Local Dagging of Decision Stumps for Regression and Classification Problems

D. S. Anyfantis*, M. G. Karagiannopoulos*, S. B. Kotsiantis* and P. E. Pintelas*
* University of Patras /Department of Mathematics, Patras, Hellas

Abstract— **Numerous data mining problems involve an investigation of relationships between features in heterogeneous datasets, where different prediction models can be more appropriate for different regions. We propose a technique of dagging localized weak learners. We recognize local regions having similar characteristics and then build local experts on each of these regions describing the relationship between the data characteristics and the target value. We performed a comparison with other well known combining methods on standard classification and regression benchmark datasets using decision stump as based learner, and the proposed technique produced the most accurate results.**

I. Introduction

Instance-based learners classify an instance by comparing it to a database of pre-classified examples. The fundamental assumption is that similar instances will have similar target values. The corresponding components of an instance-based learner are the distance function which determines how similar two instances are, and the prediction function which specifies how instance similarities yield a final prediction for the new instance [1]. Local learning [2] can be understood as a general principle that allows extending learning techniques designed for simple models, to the case of complex data for which the model's assumptions would not necessarily hold globally, but can be thought as valid locally. A simple example is the assumption of linear separability, which in general is not satisfied globally in classification problems with rich data. Yet any classification algorithm able to find only a linear separation, can be used inside a local learning procedure, yielding an algorithm able to model complex non-linear class boundaries.

There has been much research work on ensemble learning for regression in the context of neural networks, however there has been less research carried out in terms of using homogeneous ensemble techniques to improve the performance of simple regression algorithms. Classification problems have dominated research on dagging to date. The application of dagging to regression problems, on the other hand, has received little investigation.

In this paper we develop a dagging method for classification and regression problems that works locally. We performed a comparison with other well known combining methods on standard classification and regression benchmark datasets using decision stump as based learner, and the proposed technique produced the most accurate results.

In the next section, we discuss the localized experts, while current ensemble approaches and work are described in section 3. In Section 4 we describe the proposed method and investigate its advantages and limitations. In Section 5, we evaluate the proposed method on several UCI datasets by comparing it with standard combining techniques and other lazy methods. Finally, section 6 concludes the paper and suggests further directions in current research.

II. Local Weighted Learning

When all training examples are considered when predicting the value of a new test case, the algorithm works as a global method, while when the nearest training examples are considered, the algorithm works as a local method, since only data local to the area around the testing instance contribute to the prediction. Local methods have significant advantages when the probability measure defined on the space of symbolic objects is very complex, but can still be described by a collection of less complex local approximations. When the size of the training set is small compared to the complexity of the learner, the learning machine usually overfits the noise in the training data. Thus effective control of complexity of a learner plays a key role in achieving good generalization. Some theoretical results and experimental results [18], [22] indicate that a local learning algorithm (that is learning machine trained on the training subset) provides a feasible solution to this problem.

In fact, local learning is not a new concept and it has appeared in the early years of pattern recognition. The obvious example is the k-nearest neighbor method: given a testing pattern, we estimate its target value from the closest pattern in the training set. A list of objections to k-nearest neighbor algorithms includes the following: a) voting/averaging used to combine the values of the nearest k instances, b) uniform neighborhood shape (spherical) regardless of instance location and c) uniform weight given to all features, instances and neighbors

Our ultimate goal is not to improve the nearest neighbor algorithm, but to improve the accuracy by combining local learners. The authors of [7] proposed a theoretical model of a local learning algorithm and obtained bounds for the local risk minimization estimator for pattern recognition and regression problems using structural risk minimization principle. In local learning, each local model is trained entirely independently of all other local models such that the total number of local models in the learning system does not directly affect how complex a function can be learned - complexity can only be controlled by the level of adaptability of each local model. This property avoids overfitting if a robust learning scheme exists for training the individual local model.

The authors of [9] extended the idea of constructing local simple base learners for different regions of input space, searching for appropriate architectures that should

be locally used and for a criterion to select a proper unit for each region of input space. They proposed a hybrid MLP/RBF network by combining RBF and Perceptron units in the same hidden layer and using a forward selection to add units until an error goal is reached.

III. Ensembles Of Classification And Regression Models

Empirical studies showed that ensembles are often much more accurate than the individual base learners that make them up [5], and recently different theoretical explanations have been proposed to justify the effectiveness of some commonly used ensemble methods [16]. Currently, there are two main approaches to model combination. The first is to create a set of learned models by applying an algorithm repeatedly to different training sample data; the second applies various learning algorithms to the same sample data. The predictions of the models are then combined according to an averaging/voting scheme or a stacking algorithm [16]. In this work we propose a combining method that uses one learning algorithm for building an ensemble of regression models. For this reason this section presents the most well-known methods that generate sets of base learners using one base learning algorithm.

Starting with bagging [8], we will say that this method samples the training set, generating random independent bootstrap replicates, constructs a learner on each of these, and aggregates them by a simple majority vote or averaging procedure in the final decision rule. Therefore, taking a bootstrap replicate one can sometimes avoid or get less misleading training objects in the bootstrap training set. Consequently, a learner constructed on such a training set may have a better performance.

Another method that uses different subset of training data with a single learning method is the boosting approach [13]. It assigns weights to the training instances, and these weight values are changed depending upon how well the associated training instance is learned by the learner; the weights for misclassified instances are increased. Thus, re-sampling occurs based on how well the training samples are classified by the previous model. After several cycles, the prediction is performed by taking a weighted vote of the predictions of each learner, with the weights being proportional to each learner's accuracy on its training set. AdaBoost is a practical version of the boosting approach for classification problems [13]. The AdaBoost.R algorithm [10] attacks the regression problem by reducing it to a classification problem. Friedman has also explored regression using the gradient descent approach [14]. In each iteration, the Additive Regression algorithm constructs goal values for each data-point xi equal to the (negative) gradient of the loss of its current master hypothesis on x_i. The base learner then finds a function in a class minimizing the squared error on this constructed sample.

Dagging [21] creates a number of disjoint, stratified folds out of the data and feeds each chunk of data to a copy of the supplied base learner. Predictions are made via majority vote for classification problems and via averaging procedure for regression problems.

MultiBoosting [23] is another classification method of the same category that can be considered as wagging committees formed by AdaBoost. Wagging is a variant of bagging; bagging uses resampling to get the datasets for training and producing a weak hypothesis, whereas wagging uses reweighting for each training example, pursuing the effect of bagging in a different way.

In [19] another classification meta-learner (DECORATE, Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples) is presented that uses a learner (one that provides high accuracy on the training data) to build a diverse committee. This is accomplished by adding different randomly constructed examples to the training set when building new committee members. These artificially constructed examples are given category labels that disagree with the current decision of the committee, thereby directly increasing diversity when a new classifier is trained on the augmented data and added to the committee.

IV. Proposed Algorithm

The proposed algorithm builds a model for each point to be estimated, taking into account only a subset of the training points. This subset is chosen on the basis of the preferable distance metric between the testing point and the training point in the input space. For each testing point, a dagging ensemble of a weak learner is thus learned using only the training points lying close to the current testing point. Generally, the proposed ensemble consists of the four steps (see Figure 1)

1) Determine a suitable distance metric.
2) Find the k nearest neighbors using the selected distance metric.
3) Apply dagging to the DS algorithm using as training instances the k instances
4) The answer of the daggting ensemble is the prediction for the testing instance.

Figure 1. Local dagging ensemble

The proposed ensemble has some free parameters such as the distance metric. In our experiments, we used the most well known - Euclidean similarity function - as distance metric. We used 10 iterations for the dagging process in order to reduce the time need for predicting the value of a new instance. Decision stumps (DS) are one level decision trees [15]. We can find the best stump just as we would learn a node in a decision tree: we search over all possible features to split on, and for each one, we search over all possible thresholds induced by sorting the observed values. In classification problems, each node in a decision stump represents a feature in an instance to be classified, and each branch represents a value that the node can take. Instances are classified starting at the root node and sorting them based on their feature values. In regression problems, DS (or regression stumps) do regression based on mean-squared error where each node in a decision stump represents a feature in an instance to be predicted, and each branch represents a value that the node can take. At worst a decision stump will reproduce the most common sense baseline, and may do better if the selected feature is particularly informative.

The proposed algorithm also requires choosing the value of K. There are several ways to do this. A first, simple solution is to fix K a priori before the beginning of the learning process. However, the best K for a specific dataset is obviously not the best one for another dataset. A second, more time-consuming solution is therefore to

determine this best K automatically through the minimization of a cost criterion. One way to do that is to evaluate the estimation error on a test set and thus keep as K the value for which the error is the least. In the current implementation we decided to use a fixed value for K (=50): a) in order to keep the training time low and b) about this size of instances is appropriate for a simple algorithm, to build a precise model according to [12], [18].

Our method shares the properties of other memory-based classification methods such as no need for training and more computational cost for the prediction. Besides, our method has some desirable properties, such as better accuracy and confidence interval.

V. EXPERIMENTS

We perform comparisons with other combining methods on classification and regression problems.

A. Using the proposed technique as classification method

We have experimented with 27 classification datasets from the UCI repository [4]. These datasets cover many different types of problems having discrete, continuous, and symbolic variables. Some datasets have missing values, and some have a mixture of all the above.

In order to calculate the classifiers' accuracy, cross validation was run 10 times for each algorithm and the average value of the 10-cross validations was calculated. It must be mentioned that we used the free available source code for most of the algorithms by [25] for our experiments.

We compare the proposed ensemble methodology with:

- K-nearest neighbors using k=3 (most common used number of neighbors), as well as k=50 because the proposed algorithm uses 50 neighbors. In addition, we tested Kstar: another instance-based learner which uses entropy as distance measure [11].
- Local weighted DS using 50 instances
- Bagging DS, Boosting DS and MultiBoost DS (using 25 sub-classifiers)
- Decorate DS and Dagging DS

In following Tables, we represent as "v" that the specific algorithm performed statistically better than the proposed ensemble according to t-test with $p<0.05$. Throughout, we speak of two results for a dataset as being "significant different" if the difference is statistical significant at the 5% level according to the corrected resampled t-test [20], with each pair of data points consisting of the estimates obtained in one of the 100 folds for the two learning methods being compared. On the other hand, "*" indicates that the proposed ensemble performed statistically better than the specific algorithm according to t-test with $p<0.05$. In all the other cases, there is no significant statistical difference between the results (Draws). In the last row of the table one can also see the aggregated results in the form (α/b/c). In this notation "α" means that the proposed ensemble is significantly less accurate than the compared algorithm in α out of 27 datasets, "c" means that the proposed algorithm is significantly more accurate than the compared algorithm in c out of 27 datasets, while in the remaining cases (b), there is no significant statistical difference.

In the last raw of the Table I one can see the aggregated results. The proposed ensemble is significantly more accurate than single DS in 18 out of the 27 datasets, while it has significantly higher error rate in none dataset. What is more, the proposed ensemble is significantly more accurate than 3NN and 50NN in 5 and 12 out of the 27 datasets, respectively, whilst it has significantly higher error rate in 2 and 1 datasets. Likewise, the proposed ensemble is significantly more accurate than local weighted DS and Kstar in 9 out of the 27 datasets, whilst it has significantly higher error rate in 2 and 5 datasets, respectively. In addition, local dagging is significantly more accurate than Bagging DS in 17 out of the 27 datasets, whilst it has significantly higher error rate in none dataset. Furthermore, Multiboost DS and Decorate DS have significantly lower error rates in 2 and none out of the 27 datasets than the proposed ensemble, respectively whereas they are significantly less accurate in 10 datasets. Adaboost DS has significantly lower error rates in 3 out of the 27 datasets than local dagging, whereas it is significantly less accurate in 10 datasets. Finally, local dagging is significantly more accurate than simple dagging DS in 12 out of the 27 datasets, whilst it has significantly higher error rate in none dataset.

To sum up, the performance of the proposed ensemble is more accurate than the other well-known ensembles that use only the DS algorithm. The proposed ensemble can achieve an increase in classification accuracy about 17% compared to simple DS. The average relative accuracy improvement of the proposed methodology is from 1% to 10% in relation to the remaining methods.

B. Using the proposed technique as regression method

We experimented with 24 regression datasets from the UCI repository [4]. We compared the proposed ensemble methodology with other methods that are based on the same learning algorithm - DS:

- Simple DS algorithm and Dagging DS
- Local DS using 50 local instances. This method differs from the proposed technique since it has no boosting process.
- Bagging DS and Additive regression DS (using 10 sub-models). Both these methods work globally whereas the proposed method works locally.

Cross validation was run 10 times for each algorithm and the average correlation coefficient value of the 10-cross validations was calculated. In the last raw of the Table II one can see the aggregated results. The proposed ensemble is significantly more accurate than simple DS in 16 out of the 24 datasets, whilst it is significantly less accurate in 1 dataset. In addition, the proposed ensemble is significantly more accurate than Local DS in 11 out of the 24 datasets, while it is significantly less accurate in 9 datasets. What is more, the proposed ensemble is significantly more accurate than Bagging DS in 15 out of the 24 datasets, whilst it is significantly less accurate in 4 datasets. Furthermore, additive regression DS is significantly more accurate in 10 datasets than the proposed ensemble, whereas it is significantly less accurate in 12 datasets. Finally, dagging DS is significantly more accurate in 4 datasets than the proposed ensemble, whereas it is significantly less accurate in 14 datasets.

TABLE I.
COMPARING LOCAL DAGGING DS WITH INSTANCE BASED CLASSIFIERS AND OTHER ENSEMBLES THAT USE DS AS BASE LEARNER

Datasets	Local Dagging DS	Kstar	3NN	Local DS	DS	50NN	Boost DS	Bagging DS	Multiboost DS	Decorate DS	Dagging DS
audiology	72.05	80.32 v	67.97 *	72.68	46.46 *	35.95 *	46.46 *	46.46 *	46.46 *	46.46 *	46.69*
autos	75.03	72.01*	67.23 *	74.82	44.9 *	48.18 *	44.9 *	44.95 *	44.9 *	51.81 *	48.39*
balance-scale	88.13	88.72	86.74	84.16*	56.72 *	89.01	71.77 *	68.21 *	71.77 *	81.25 *	78.06*
breast-cancer	73.54	73.73	73.13	72.68	69.27	70.75	71.55	73.38	71.76	75.18	71.61
breast-w	96.32	95.35	96.61	96.4	92.33 *	95.9	95.28	92.56 *	95.05	95.78	95.45
colic	83.69	75.71*	80.95*	80.87*	81.52	84.04	82.72	81.52	82.9	82.03	81.84
credit-rating	86.61	79.1 *	84.96	83.61*	85.51	86.16	85.57	85.51	85.39	85.28	85.51
diabetes	73.66	70.19 *	73.86	73.2	71.8	74.68	75.37	72.45	75.19	76.09	75.23
Glass	70.23	75.31v	70.02	70.58	44.89 *	56.16 *	44.89 *	45.08 *	44.89 *	53.12 *	55.74*
haberman	72.44	70.27	69.77	69.81	71.57	72.91	74.06	73.07	73.8	71.61	73.94
heart-c	80.37	75.18 *	81.82	78.29	72.93 *	81.58	83.11 v	75.26 *	83.54 v	72.43 *	79.54
heart-h	82.55	77.83*	82.33	79.17*	81.78	83.98	82.42	81.41	81.91	81.78	81.61
heart-statlog	81.22	76.44*	79.11	76.33*	72.3 *	83.74	81.81	75.33 *	82.89	81.48	80.07
hepatitis	81.76	80.17	80.85	83.04	77.62 *	79.38	81.5	80.61	82.21	80.02	81.43
ionosphere	85.05	84.64	86.02	88.24v	82.57 *	71.65 *	92.34 v	82.66 *	90 v	90.4 v	84.59
Iris	94.67	94.67	95.2	94	66.67 *	90.53 *	95.07	68.87 *	94.73	93.93	80.13*
Labor	89.8	92.03	92.83v	85.3*	78.77 *	64.67 *	90.57	81.97 *	89.97	91.07	79.4*
lymphography	80.24	85.08 v	81.74	76.67 *	75.31 *	80.59	75.44 *	74.5 *	74.96 *	72.25 *	77.79*
monk1	82.17	80.27	78.97	77.22 *	73.41 *	59.8 *	69.79 *	73.41 *	70.37 *	70.94 *	67.28
monk2	59.42	58.35	54.74*	48.75*	59.58	62.13 v	53.99 *	61.13	54.19 *	61.95	59.71
monk3	90.35	86.22*	86.72*	93.44v	76.01 *	82.46 *	90.92	82.41 *	92.3	93.45 v	81.31*
Primary-tumor	43.71	38.02*	44.98	43.22	28.91 *	39.26 *	28.91 *	28.91 *	28.91 *	29.09 *	27.67*
sonar	77.99	85.11 v	83.76 v	76.62	72.25 *	68.25 *	81.06 v	73.21 *	77.54	72.91 *	73.02*
Titanic	78.79	77.56	78.9	79.05	77.6	77.56	77.83	77.6	77.62	77.6	77.6
Vehicle	69.86	70.22	70.21	69.58	39.81 *	63.47 *	39.81 *	40.14 *	39.81 *	47.35 *	45.55 *
Vote	94.96	93.22	93.08	95.4	95.63	90.41 *	96.41	95.63	95.63	95.59	95.63
Wine	95.79	98.72 v	95.85	96.79	57.91 *	96.46	91.57 *	86.27 *	91.22 *	96.45	83.2 *
W/D/L		5/13/9	2/20/5	2/16/9	0/9/18	1/14/12	3/14/10	0/10/17	2/15/10	2/15/10	0/15/12
Average accuracy	80.01	79,05	79,20	78,52	68,67	73,69	74,26	71,20	74,07	75,09	72.89

To sum up, the performance of the proposed ensemble is better than the other well-known ensembles that use only the DS algorithm. Several other models have been proposed for modeling and approximating the values of a continuous objective attribute by using the values of conditional attributes, such as local learners, regression trees, and neural networks.

Subsequently, we compared the proposed ensemble methodology with other well known regression algorithms:

- K-nearest neighbors using k=50 because the proposed algorithm uses 50 neighbors. In addition, we tested Kstar: another instance-based learner which uses entropy as distance measure [11].
- The most well known algorithm for training artificial neural networks – Back Propagation (BP) with one hidden layer and five neurons in this layer [25].
- Regression tree algorithm - RepTree [25] and Decision Table [15]

In the last raw of Table III one can see the aggregated results. The proposed ensemble is significantly more accurate than simple Kstar in 12 out of the 24 datasets, whilst it is significantly less accurate in 7 datasets. In addition, the proposed ensemble is significantly more accurate than 50NN in 10 out of the 24 datasets, while it is significantly less accurate in 9 datasets. What is more, the proposed ensemble is significantly more accurate than Decision Table in 9 out of the 24 datasets, whilst it is significantly less accurate in 7 datasets. Furthermore, the BP algorithm is significantly more accurate in 11 datasets than the proposed ensemble, whereas it is significantly less accurate in 12 datasets.

TABLE II.
COMPARING THE ALGORITHMS

Dataset	Local Dagging DS	Local DS		Bagging DS		Additive Regression DS		DS		Dagging DS	
auto93	0.77	0.72	*	0.74	*	0.77		0.59	*	0.71	*
autoHorse	0.88	0.92	v	0.79	*	0.86	*	0.72	*	0.82	*
autoMpg	0.93	0.89	*	0.77	*	0.87	*	0.74	*	0.83	*
autoPrice	0.85	0.89	v	0.82	*	0.90	v	0.81	*	0.84	
baskball	0.46	0.44	*	0.49	v	0.49	v	0.48		0.44	*
bodyfat	0.91	0.94	v	0.84	*	0.95	v	0.82	*	0.9	
breastTumor	0.18	0.10	*	0.23	v	0.28	v	0.22	v	0.22	v
cholesterol	0.19	0.12	*	0.12	*	0.15	*	0.04	*	0.13	*
cleveland	0.68	0.63	*	0.59	*	0.65	*	0.40	*	0.64	*
cloud	0.61	0.69	v	0.76	v	0.87	v	0.39	*	0.81	v
cpu	0.81	0.92	v	0.85	v	0.95	v	0.31	*	0.81	
echoMonths	0.7	0.62	*	0.69		0.63	*	0.70		0.65	*
fishcatch	0.83	0.94	v	0.85		0.94	v	0.83		0.88	v
housing	0.8	0.84	v	0.73	*	0.84	v	0.60	*	0.8	
lowbwt	0.77	0.78		0.78		0.78		0.78		0.78	
pbc	0.45	0.43	*	0.45		0.53	v	0.43		0.49	v
pwLinear	0.85	0.84		0.68	*	0.83	*	0.68	*	0.73	*
quake	0.12	0.09	*	0.09	*	0.10	*	0.09	*	0.1	
sensory	0.48	0.47		0.29	*	0.38	*	0.29	*	0.29	*
servo	0.81	0.89	v	0.79		0.85	v	0.79		0.78	*
stock	0.97	0.99	v	0.79	*	0.94	*	0.78	*	0.79	*
triazines	0.48	0.47		0.24	*	0.37	*	0.04	*	0.3	*
veteran	0.43	0.28	*	0.32	*	0.40	*	0.15	*	0.31	*
wisconsin	0.29	0.22	*	0.26	*	0.11	*	0.27		0.26	*
W-D-L		*9/4/11*		*4/5/15*		*10/2/12*		*1/7/16*		*4/6/14*	

Finally, the RepTree algorithm is significantly more accurate in 8 datasets than the proposed ensemble, whereas it is significantly less accurate in 12 datasets. To sum up, the performance of the proposed ensemble is more accurate than the other well-known regression methods.

VI. CONCLUSION

Local memory-based algorithms defer processing of the dataset until they receive request for prediction. A database of observed input-output data is always kept and the estimate for a new operating point is derived from an interpolation based on a neighborhood of the query point. Local techniques are an old idea in time series prediction [3]. Local learning can reduce the complexity of component learners and improve the generalization performance although the global complexity of the system can not be guaranteed to be low. In this paper we propose local dagging and our experiment for some real datasets shows that the proposed combining method outperforms other well known combining classification and regression methods as well as any individual learner.

The benefit of allowing multiple local models is somewhat offset by the cost of storing and querying the training dataset for each test set instance which means that lazy learners do not scale well for the large amount of data associated with many applications. Local weighted learning algorithms must often decide what instances to store for use during generalization in order to avoid excessive storage and time complexity. By removing a set of instances from a database the response time for the predictions will decrease, as fewer instances are examined when a query instance is presented. This objective is primary when we are working with large databases and have limited storage.

In a following work we will focus on the problem of reducing the size of the stored set of instances while trying to maintain or even improve generalization accuracy by avoiding noise and overfitting. In [6] and [24] can be found numerous instance selection methods that can be combined with local dagging technique.

REFERENCES

[1] D. Aha, Lazy Learning. Dordrecht: Kluwer Academic Publishers, 1997.

[2] C. G. Atkeson, A. W. Moore, and S. Schaal, Locally weighted learning for control. Artificial Intelligence Review, 11: pp. 75–113, 1997.

[3] C. Atkeson, A. Moore & S. Schaal, Locally weighted learning. Artificial Intelligence Review, 11(1-5), pp. 11-73, 1997.

TABLE III.
COMPARING THE ALGORITHMS

Dataset	Local Dagging DS	Kstar		Decision Table		BP		50NN		RepTree	
auto93	0.77	0.77		0.68	*	0.85	v	0.71	*	0.23	*
autoHorse	0.88	0.90		0.85	*	0.95	v	0.85	*	0.83	*
autoMpg	0.93	0.91		0.90		0.91	*	0.86	*	0.88	*
autoPrice	0.85	0.91	v	0.81	*	0.90	v	0.89	v	0.88	v
baskball	0.46	0.46		0.53	v	0.54	v	0.56	v	0.39	*
bodyfat	0.91	0.87	*	0.97	v	0.98	v	0.91		0.98	v
breastTumor	0.18	0.19		0.16		0.09	*	0.23	v	0.15	*
cholesterol	0.19	0.04	*	0.07	*	0.08	*	0.17		0.07	*
cleveland	0.68	0.55	*	0.52	*	0.44	*	0.71	v	0.54	*
cloud	0.61	0.81	v	0.84	v	0.88	v	0.77	v	0.80	v
cpu	0.81	0.97	v	0.92	v	1.00	v	0.92	v	0.90	v
echoMonths	0.7	0.39	*	0.72		0.42	*	0.72		0.70	
fishcatch	0.83	0.99	v	0.94	v	0.99	v	0.78	*	0.95	v
housing	0.8	0.90	v	0.81		0.90	v	0.77	*	0.85	v
lowbwt	0.77	0.62	*	0.78		0.60	*	0.75		0.78	
pbc	0.45	0.30	*	0.40	*	0.32	*	0.52	v	0.46	
pwLinear	0.85	0.72	*	0.83	*	0.90	v	0.85		0.89	v
quake	0.12	0.08	*	0.09	*	0.08	*	0.06	*	0.07	*
sensory	0.48	0.39	*	0.57	v	0.29	*	0.36	*	0.45	*
servo	0.81	0.86	v	0.80		0.94	v	0.65	*	0.86	v
stock	0.97	1.00	v	0.97	v	0.99		0.98		0.98	
triazines	0.48	0.45	*	0.47		0.44	*	0.25	*	0.27	*
veteran	0.43	0.31	*	0.41		0.25	*	0.35	*	0.23	*
wisconsin	0.29	0.04	*	0.04	*	0.04	*	0.35	v	0.15	*
W-D-L		*7/5/12*		*7/8/9*		*11/1/12*		*9/5/10*		*8/4/12*	

[4] C. Blake & C. Merz, UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science. http://www.ics.uci.edu/~mlearn/MLRepository.html

[5] E. Bauer and R.. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. Machine Learning, 36(1/2): pp. 525–536, 1999.

[6] H. Brighton, C. Mellish, Advances in Instance Selection for Instance-Based Learning Algorithms, Data Mining and Knowledge Discovery, 6, pp. 153–172, 2002.

[7] L. Bottou and V. Vapnik, Local learning algorithm, Neural Computation, vol. 4, no. 6, pp. 888-901, 1992.

[8] L. Breiman, Bagging Predictors. Machine Learning, 24(3), pp. 123-140. Kluwer Academic Publishers, 1996.

[9] S. Cohen and N. Intrator. Automatic Model Selection in a Hybrid Perceptron/ Radial Network. In Multiple Classifier Systems. 2nd International Workshop, MCS 2001, pp 349–358.

[10] Duffy, N. Helmbold, D., Boosting Methods for Regression, Machine Learning, 47, (2002) 153–200.

[11] C. John and L. Trigg, K*: An Instance- based Learner Using an Entropic Distance Measure", Proc. of the 12th International Conference on ML, pp. 108-114, 1995.

[12] E. Frank, M. Hall and B. Pfahringer, Locally weighted naive Bayes. Proc. of the 19th Conference on Uncertainty in AI. Acapulco, Mexico. Morgan Kaufmann, 2003.

[13] Y. Freund and R. Schapire, Experiments with a New Boosting Algorithm, Proc. ICML'96, pp. 148-156, 1996

[14] Friedman J., "Stochastic Gradient Boosting," Computational Statistics and Data Analysis 38 (2002) pp. 367-378.

[15] W. Iba, & P. Langley, Induction of one-level decision trees. Proc. of the Ninth International Machine Learning Conference (1992). Aberdeen, Scotland: Morgan Kaufmann.

[16] E.M. Kleinberg. A Mathematically Rigorous Foundation for Supervised Learning. volume 1857 of Lecture Notes in Computer Science, pp. 67–76. Springer-Verlag, 2000.

[17] Kohavi R., The Power of Decision Tables. In Proc European Conference on Machine Learning (1995) pp. 174 - 189.

[18] Loader, C., Local Regression and Likelihood. Springer, New York, (1999).

[19] P. Melville and R. Mooney, Constructing Diverse Classifier Ensembles using Artificial Training Examples, Proc. of the IJCAI-2003, pp. 505-510, Acapulco, Mexico, August 2003

[20] Nadeau, C., Bengio, Y., Inference for the Generalization Error. Machine Learning, 52 (2003) pp. 239-281.

[21] Ting, K. M., Witten, I. H.: Stacking Bagged and Dagged Models. In: Fourteenth international Conference on Machine Learning, San Francisco, CA, pp. 367-375, 1997.

[22] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.

[23] G. I. Webb, MultiBoosting: A Technique for Combining Boosting and Wagging, Machine Learning, 40, pp. 159–196, Kluwer Academic Publishers, 2000.

[24] D. Wilson, and T. Martinez, Reduction Techniques for Instance-Based Learning Algorithms, Machine Learning, 38, pp. 257–286, 2000.

[25] I. Witten, and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Mateo, CA, 2000.