package=e1071

Ridgeway, G., 2013. gbm: Generalized Boosted Regression Models. R package version 2.1.
URL http://CRAN.R-project.org/package=gbm

Rodriguez, J. J., Kuncheva, L. I., Alonso, C. J., 2006. Rotation forest: A new classifier ensemble method. IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (10), 1619–1630.

Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A., 2011. Pegasos: Primal estimated sub-gradient solver for SVM. Mathematical Programming 127 (1), 3–30.

Strobl, C., Boulesteix, A.-L., Zeileis, A., Hothorn, T., 2007. Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC Bioinformatics 8 (25).

Suykens, J. A., Vandewalle, J., 1999. Least squares support vector machine classifiers. Neural Processing Letters 9 (3), 293–300.

Venables, W. N., Ripley, B. D., 2002. Modern Applied Statistics with S, 4th Edition. Springer, New York, iSBN 0-387-95457-0.
URL http://www.stats.ox.ac.uk/pub/MASS4

Wang, Z., 2014. bst: Gradient Boosting. R package version 0.3-4.
URL http://CRAN.R-project.org/package=bst

Weihs, C., Ligges, U., Luebke, K., Raabe, N., 2005. klar analyzing german business cycles. In: Baier, D., Decker, R., Schmidt-Thieme, L. (Eds.), Data Analysis and Decision Support. Springer-Verlag, Berlin, pp. 335–343.

Wolpert, D. H., 1992. Stacked generalization. Neural Networks 5 (2), 241–259.

Wright, M. N., Ziegler, A., 2015. ranger: A fast implementation of random forests for high dimensional data in C++ and R. arXiv:1508.04409.

Zajac, Z., 2016. What is better: gradient-boosted trees, or a random forest? `http://fastml.com/what-is-better-gradient-boosted-trees-or-random-forest/`.

Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B 67 (2), 301–320.

A Datasets

The table below list the characteristics of all datasets, ordered by size. The name of the dataset is the same as the ones used in Fernández-Delgado et al. (2014). The size refers to one half of the dataset. The *nfeat* column is the number of features of the datasets; *ndat* the number of data; *prop* the proportion of data of the positive class. *Notes* has the following values:

- *m* the dataset was multivalued, and so was converted to a binary problem using the procedure discussed in section 2
- *l* large dataset. Only 5000 of the data was used to search fror the hyperparameters (see section 2).

Table 9: The datasets

dataset	nfeat	ndat	prop	note
fertility	10	50	0.96	
zoo	17	50	0.62	m
pittsburg-bridges-REL-L	8	51	0.82	m
pittsburg-bridges-T-OR-D	8	51	0.86	
pittsburg-bridges-TYPE	8	52	0.62	m

continued in the next page

Table 9: The datasets

dataset	nfeat	ndat	prop	note
breast-tissue	10	53	0.53	m
molec-biol-promoter	58	53	0.49	
pittsburg-bridges-MATERIAL	8	53	0.94	m
acute-inflammation	7	60	0.52	
acute-nephritis	7	60	0.67	
heart-switzerland	13	61	0.38	m
echocardiogram	11	65	0.74	
lymphography	19	74	0.46	m
iris	5	75	0.72	m
teaching	6	75	0.64	m
hepatitis	20	77	0.22	
hayes-roth	4	80	0.57	m
wine	14	89	0.60	m
planning	13	91	0.74	
flags	29	97	0.47	m
parkinsons	23	97	0.25	
audiology-std	60	98	0.64	m
breast-cancer-wisc-prog	34	99	0.77	
heart-va	13	100	0.52	m
conn-bench-sonar-mines-rocks	61	104	0.54	
seeds	8	105	0.64	m
glass	10	107	0.40	m
spect	23	132	0.67	m
spectf	45	133	0.19	
statlog-heart	14	135	0.53	
breast-cancer	10	143	0.69	
heart-hungarian	13	147	0.63	
heart-cleveland	14	151	0.75	m
haberman-survival	4	153	0.75	
vertebral-column-2clases	7	155	0.68	
vertebral-column-3clases	7	155	0.67	m
primary-tumor	18	165	0.68	m
ecoli	8	168	0.64	m
ionosphere	34	175	0.30	
libras	91	180	0.51	m
dermatology	35	183	0.64	m

continued in the next page

Table 9: The datasets

dataset	nfeat	ndat	prop	note
horse-colic	26	184	0.64	
congressional-voting	17	217	0.59	
arrhythmia	263	226	0.67	m
musk-1	167	238	0.58	
cylinder-bands	36	256	0.38	
low-res-spect	101	265	0.25	m
monks-3	7	277	0.48	
monks-1	7	278	0.51	
breast-cancer-wisc-diag	31	284	0.63	
ilpd-indian-liver	10	291	0.72	
monks-2	7	300	0.64	
synthetic-control	61	300	0.51	m
balance-scale	5	312	0.53	m
soybean	36	341	0.42	m
credit-approval	16	345	0.43	
statlog-australian-credit	15	345	0.32	
breast-cancer-wisc	10	349	0.66	
blood	5	374	0.76	
energy-y1	9	384	0.80	m
energy-y2	9	384	0.75	m
pima	9	384	0.66	
statlog-vehicle	19	423	0.52	m
annealing	32	449	0.81	m
oocytes_trisopterus_nucleus_2f	26	456	0.41	
oocytes_trisopterus_states_5b	33	456	0.98	m
tic-tac-toe	10	479	0.34	
mammographic	6	480	0.56	
conn-bench-vowel-deterding	12	495	0.53	m
led-display	8	500	0.51	m
statlog-german-credit	25	500	0.72	
oocytes_merluccius_nucleus_4d	42	511	0.31	
oocytes_merluccius_states_2f	26	511	0.93	m
hill-valley	101	606	0.49	
contrac	10	736	0.79	m
yeast	9	742	0.55	m
semeion	257	796	0.50	m

continued in the next page

Table 9: The datasets

dataset	nfeat	ndat	prop	note
plant-texture	65	799	0.50	m
wine-quality-red	12	799	0.56	m
plant-margin	65	800	0.52	m
plant-shape	65	800	0.52	m
car	7	864	0.28	m
steel-plates	28	970	0.66	m
cardiotocography-10clases	22	1063	0.39	m
cardiotocography-3clases	22	1063	0.86	m
titanic	4	1100	0.66	
image-segmentation	19	1155	0.58	m
statlog-image	19	1155	0.55	m
ozone	73	1268	0.97	
molec-biol-splice	61	1595	0.75	m
chess-krvkp	37	1598	0.48	
abalone	9	2088	0.68	m
bank	17	2260	0.88	
spambase	58	2300	0.61	
wine-quality-white	12	2449	0.48	m
waveform-noise	41	2500	0.66	m
waveform	22	2500	0.68	m
wall-following	25	2728	0.78	m
page-blocks	11	2736	0.92	m
optical	63	2810	0.51	m
statlog-landsat	37	3217	0.56	
musk-2	167	3299	0.85	m
thyroid	22	3600	0.94	m
ringnorm	21	3700	0.49	
twonorm	21	3700	0.49	
mushroom	22	4062	0.51	
pendigits	17	5496	0.51	ml
nursery	9	6480	0.68	ml
magic	11	9510	0.65	l
letter	17	10000	0.50	ml
chess-krvk	7	14028	0.53	ml
adult	15	24421	0.76	l
statlog-shuttle	10	29000	0.84	ml

continued in the next page

Table 9: The datasets

dataset	nfeat	ndat	prop	note
connect-4	43	33778	0.75	l
miniboone	51	65032	0.28	l

B Full p-value tables for the pairwise comparison of the 1-train-test and per hyperparameter costs

	rf	svmRadial	gbm	nnet	rknn	svmPoly	knn	svmLinear	glmnet	elm	lvq	sda	nb
svmRadial	0.00												
gbm	0.01	0.69											
nnet	0.37	0.06	1.00										
rknn	1.00	0.00	0.00	0.13									
svmPoly	0.00	1.00	0.08	0.00	0.00								
knn	0.00	0.00	0.00	0.00	0.00	0.00							
svmLinear	0.00	0.00	0.00	0.00	0.00	0.00	0.63						
glmnet	0.00	0.20	0.00	0.00	0.00	0.90	0.00	0.00					
elm	0.00	0.00	0.00	0.00	0.00	0.00	0.66	1.00	0.00				
lvq	0.00	1.00	0.08	0.00	0.00	1.00	0.00	0.00	0.89	0.00			
sda	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.07	0.01	0.08	0.00		
nb	0.60	0.00	0.00	0.00	0.89	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
bst	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.91	0.00	0.92	0.00	0.96	0.00

Table 10: The full p-value table for the 1-train-test time

	rf	svmRadial	gbm	nnet	rknn	svmPoly	knn	svmLinear	glmnet	elm	lvq	sda	nb
svmRadial	0.00												
gbm	0.00	0.00											
nnet	0.03	0.00	0.61										
rknn	0.94	0.00	0.00	0.00									
svmPoly	0.00	0.17	1.00	0.05	0.00								
knn	0.00	0.00	0.00	0.00	0.00	0.00							
svmLinear	0.35	0.00	0.00	0.00	1.00	0.00	0.00						
glmnet	0.00	1.00	0.11	0.00	0.00	0.80	0.00	0.00					
elm	0.05	0.00	0.00	0.00	0.92	0.00	0.00	1.00	0.00				
lvq	0.00	1.00	0.00	0.00	0.00	0.09	0.00	0.00	1.00	0.00			
sda	0.00	1.00	0.01	0.00	0.00	0.29	0.00	0.00	1.00	0.00	1.00		
nb	0.98	0.00	0.00	0.67	0.13	0.00	0.00	0.01	0.00	0.00	0.00	0.00	
bst	0.01	0.00	0.00	0.00	0.64	0.00	0.00	0.99	0.00	1.00	0.00	0.00	0.00

Table 11: The full p-value table for the per hyperparameter times

C Bayesian model verification

Figure 2 plots the histogram of the error rates across the 115 datasets for the different algoritms, superimposed with the best fit Gaussian. One can

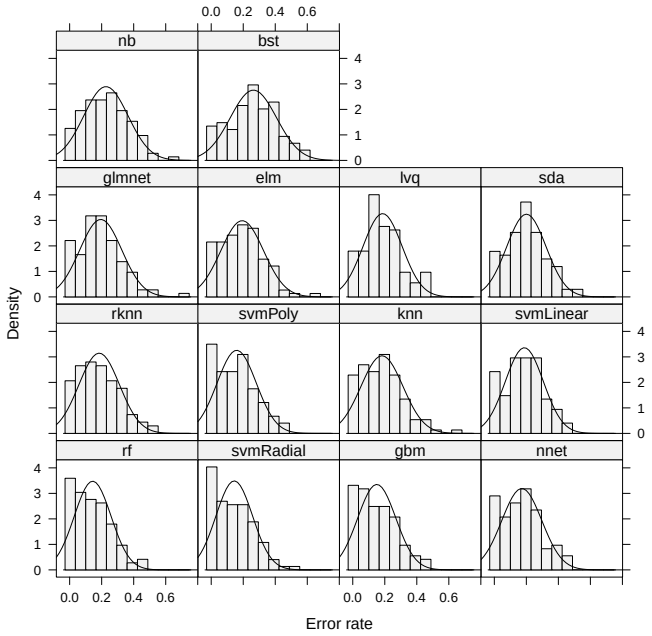

Figure 2: Histogram of error rates for each algorithm across the different datasets, and the best fit Gaussian distribution.

argue that the distribution of error rates can reasonably be considered as normal, which would match the assumptions of the equation 2e. Similarly, Figure 3 is the distribution of error rates for 20 random datasets, and again one argue that the assumption in equation 2f is reasonable.

A more formal model verification is the procedure of posterior predictive check proposed by Gelman et al. (1996). The procedure calculates how unlikely is the true data when compared to data generated by the model, given the posterior distribution of the hyperparameters of the model. The data (true and generated) are summarized by the χ^2 discrepancy (Gelman et al., 1996).

Figure 4 shows the relation between the discrepancies of the true data and that of the generated data, for 1667 of the points generated by the MCMC algorithm, following the posterior of the parameters of the model. The probability of that the real data (or data with even more extreme discrepancies) were generated by the model is the proportion of the points above the $x = y$ line. As one can see, that probability is around 0.5, and

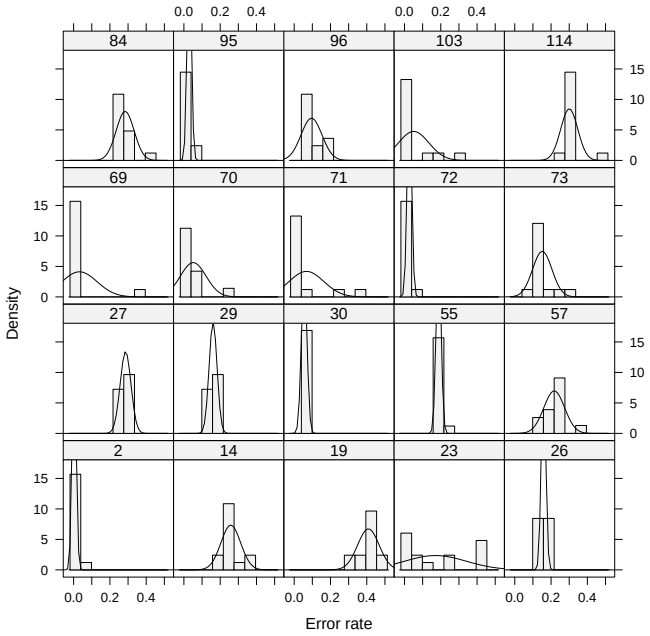

Figure 3: Histogram of error rates for 20 random datasets across the different algorithms, and the best fit Gaussian distribution.

the real data is clearly among the most likely of the data generated by the model. Thus, under this criterion, the model is appropriate. But we must also point out that 3% of the data generated was negative, which is at least esthetically unpleasant.

D Convergence of the MCMC

There was no attempts to optimize the number of simulations of the MCMC algorithm. We used JAGS, with 5000 burnin steps and 5000 adaptative steps. Finally we ran 100000 total interactions on 4 separated chains.

Below is the Gelman and Rubin diagnostic (Brooks and Gelman, 1998) which compares the variance within and between chains, as reported by the function `gelman.diag` from the R package `coda`, where the variables saved are the ones in Equation 2b: `b0` is α and `b1[a]` are the β_a for each algorithm, `b2[d]` are the δ_d for each dataset and `ySigma` is σ_0 from Equation 2a, `a1SD` and `a2SD` are σ_a and σ_d from 2g and 2h. Not all lines that refer to the `b2[d]`

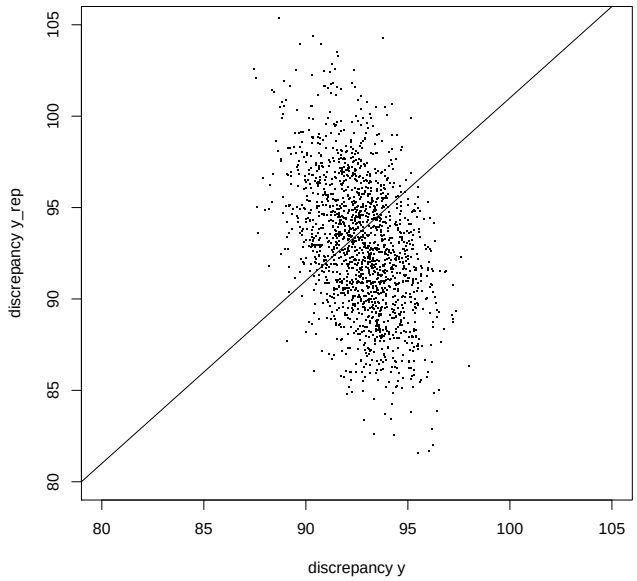

Figure 4: Discrepancy (χ^2) of the generated data against the discrepancy of the true data, for 1667 data from the MCMC chains

variables are shown, but they all have the same values.

```
Potential scale reduction factors:

        Point est. Upper C.I.
b0               1          1
b1[1]            1          1
b1[2]            1          1
b1[3]            1          1
b1[4]            1          1
b1[5]            1          1
b1[6]            1          1
b1[7]            1          1
b1[8]            1          1
b1[9]            1          1
b1[10]           1          1
```

```
b1[11]           1          1
b1[12]           1          1
b1[13]           1          1
b1[14]           1          1
b2[1]            1          1
b2[2]            1          1
b2[3]            1          1
...
b2[113]          1          1
b2[114]          1          1
b2[115]          1          1
ySigma           1          1
a1SD             1          1
a2SD             1          1

Multivariate psrf

1
```

Values between 1 and 1.1 suggest convergence of the interactions.

The effective sizes of the interactions shows that there is no problem of high autocorrelation. Again not all values for the `b2[d]` variables are shown.

```
       b0     b1[1]     b1[2]     b1[3]     b1[4]     b1[5]     b1[6]     b1[7]
100000.00 101017.13  98356.57  99052.19  99172.48  99820.25  99350.88 100262.05
    b1[8]     b1[9]    b1[10]    b1[11]    b1[12]    b1[13]    b1[14]     b2[1]
100541.69  98992.43  98427.81 100000.00  97006.48  97467.65  94816.02 100000.00
    b2[2]     b2[3]     b2[4]     b2[5]     b2[6]     b2[7]     b2[8]     b2[9]
 99043.72  97966.91  99380.08 100000.00  99530.18 101053.74 100087.01 100000.00
 ...
  b2[114]   b2[115]    ySigma      a1SD      a2SD
100204.87 100216.75  55353.48  16108.44  51977.11
```

E Results with the robust Bayesian model

This section displays the results of the Bayesian analysis based on the robust model. Table 12 is the full probability table from the Bayesian ANOVA.

We cannot perform the verification of the model using posterior predictive check because the χ^2 discrepancy needs the variance of the data. Under the robust model, the variance of the data depends also on the degrees of

	rf	svmRadial	gbm	nnet	rknn	svmPoly	knn	svmLinear	glmnet	elm	lvq	sda	nb
svmRadial	1.00												
gbm	1.00	1.00											
nnet	0.64	0.93	0.95										
rknn	0.74	0.96	0.97	1.00									
svmPoly	0.50	0.85	0.88	1.00	1.00								
knn	0.45	0.83	0.87	1.00	1.00	1.00							
svmLinear	0.07	0.32	0.38	0.99	0.98	0.99	1.00						
glmnet	0.12	0.43	0.49	0.99	0.98	1.00	1.00	1.00					
elm	0.00	0.01	0.01	0.61	0.49	0.73	0.78	0.98	0.96				
lvq	0.00	0.00	0.01	0.53	0.41	0.66	0.71	0.97	0.94	1.00			
sda	0.00	0.01	0.01	0.54	0.42	0.66	0.71	0.97	0.94	1.00	1.00		
nb	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.03	0.39	0.45	0.44	
bst	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.52

Table 12: The probability that the difference between the error rate is within the limits of irrelevance (from -0.0112 to 0.0112) for all pairs of algorithms using the robust model.

freedom of the student-t distribution, and in the case of the robust simulations is 1.12, and the variance of the student-t distribution is not defined for degrees of freedom below 2.