CSCI447: Analysis of Traditional Machine-Learning Algorithms

Christopher R. Barbour, Brandon Fenton, John Sherrill

September 14, 2015

Abstract

Ten machine learning algorithms were assigned for performance analysis on 10 user specified datasets obtained from the UCI Machine Learning Database. First, the experimental method is presented and then the algorithms are breifly described. Hypotheses for classifier performance did not single out one algorithm as an across-the-board better classifier. Results are presented for the experiments. As generally hypothesized, no classifier consistently outperformed the rest.

1 Introduction

The purpose of this study is to assess the classification performance of 10 machine-learning algorithms on 10 datasets from the UCI Machine Learning Repository [8]. The algorithms, described in Section 2.2, cover a broad spectrum of classification techniques with varying degrees of computational intensity and inductive bias. The latter of which will be the focus of the performance hypothesis for each dataset and of the interpretation of results.

2 Experimental Design

For each of the selected datasets and algorithms, we will randomly partition the examples into a training dataset which will be used to construct the classifier and a test dataset which will be used to test the classifier's ability to generalize, i.e. correctly classify similar examples from the same population of interest. Each algorithm will be heuristically tuned to each dataset prior to the experiment using a single run 10-fold cross-validation scheme.

Three measures were chosen to assess different capabilities of each algorithm:

1. weighted average F-measure: The F-measure for a given class is a function of the precision and recall of a test (the harmonic mean of the precision and recall). The weighted average F-measure is the sum of the F-measures for all the classes in the data weighted by the number of instances in each class.
2. one minus the misclassification rate: Misclassification rates measure the proportion of incorrectly classified observations, so one minus this quantity represents the proportion of correctly classified observations.
3. weighted average Area Under the ROC curve: For a given class, area under the ROC (AUROC) assesses the classifier's sensitivity (probability of predicting the correct class, also referred to as precision) and specificity (probability of not predicting the incorrect class). The weighted AUROC is the sum of the AUROCs over the classes in the data weighted by the number of instances in each class.

2.1 Datasets

Ten datasets were obtained from the UCI Machine Learning Repository. Features found to have unique values for each instance (e.g. identification numbers or the name of an animal) were removed.

2.2 Algorithm Descriptions

2.2.1 Instance-Based Methods (IB1 and IBk)

The nearest neighbor algorithm [1] classifies instances by way of referencing a set X of pre-classified instances. For a given instance x of unknown class, the algorithm searches X for an element y that most closely resembles x in attribute. Various metrics may be used for defining "resemblance" although Euclidean distance is most often used. The class of x is then deteremined to be the class of y.

The k-Nearest Neighbor (k-NN) classifier is a generalization of the simple nearest neighbor algorithm. For instance x to be classified, the k-NN algorithm searches a reference set X for the $k \in \mathbb{N}$ elements most closely resembling x. The class of x is assigned as be the most frequent class of the k elements inspected. Modifications to this algorithm include weighting the neighbors based on proximity to the instance being classified, as well as a Parzen windows implementation. We chose to tune this algorithm in the cross-validation tuning procedure over the possible combinations of $k \in \{3, \ldots, 8\}$ and whether to use no distance weighting or the reciprocal of distance as the weighting. We did not utilize the Parzen windows option.

The inductive bias of these instance-based methods lies in the assumption that observations that are close to one another are similar in there class assignment.

2.2.2 Naive Bayes

Naive Bayes estimation [6] assumes conditional independence of the covariates conditioned on the class and applies Bayes rule to get the maximum class probability conditioned on the covariates.

For categorical predictors, we can simply estimate the probability of X conditioned on C is simply the number of examples that have both predictor X and class C divided by the total examples in class C. However, this can result in a probability of 0 if there are no examples in class C with covariate Xi, so we typically assume a dirichlet prior on the distribution on the classes and produce a maximum a posterior (MAP) estimator, which effectively "smooth" these probabilities away from 0. The estimated probability of class C is simply the total number of examples in class C divided by the total number of examples in the data. For continuous input, we can either assume that X_i conditioned on class C_i is a Gaussian random variable and use probability theory to estimate the conditional mean and variance, or we can use a kernel density estimation instead. This reference show that kernel estimation can improve prediction performance when covariates are not conditionally Gaussian and does not hinder performance much when they are. Therefore, we will use kernel density estimation in our computations involving continuous predictors.

The inductive bias of naive bayes estimators lies in the assumptions that 1) the classes are linearly separable and that 2) preditors are conditional independent which is often violated in practice.

2.2.3 Logistic Regression with Regularization

Logistic regression [5], [7] is a form of generalized linear modeling, where a logit link function is used and a binomial (or more generally, a multinomial) likelihood is assumed. The parameters of logistic regression are estimated using a gradient ascent search method to maximize the conditional log-likelihood from the data. Since this is a gradient search method, it is possible to get stuck at a local maximum instead of the global maximum. This algorithm assumes linear separation of the groups, as well as independence of the covariates.

One can choose to maximize a function of the log-likelihood, where a pentaly is added. The penalty corresponds to a constant multiplied by the squared L_2 norm of the model parameters. This corresponds to a "ridge" penalty and will have the effect of shrinking out unimportant parameters in the model. Ridge estimators have been shown to often improve predictive performance in the estimators [7]. For the tuning procedure, the grid of values selected for λ, the ridge penalty, were 0.00001, 0.0001, 0.001, 0.01, 0.1, and 1.

The inductive bias of the logistic regression algorithm comes from its assumption of linear seperability between the classes.

2.2.4 C4.5 Decision Tree (J48)

Decision Tree algorithms classify instances by forming a cascading decision making mechanism in the form of a "descision tree". The tree is constructed such that attributes found to carry more information about the class of the instances are used earlier in the mechanism than those attributes found to carry less information. After the tree is constructed, it is typically pruned (for a fixed confidence factor, which we will call c). J48 is an open source Java implementation of Quinlan's C4.5 algorithm [10]. For the datasets mentioned above, J48 was tuned using a grid of confidence factors $c \in \{0.1, 0.25, 0.5, 0.75, 1\}$

The inductive bias of decision trees lie in the preferences for simpler trees versus complicated ones. Decision trees do not need linearly seperable groups in order to be succesful classifiers.

2.2.5 RIPPER (JRip)

The Repeated Incremental Pruning to Produce Error Reduction (RIPPER) algorithm [3] classifies instances by a utilizing a set of rules created by iteratively modifying an initial set of rules. The initial set of rules is created with the IREP* algorithm. For each rule, two new rules are generated: one is created by adding and pruning rule hypotheses (antecedents) to make a "replacement" rule and the other is created by greedily adding conditions to create a "revision rule". The MDL heuistic is then used to select the best rule between the original, the revision, and the replacement rules. IREP* is then used to cover any remaining positive examples. This algorithm was tuned for each data set by testing the number of optimizations $n \in \{1, \ldots, 10\}$.

The inductive bias of the RIPPER algorithm is similar to that of decision trees in that shorter, simpler rules are preferred. This method does not assume that the groups are linearly seperable.

2.2.6 Support vector machine (LibSVM or SMO)

Support Vector Machines classify instances by orginizing points into partitions of the domain. The partitions are defined by planes chosen to have maximal "margin" between support vectors. to find a plane (or hyperplane in more than 3dimensions) that maximizes the margins separating the groups. To generalize to 3 or more classes, support vector machines are constructed for each pairwise-class comparison, and the majority winner of each of the pairwise comparison is the class prediction for a new example. In cases where groups are not linearly seperable, a kernal transformation can be used to project the problem into a higher dimensional space where the classes are linearly seperable. This algorithm was tuned for each dataset by testing the cache size $c \in \{0.5, 1, 1.5\}$ and the number of random seeds $d \in \{1, 2, 4, 6\}$.

The inductive bias of support vector machines lies in the assumption that the classes are linearly seperated by an appreciable amount (by some margin) but not necessarily in the original dimension if the "kernel trick" is implemented.

2.2.7 Feedforward neural networks (Multilayer Perceptron)

Feedforward neural networks (FFN) [12] classify instances by forming a directed graph that contains no cycles, that is, the information passes through the constructed network in a forward only direction. The graph consists of connected nodes (neurons) that either engage or don't based upon signals presented to them from other neurons upstream. The signals are added in a weighted manner and the neuron fires if a specified threshold is met.

Back-propagation [11] is the most common learning method for multi-layer perceptrons. Neurons feed other neurons by manner of weights and ideal weights are found by utilizing gradient descent on a specified loss function where the loss function is minimized over the space of possible weights. The algorithm was tuned for each dataset by testing the learning rate $\lambda \in \{0.1, 0.3, 0.5\}$.

The inductive bias for FFN is the assumption that there is linear separability of the classes.

2.2.8 Kernel neural network (RBFNetwork)

An RBF Network [2] is essentially a neural network with that uses radial basis functions as the activation function for neurons. There is customarily one hidden layer of nodes (a two layer network topology), in contrast to FNN where there may be multiple hidden layers. The input layer is fed into the single hidden layer and this hidden layer uses RBFs for activation functions. There is a nueron in the hidden layer corresponding to each instance in the training set $\mathbf{x}^p$, $p \in \{1, \ldots, N\}$. RBFs are simply functions of the distance between two points: $\phi(|\mathbf{x} - \mathbf{y}|)$. Thus, for classifying instance $\mathbf{x}$ with features $\{x_i, \ldots, x_k\}$ each of the N hidden nuerons fires if $\phi(|\mathbf{x} - \mathbf{x}^p|)$ is sufficiently large. The algorithm was tuned for each dataset by testing the ridge factor for quadratic penalty on output weights $\lambda \in \{0.1, 0.001, 0.00001\}$.

The inductive bias for these algorithms is the same as those for FFN; it is assumed that there is linear separability of the classes

2.2.9 Ensemble Algorithm (AdaboostM1)

AdaboostM1 is an algorithm built from an ensemble of classifiers. The algorithm is described in detail in [4]. The general idea of boosting is that using an ensemble of different classifiers and taking a majority vote will often produce a more accurate classifier. At each iteration of the algorithm, larger weights w_i are applied to observations who were incorrectly classified by the previous step. This effectively forces the algorithm to put more effort into classifying these observations. A boosting weight is computed at each step which quantifies how well each iteration classifies correctly. The final output takes a "weighted-majority" vote to classify the observations.

The J48 classifier was chosen as the base algorithm for boosting. It was tuned by the number of boosting iterations (using 10, 20, 30, and 40 iterations) and the pruning weights (weights of 0, 50, and 100).

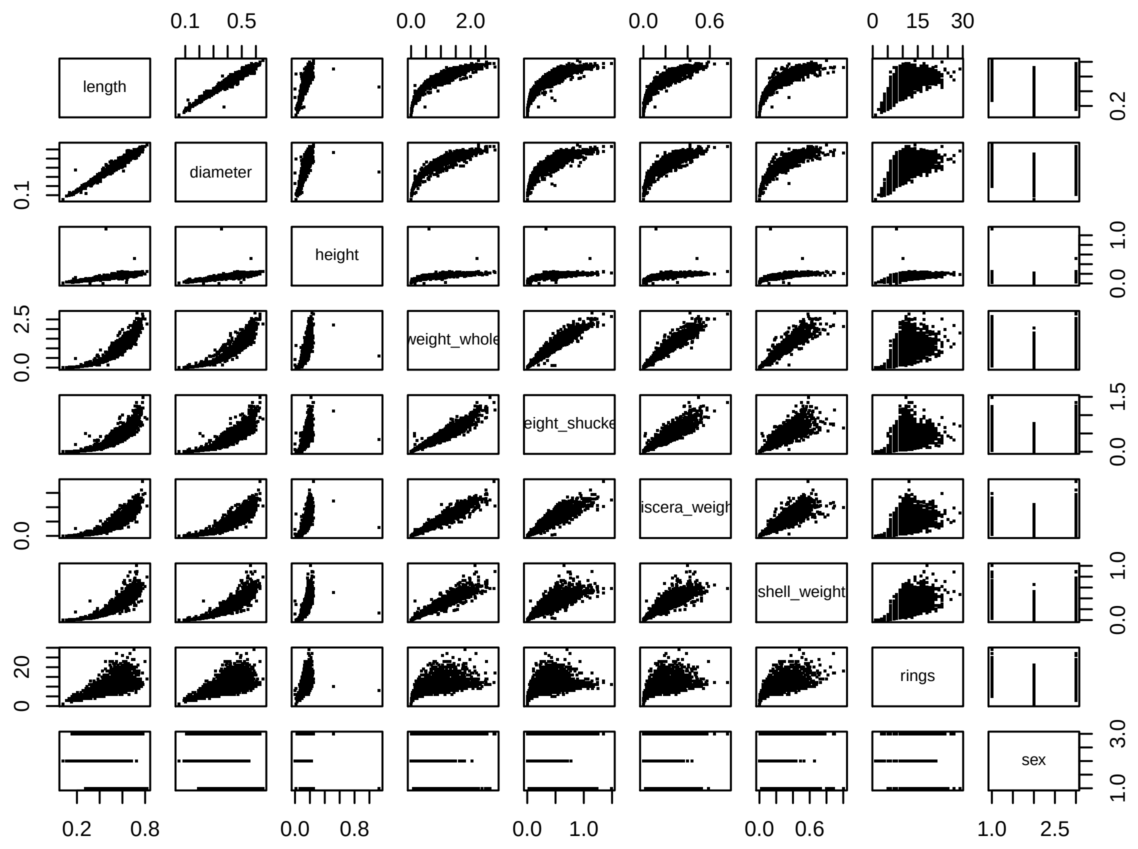
The inductive bias of Adaboost comes from the assumption that a collection of classifiers will perform better than a single classifier, which could be false if we are creating a collection of poor performing classifiers.

2.3 Hypothesis

Hypothetical performance of the different algorithms on the different datasets will no be postulated. Of key consideration will be the inductive bias of the selected classifiers. On a general note, the authors hypothesize that no single classifier will out perform the rest as the datasets chosen are not homogeneous in size, number of features, or character of features.

2.3.1 Abalone

An initial plot of the abalone predictors yeilds a few peices of information.



The first being that the covariates (the rows and columns not corresponding to sex in the plot above) share much information with one another. When their is a lot of shared information in the predictors, some methods may have less stable solutions due to multicollinearity. The second peice is that this may not be a completely linearly seperable problem. The plots do not show that there is a clear distinction between males, females, and infants (V1). So the algorithms with a linearly seperable inductive bias may not do well classifying such data nor algorithms that require points close together to in a metric sense to be classified similarly. Thus, as a simply hypothesis, it is expected that the boosting algorithm, the decision tree algorithms (J48 and JRip), Naive Bayes, and Logistic regression will work the best across all three loss functions.

Of course, we are only looking at the pairwise relationships in 2 dimensions, and their may be linear seperation in higher-dimensions.

2.3.2 Car Evaluation

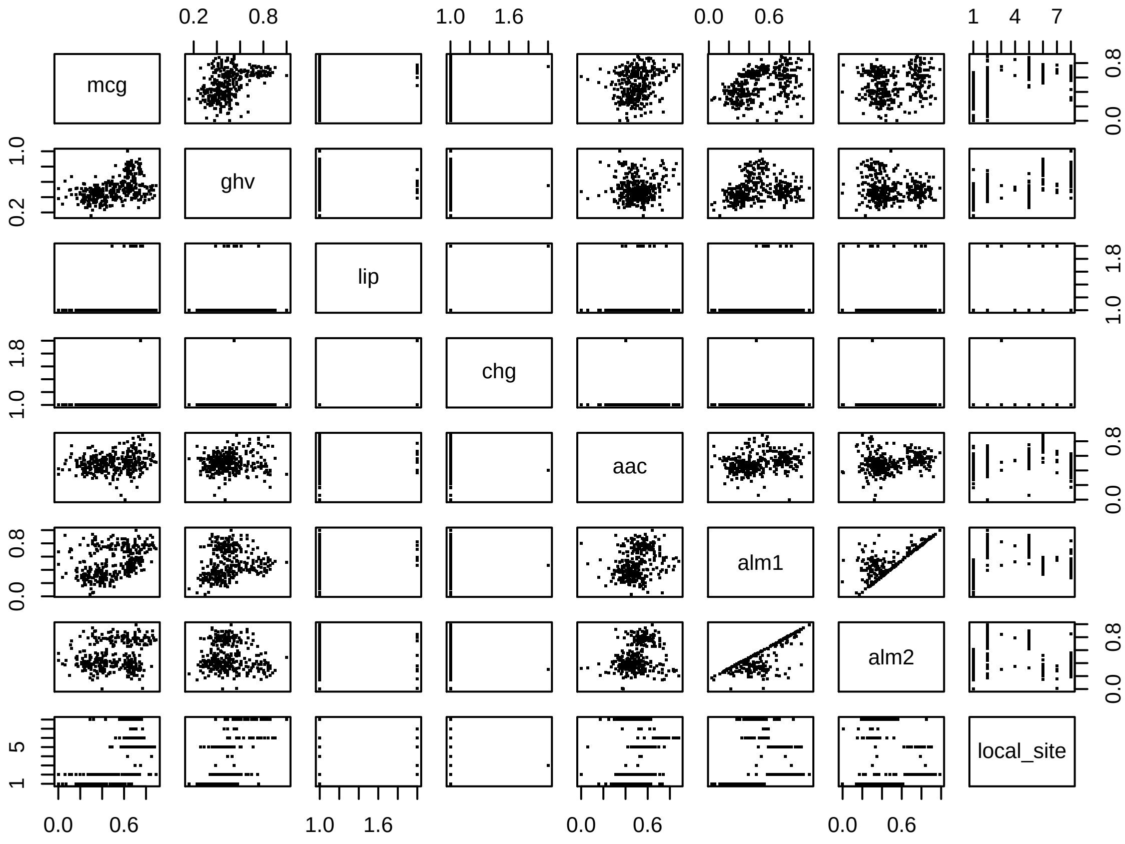
This data was derived from a simple hierarchical decision mode, which the authors assume implies that this is fake data generated by some decision tree method. Thus, one would expect the decision tree algorithms (J48, JRip) to perform best. Also, one would expect a Multilayer Peceptron to work well as it implements a heirachy of sorts in the layers of the neural network.

2.3.3 Contraceptive Method

The authors are unable to determine if this data is likely linearly seperable or not. One would suspect that conditional independence among the predictors is highly violated with this type of social data. Thus, it is hypothesized that the Naive Bayes classifier will work most poorly. The decision tree classifiers will work best as there may likely be one or two predictors that carry significantly more information than the others concerning contraceptive methods imployed (e.g. education of mother).

2.3.4 Ecoli

Initial plots of the ecoli data show that while groups can be visually distinugeshed in some dimensions, other are more problematic. The classifier variable is denoted in the plot below by "local_site".



There is also a relatively small amount of training data. The linearly separable assumption is more appropriate than for the yeast dataset; distinct differences in some attributes across the different localization sites can be seen. We also have very few examples in some of the classes, making the classificaion of these to be difficult. Given that linear seperability does not seem feasible, it is proposed that IBk, J48, Ripper, and Adaboost will acheive the best performance amongst the candidates. Since boosting generally increases performance, it is predict that his will perform the best.

2.3.5 Flag

There is a high degree of conditional dependence within this data set. For example, there is an indicator variable for the color red anywhere on the flag and a variable for the color in the bottom-left corner and these two variables are clearly not independent. Also, several continents share similarities among the flags of contries within (e.g. African flags commonly have red, green, and black colors) thus it is hypothesized that the IBk and SVM algorithms will perform best.

2.3.6 Tic Tac Toe

Because Tic-Tac-Toe is a deterministic game, it seems likely that the neural networks will be good classifiers for determining if "x" or "o" wins the game. Also, it is hypothesized that the decision tree algorithms would likely do well, J48 and JRip.

2.3.7 Wine

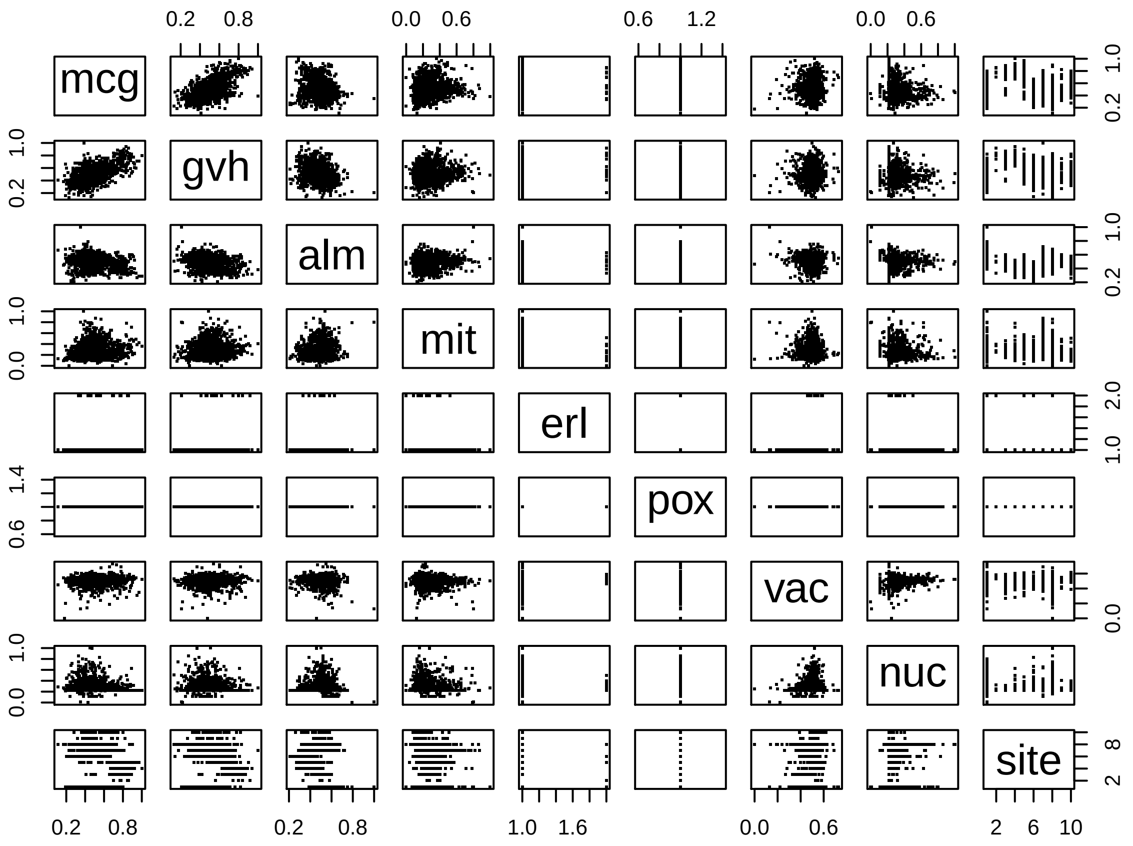
There are too many features to make exploratory graphics without some sort of prior knowledge about the data. On the description page for the data set, the author writes "I think that the initial data set had around 30 variables, but for some reason I only have the 13 dimensional version." It seems like a reasonable assumption that there is a significant amount of predictive information missing in this data set. Consequently, noise in the data could be problematic. This would highlight RBF networks as a poor classifier and perhaps SVM and Naive Bayes classifiers as the best of the candidates.

2.3.8 Wine Quality

The feature to be classified is the "quality" of wines which is a human-defined metric. Thus, one would think that perhaps a multilayer perceptron would do well at estimating the nuance in the wine quality score. Again, there are too many features for plots to readily be of use in this context.

2.3.9 Yeast

Initial plots of the yeast data show that certain classes can be visually distingueshed in some of the dimensions.



Although the linear seperable assumption does not seem to hold, it does appear that observations close to one another seem to have similar classes overall. Although it may not be as efficient as the instance based methods, the rule or tree based methods could potentially separate these groups if appropriately grown and pruned. It is postulated that IBk will perform the best amongst the candidates, followed by J48, Ripper, and Adaboost. Since boosting generally performs better than a single instance of a classifier, we predict that the ensamble will have improved performance compared to a single decision tree.

2.3.10 Zoo

When examining the raw data, it can be seen that most of the classes are almost completely separable in a handful of the attributes present. So it seems to reason that combining them together would result is highly successful classification. This is also some evidence of a linearly separable problem. Due to this, we hypothesize that our loss measures will be greatest overall for Naive Bayes, Logistic regression, and MultilayerPerceptron. Due to the relatively small amount of training data for this problem, we hypothesize that Naive Bayes will perform the best of these three.

3 Tuning

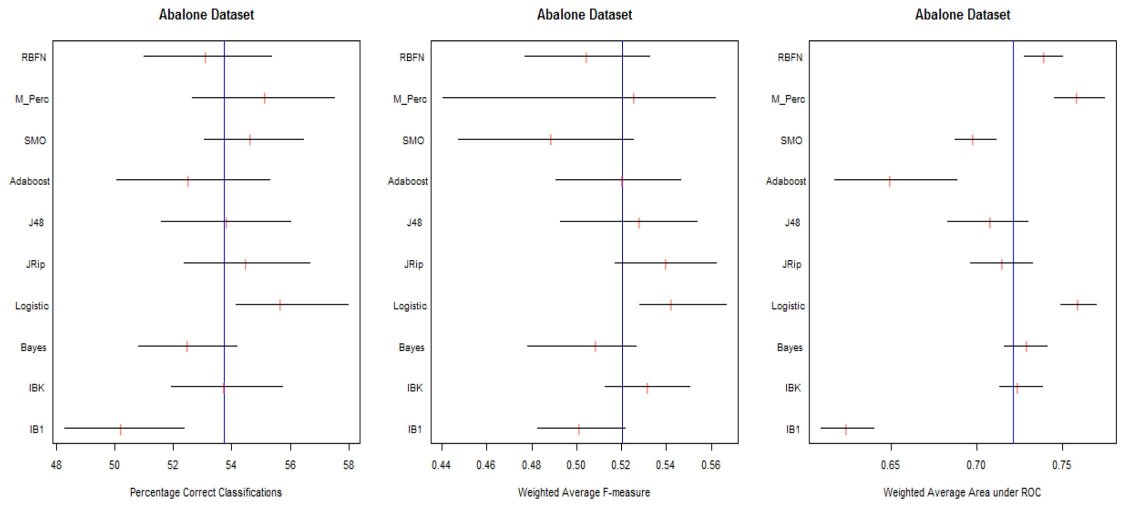
As stated earlier, individual tuning to each dataset was done prior to the experiment, and these results are displayed in a supplementary section at the end of this report. For each dataset, the algorithms were tuned using a grid of tuning parameters and 10-fold cross-validation, selecting the value which minimized the average root mean-squared error of prediction. Unless, otherwise stated, the options for the algorithms in Weka were set to the default setting.

4 Results

The classifiers are now compared by dataset with a special graphic included for each analysis. This graph shows the middle 95% of loss function values from the experiments run with the median ploted on each interval with a red line.

4.1 Abalone

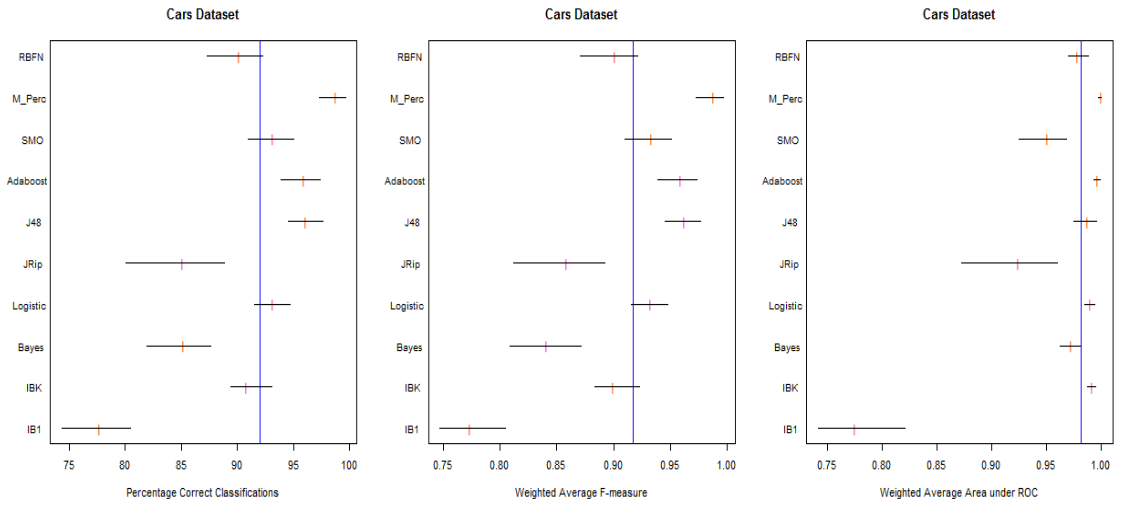
Since the abalone dataset was not used originally for classifying sex, it makes sense that all classifiers performated rather poolry. The clear winner across all threee measures was logistic regression.



In terms of classification percentage, the differences between the algorithms were quite small. For the weighted F-measure, many of the algorithms performed similarly as well. It is interesting to note that the Multilater Perceptron had a very wide variability on this loss scale. The clearest differences in the algorithms are manifested in the weighted AUC measures, with Logistic and Multilayer Perceptron performing the best. Ripper and J48 performed similarly on this dataeset. We see a big increase in using IBk over IB1. Interstingly, Adaboost performed poorly compared to the other algorithms, so the ensemble of trees actually decreased the classification performance in this example.

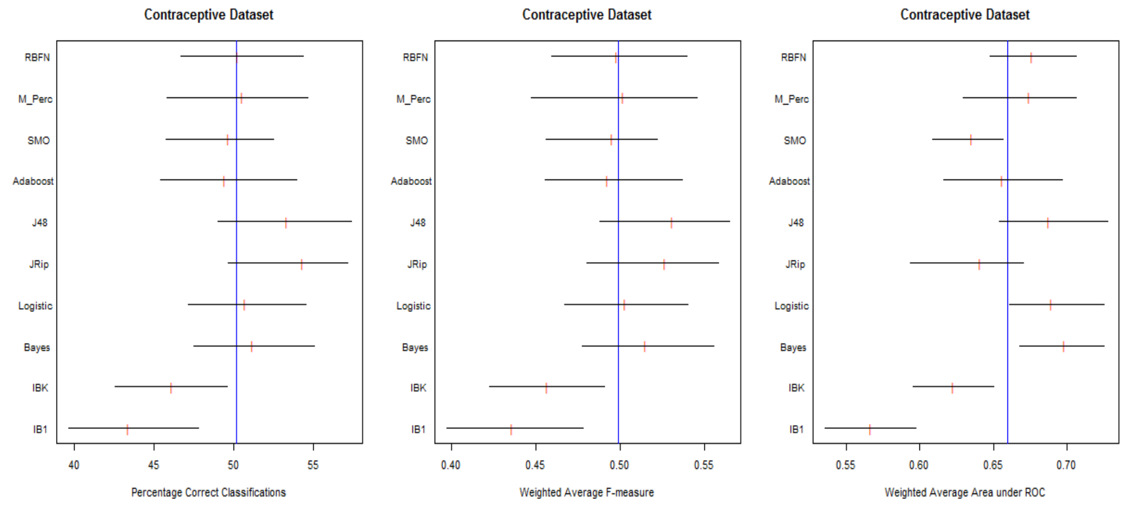
4.2 Cars

As this data was most likely fake, it is no surpise that this was easy data to classify. The MLP algorithm performed best, achieving nearly perfect classification results under all three loss measures. The J48 and boosting classifiers also performed very well across all three loss measures as was hypothesized. The worst classifier appears to be the IB1 classifier.



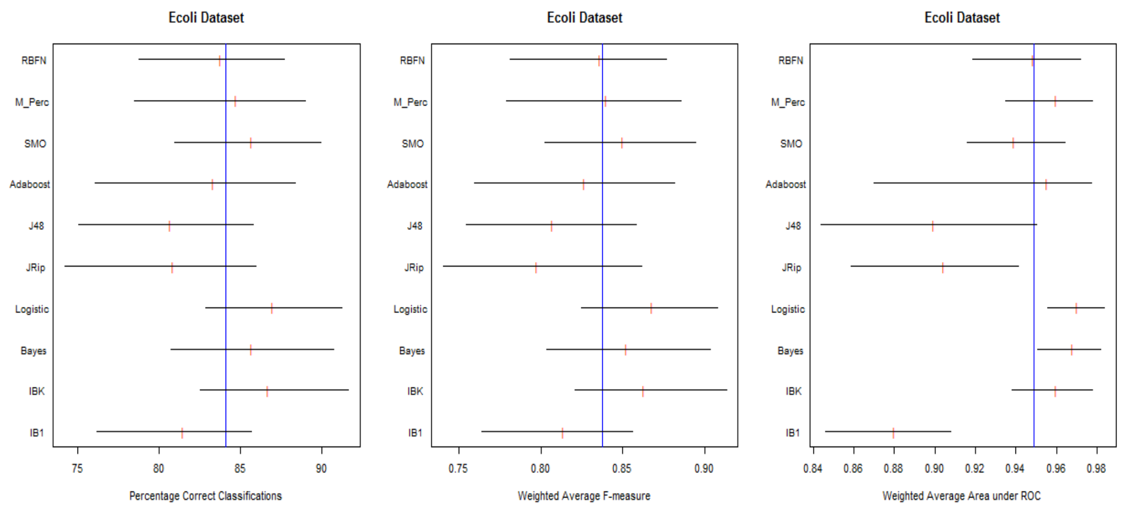
4.3 Contraceptive Use

For this set of experiments, it appears that the J48 learner performed best as hypothesized. Although, the performance was not incredible in an absolute sense (the mean percentage of correct classifications was approximately 53%) and variability in performance was large across all three loss funtions. Again, the worst classifier was the IB1 algorithm.



4.4 Ecoli

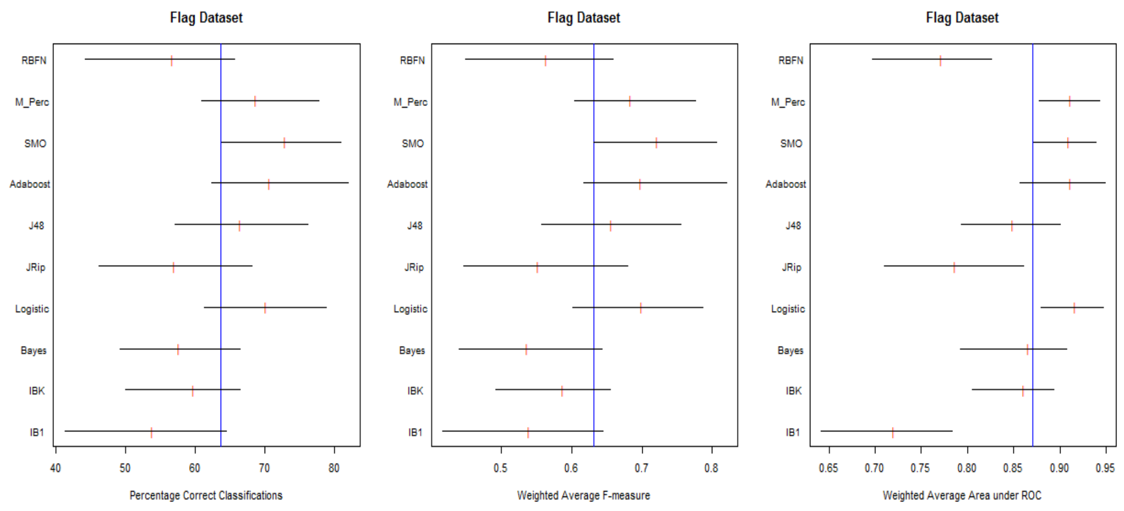
The algorithms performed better on the Ecoli dataset compared to the yeast dataset. We see overall higher classifications, F-measures, and AUROCs. Logistic regression, Naive Bayes, and K nearest neighbors perform at the top of the list. A decrease in performance using the rule based methods and the ensemble is seen. The wide variation in these measures could be a result of the overfitting or the methods inability to easily pick classes in linearly seperable problems, something that the top methods can do well.



4.5 Flags

The best performing algorithms for this dataset appear to be MLP, SVM, Logistic Regression, and the boosted decision tree classifiers. It was hypothesised that the SVM classifier would perform well, but the others were a surprise. Also, it was hypothesized that the nearest neighbor methods would perform well and it appears that the opposite is true.

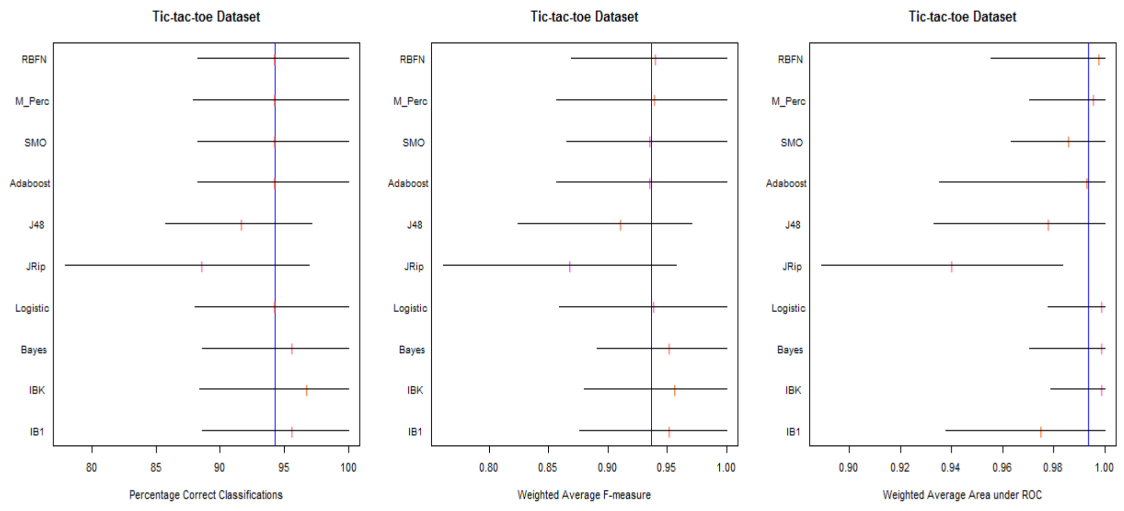
One interesting thing to note is that for all learners, the correct classification rates were not particularly high (53% to 72%) but for the four highest performing classifiers listed above the AUROC values were above .9 which is relatively high. This suggests that for this data set, it is more difficult to be correct than it is to be not wrong, i.e. that precision is easier to achieve than recall.



4.6 Tic-tac-toe

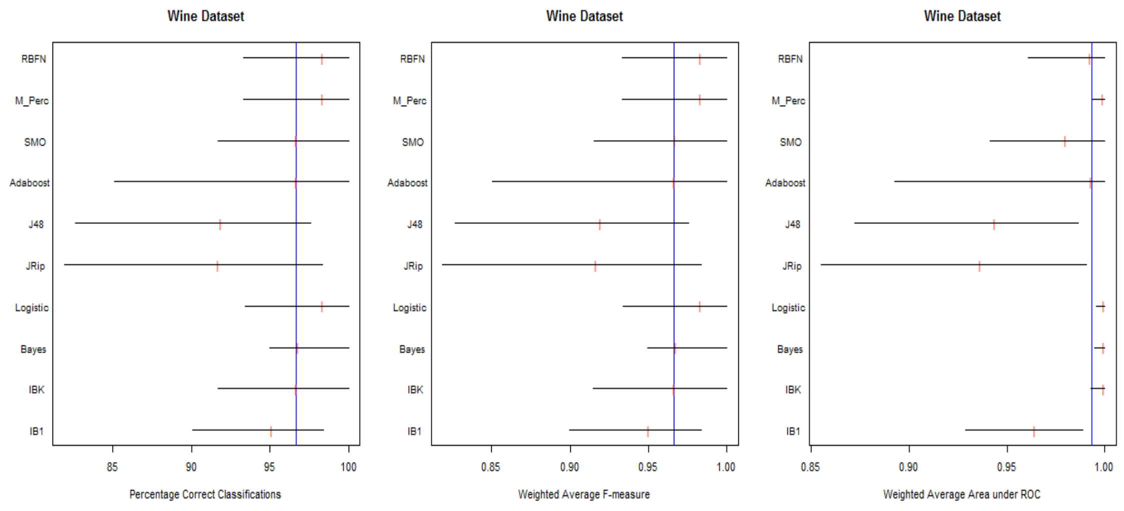
Because this was an intuitively relatively easy problem to classify, it is no surprise that all classifiers performed well. In fact, all algorithms acheive perfect values of the loss functions except for the JRip and J48 algorithms.

This is exactly the opposite of what was hypothesised. Perhaps because classification was so simple, the inductive bias was not a useful tool in hypothesizing about classification rates.



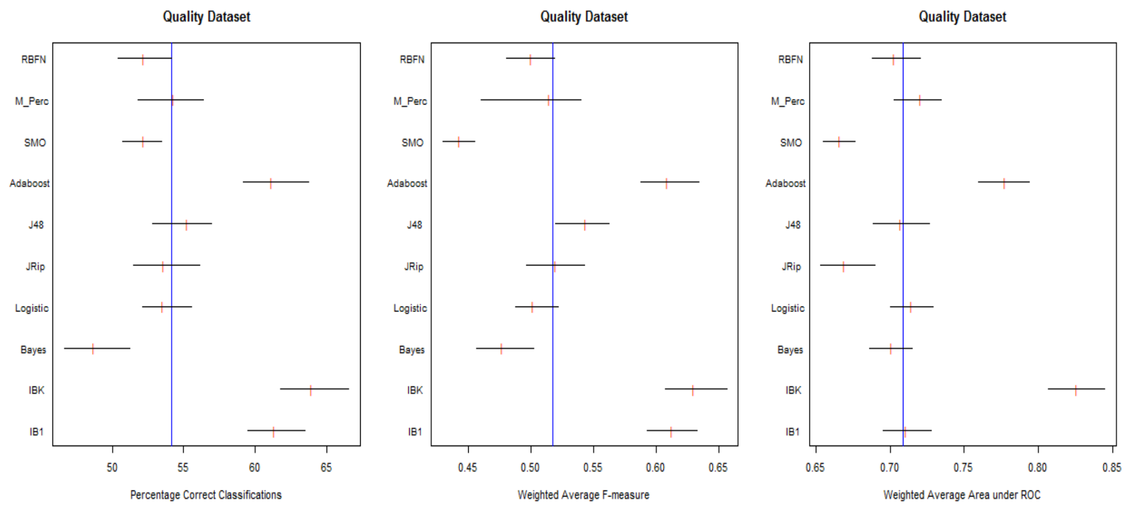
4.7 Wine

All classifiers performed well on this dataset with IBk, Naive Bayes, Logistic Regression, and MLP performing at a nearly perfect rate especially in terms of weighted AUROC. This may be because the size of the dataset was so large and consequently large training sets were available. The hypotheses listed above are mostly off-point. Particularly, the alleged missing features did not seem to impair classification whatsoever.



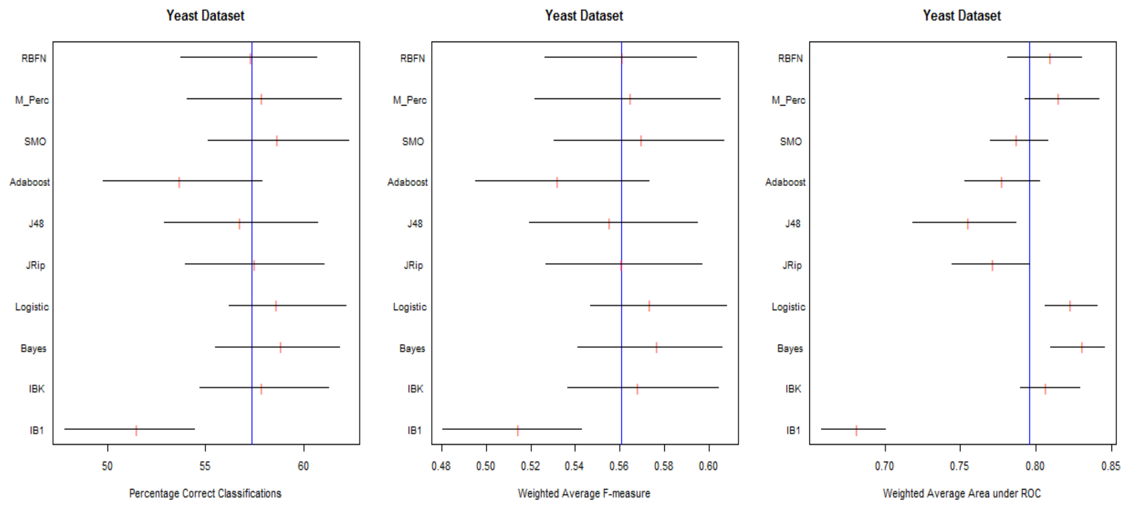
4.8 Wine Quality

Oddly, IBk performed much better than all other classifiers. The boosted decision tree algorithm classified second best across all loss measures, although none were incredible performers. It was hypothesized that the neural networks would work well with this data set and that does not appear to be the case as neither classifier performed better than average under any metric investigated.



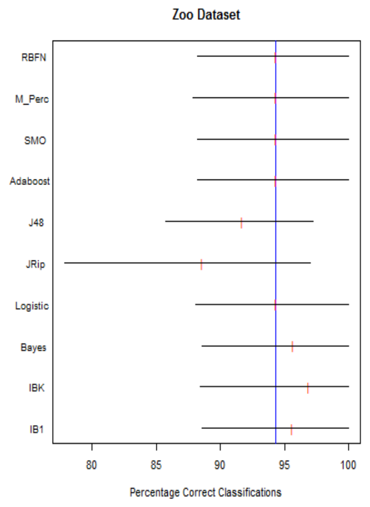
4.9 Yeast

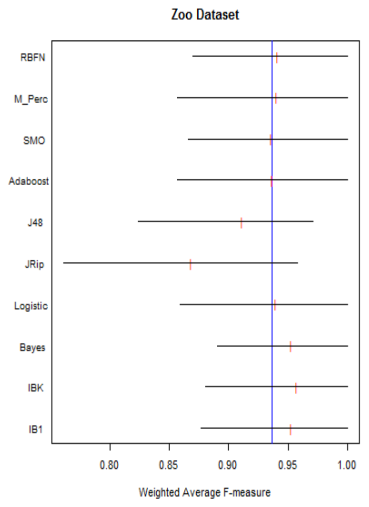
For misclassifcation rate and the Average F-measure, small differences between the algorithms are seen, with IB1 performing noticably worse. An interesting result is noticed when examining Adaboost results. Adaboost performs slightly worse than J48 in terms of misclassification rate and average F-measure, but performs slightly better in terms of the AUC. Two of the classes compose 60% of the examples in this dataset, and when looking at individual runs of the algorithm, we see that they perform well at classifying these sites and poorly on the others. In terms of weighted AUC, top performances come from Logistic regression, Naive Bayes, and MultilayerPerceptron, followed closely by the IBK and RBFNetworks.

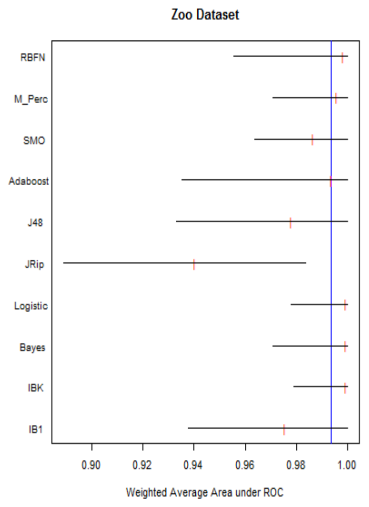


4.10 Zoo

All algorithms performed quite well on the zoo dataset. This seems to follow the hypotheses above: given the amount of information present on the different species it seems reasonable that one could classify them into similar groups of species. The classificaiton rate and the weighted F-measure provide practically the same information, and there is not much difference between the algorithms other than the slight decrease in performance using Ripper and J48, with J48 using performing slightly better. In the weighted AUC, it is not visually clear which algorithm performs best overall, although IBK, NaiveBayes, Logistic, and MultilayerPerceptron all have a median value very close to 1. There was also good performance from RBFNetwork and SMO, but these algorithms had slightly more variation in the observed values, making them slightly less desirable than those stated earlier. The ensemble of trees in Adaboost generally performed slightly better than the single tree produced by J48.







Supplementary: Tuning Parameter Information

	Dataset	IBK.K	IBK.wts	log.rdg	J48.C	Ripper.OptRun	SMO.C	SMO.d
1	Abalone	8	N	0.00	0.10	10	1.50	2
2	Car	3	Y	0.00	0.25	7	1.50	1
3	CMC	8	N	1.00	0.10	10	1.00	2
4	Ecoli	8	Y	0.10	0.75	4	0.50	4
5	Flag	7	Y	1.00	0.10	7	0.50	1
6	TicTacToe	3	Y	0.10	0.10	2	1.00	1
7	Wine	8	Y	0.01	0.10	10	0.50	4
8	WineQuality	8	Y	0.10	0.10	10	0.50	2
9	Yeast	8	Y	0.01	0.10	3	1.00	4
10	Zoo	3	Y	0.00	0.10	10	1.00	1

	Dataset	MP.learn	RBFN.K	RBFN.ridge	Ada.iter	Ada.thresh
1	Abalone	0.30	5	0.00	10	0
2	Car	0.50	5	0.10	30	50
3	CMC	0.30	5	0.10	10	50
4	Ecoli	0.30	2	0.10	40	50
5	Flag	0.50	2	0.00	10	100
6	TicTacToe	0.10	5	0.10	40	100
7	Wine	0.50	5	0.10	40	100
8	WineQuality	0.50	2	0.10	10	50
9	Yeast	0.30	2	0.00	10	50
10	Zoo	0.50	2	0.10	10	100

References

[1] Aha, D., and D. Kibler (1991). “Instance-based Learning Algorithms.” *Machine Learning.* 6:37-66.

[2] Broomhead, D. S., and David Lowe. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks (Technical report). RSRE. 4148.

[3] Cohen, W. (1995) “Fast Effective Rule Induction”. *Twelfth International Conference on Machine Learning.* 115-123.

[4] Freund, Y., and R. Schapire. (1996). “Experiments with a new boosting algorithm”. *Thirteenth International Conference on Machine Learning*, 148-156.

[5] Hastie, T., et al. (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer.

[6] John, George H., and Pat Langley (1995). “Estimating Continuous Distributions in Bayesian Classifiers.” Eleventh Conference on Uncertainty in Artificial Intelligence, San Mateo, 338-345.

[7] le Cessie, S., and J.C. van Houwelingen. (1992). “Ridge Estimators in Logistic Regression.” *Applied Statistics.* 41(1), 191-201.

[8] Lichman, M. (2013). *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[9] Mitchell, Tom M. (2015) “Chapter 3: Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression”. *Machine Learning.*

[10] Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

[11] Rumelhart, D.E., pet. al. Nature 323, 533-536, 9 October 1986.

[12] Warner, B., and M. Misra. (1996). “Understanding Neural Networks as Statistical Tools”. *The American Statistician.* 50(4):284-293.