Improving on Bagging with Input Smearing

Eibe Frank and Bernhard Pfahringer

Department of Computer Science
University of Waikato
Hamilton, New Zealand
{eibe, bernhard}@cs.waikato.ac.nz

Abstract. Bagging is an ensemble learning method that has proved to be a useful tool in the arsenal of machine learning practitioners. Commonly applied in conjunction with decision tree learners to build an ensemble of decision trees, it often leads to reduced errors in the predictions when compared to using a single tree. A single tree is built from a training set of size N. Bagging is based on the idea that, ideally, we would like to eliminate the variance due to a particular training set by combining trees built from all training sets of size N. However, in practice, only one training set is available, and bagging simulates this platonic method by sampling with replacement from the original training data to form new training sets. In this paper we pursue the idea of sampling from a kernel density estimator of the underlying distribution to form new training sets, in addition to sampling from the data itself. This can be viewed as "smearing out" the resampled training data to generate new datasets, and the amount of "smear" is controlled by a parameter. We show that the resulting method, called "input smearing", can lead to improved results when compared to bagging. We present results for both classification and regression problems.

1 Introduction

Ensembles of multiple prediction models, generated by repeatedly applying a base learning algorithm, have been shown to often improve predictive performance when compared to applying the base learning algorithm by itself. Ensemble generation methods differ in the processes used for generating multiple different base models from the same set of data. One possibility is to modify the input to the base learner in different ways so that different models are generated. This can be done by resampling or reweighting instances [1, 2], by sampling from the set of attributes [3], by generating artificial data [4], or by flipping the class labels [5]. A different possibility is to modify the base learner so that different models can be generated from the same data. This is typically done by turning the base learner into a randomized version of itself, e.g. by choosing randomly among the best splits at each node of a decision tree [6]. This paper investigates an ensemble learning method that belongs to the former category. We call it "input smearing" because we randomly modify the attribute values of an instance, thus smearing it out in instance space. We show that, when combined with bagging, this method can improve on using bagging alone, if the amount of smearing is chosen appropriately for each dataset. We show that this can be reliably achieved using internal cross-validation, and present results for classification and regression problems.

The motivation for using input smearing is that it may be possible to increase the diversity of the ensemble by modifying the input even more than bagging does. The aim of ensemble generation is a set of classifiers such that they are simultaneously as different to each other as possible while remaining as accurate as possible when viewed individually. Independence—or "diversity"—is important because ensemble learning can only improve on individual classifiers when their errors are not correlated. Obviously these two aims—maximum accuracy of the individual predictors and minimum correlation of erroneous predictions—conflict with each other, as two perfect classifiers would be rather similar, and two maximally different classifiers could not at the same time both be very accurate. This necessary balance between diversity and accuracy has been investigated in various papers including [7], which among other findings reported that bagged trees are usually much more uniform than boosted trees. But it was also found that increasing levels of noise lead to much more diverse bagged trees, and that bagging starts to outperform boosted trees for high noise levels.

Commonly the attribute values of the examples are not modified in any way in the ensemble generation process. One exception to this "rule" is called "output smearing" [5], which modifies the class labels of examples by adding a controlled amount of noise. In this paper we investigate the complimentary process of applying "smearing" not to the output variable, but to the input variables. Initial experiments showed that smearing alone could not consistently improve on bagging. This lead us to the idea of combining smearing and bagging, by smearing the subsamples involved in the bagging process. The amount of smearing enables us to control the diversity in the ensemble, and more smearing increases the diversity compared to bagging alone. However, more smearing also means that the individual ensemble members become less accurate. Our results show that cross-validation can be used to reliably determine an appropriate amount of smearing.

This paper is structured as follows. In Section 2 we discuss previous work on using artificial data in machine learning and explain the process of "input smearing" in detail. Section 3 presents our empirical results on classification and regression datasets, and Section 4 discussed related work. Section 5 summarizes our findings and points out directions for future work.

2 Using artificial training data

One way of viewing input smearing is that artificial examples are generated to aid the learning process. Generating meaningful artificial examples may seem straightforward, but it is actually not that simple. The main issue is the problem of generating meaningful class values or labels for fully artificially generated examples. Theoretically, if the full joint distribution of all attributes including the class attribute were known, examples could simply be drawn according to this full joint distribution, and their class labels would automatically be meaningful. Unfortunately this distribution is not available for practical learning problems.

This "labelling" problem is the most likely explanation as to why artificially generated training examples are rarely used. One exception is the approach reported in [8]. This work is actually not concerned with improving the predictive accuracy of an ensemble, but instead tries to generate a single tree with similar performance to an ensemble generated by an ensemble learning method. The aim is to have a comprehensible model with a similar predictive performance as the original ensemble. The method generates artificial examples and uses the induced ensemble to label the new examples. It has been shown that large sets of artificial examples can lead to a large single tree capable of approximating the predictive behaviour of the original ensemble.

Another exception is the work presented in [9], which investigates the problem of very skewed class distributions in inductive learning. One common idea is oversampling of the minority class to even out the class distribution, and [9] takes that one step further by generating new artificial examples for the minority class. This is done by randomly selecting a pair of examples from the minority class, and then choosing an arbitrary point along the line connecting the original pair. Furthermore the method makes sure that there is no example from the majority class closer to the new point than any of the minority examples. The main drawback of this method is that it is very conservative, and that it relies on nearest neighbour computation, which is of questionable value in higher-dimensional settings. In the case of highly skewed class distributions such conservativeness might be appropriate, but in more general settings it is rather limiting.

Finally, the Decorate algorithm [4] creates artificial examples adaptively as an ensemble of classifiers is being built. It assigns labels to these examples by choosing those labels that the existing ensemble is least likely to predict. It is currently unclear why this method works well in practice [4].

We have chosen a very simple method for generating artificial data to improve ensemble learning. Our method addresses the labelling problem in a similar fashion as what has been done for skewed class distributions, taking the original data as the starting point. However, we then simply modify the attribute values of a chosen instance by adding random attribute noise. The method we present here combines bagging with this modification for generating artificial data. More specifically, as in bagging, training examples are drawn with replacement from the original training set until we have a new training set that has the same size as the original data. The next step is new: instead of using this new dataset as the input for the base learning algorithm, we modify it further by perturbing the attribute values of all instances by a small amount (excluding the class attribute). This perturbed data is then fed into the base learning algorithm to generate one ensemble member. The same process is repeated with different random number seeds to generate different datasets, and thus different ensemble members.

This method is very simple and applicable to both classification and regression problems (because the dependent variable is not modified), but we have not yet specified how exactly the modification of the original instances is performed. In this paper we make one simplification: we restrict our attention to datasets with numeric attributes. Although the process of input smearing can be applied to nominal data as well (by changing a given attribute value with a certain probability to a different value) it can be more naturally applied with numeric attributes because they imply a notion of distance. To modify the numeric attribute values of an instance we simply add Gaussian noise to them. We take the variance of an attribute into account by scaling the amount of noise based on this variance (using Gaussian noise with the same variance for every attribute would obviously not work, given that attributes in practical datasets are often on different scales). More specifically, we transform an attribute value $a_{original}$ into a smeared value $a_{smeared}$ based on

$$a_{smeared} = a_{original} + p * N(0, \sigma_a),$$

where σ_a is the estimated global standard deviation for attribute $a_{original}$, and p is a user-specifiable parameter that determines the amount of noise to add. The original class value is left intact.

Usually the value of the smearing parameter is greater than zero but the optimum value depends on the data. Cross-validation is an obvious method for finding an appropriate value in a purely data-dependent fashion, and as we will see in the next section, it chooses quite different values depending on the dataset. In the experiments reported below we employed internal cross-validation in conjunction with a simple grid search, evaluating different values for p in a range of values that is explored in equal-size steps. As it turns out, there are datasets where no smearing ($p = 0$) is required to achieve maximum accuracy.

Another view of input smearing is that we employ a kernel density estimate of the data, placing a Gaussian kernel on every training instance, and then sample from this estimate of the joint distribution of the attribute values. We choose an appropriate kernel width by evaluating the cross-validated accuracy of the resulting ensemble (and combine the smearing process with bagging) but an alternative approach would be to first fit a kernel density estimate to the data by some regularized likelihood method, and then use the resulting kernel widths to generate a smeared ensemble. A potential drawback of our method is that the amount of noise is fixed for every attribute (although it is adjusted based on the attributes' scales). It may be that performance can be improved further by introducing a smearing parameter for every attribute and tuning those smearing parameters individually. Using an approach based on kernel density estimation may make this computationally feasible.

Note that, compared to using bagging alone, the computational complexity remains unchanged. Modifying the attribute values can be done in time linear in the number of attributes and instances. The cross-validation-based grid search for the optimal smearing parameter increases the runtime by a large constant factor but it may be possible to improve on this using a more sophisticated search strategy in place of grid search.

Figure 1 shows the pseudo code for building an ensemble using input smearing. The process for making a prediction (as well as the type of base learner

```
method inputSmearing(Dataset D, Ensemble size n, Smearing parameter p)

  compute standard deviation σ_a for each attribute a in the data
  repeat n times
    sample dataset R of size |D| from D using sampling with replacement
    S = ∅
    for each instance x in R
      for each attribute a in R
        x'_a = x_a + p * N(0, σ_a)
      add x' to S
    apply based learner to S and add resulting model to committee
```

Fig. 1. Algorithm for generating an ensemble using input smearing.

employed) depends on whether we want to tackle a regression problem or a classification problem. In the case of regression we simply average the predicted numeric values from the base models to derive an ensemble prediction. In the case of classification, we average the class probability estimates obtained from the base models, and predict the class for which the average probability is maximum. (In the experiments reported in the next section we use exactly the same method for bagging.)

3 Experimental Results

In this section we conduct experiments on both classification and regression problems to compare input smearing to bagging. As a baseline we also present results for the underlying base learning algorithm when used to produce a single model. The main parameter needed for input smearing, the noise threshold p, is set automatically using cross-validation, as explained above. We will see that this automated process reliably chooses appropriate values. Consequently input smearing competes well with bagging.

3.1 Classification

Our comparison is based on 22 classification problems from the UCI repository [10]. We selected those problems that exhibit only numeric attributes. Missing values (present in one attribute of one of the 22 datasets, the breast-w data) are not modified by our implementation of smearing.

Input smearing was applied in conjunction with unpruned decision trees built using the fast REPTree decision tree learner in Weka. REPTree is a simple tree learner that uses the information gain heuristic to choose an attribute and a binary split on numeric attributes. It avoids repeated re-sorting at the nodes of the tree, and is thus faster than C4.5. We performed ten iterations to build ten ensemble members. Internal 5-fold cross-validation was used to choose an appropriate parameter value for the smearing parameter p for each training set. To identify a good parameter value we used a simple grid search that evaluated

Dataset	Input smearing	Bagging	Unpruned tree	C4.5	Parameter value
balance-scale	85.8±3.6	81.2±3.8•	78.5±4.4•	78.1±4.1•	0.27±0.05
breast-w	96.0±2.1	95.5±2.0	93.7±2.3•	94.9±2.3	0.19±0.10
ecoli	84.7±5.6	83.1±5.4	82.0±5.5	82.8±5.3	0.22±0.08
glass	74.9±9.3	76.5±9.1	69.7±8.9	68.1±8.2•	0.06±0.07
hayes-roth	81.1±9.3	80.7±9.6	84.1±9.5	79.0±8.4	0.10±0.12
heart-statlog	80.8±6.5	78.8±6.6	74.9±7.3•	78.6±7.1	0.19±0.10
ionosphere	91.6±5.4	91.0±4.6	89.6±5.0	90.0±5.0	0.15±0.09
iris	96.1±5.0	95.3±5.5	94.3±5.6	95.4±5.4	0.17±0.10
letter	92.1±0.7	91.9±0.7	87.9±0.7•	88.1±0.8•	0.14±0.04
liver-disorders	69.0±7.0	69.8±7.7	64.5±8.1	66.2±7.8	0.08±0.08
mfeat	77.6±2.6	73.5±2.6•	68.5±3.1•	71.4±2.7•	0.28±0.03
optdigits	95.9±0.9	94.9±1.1•	90.8±1.2•	90.6±1.1•	0.29±0.02
page-blocks	97.2±0.6	97.3±0.6	96.8±0.6•	97.0±0.7	0.02±0.03
pendigits	98.4±0.4	98.1±0.5•	96.4±0.5•	96.5±0.6•	0.16±0.04
pima-diabetes	75.3±4.4	75.0±4.8	71.1±4.6•	73.8±5.3	0.18±0.09
segment	97.4±1.0	97.5±1.1	96.6±1.3	96.8±1.2	0.01±0.02
sonar	81.5±8.5	81.3±8.2	77.5±9.0	74.3±9.5	0.14±0.09
spambase	94.6±1.0	94.6±1.0	92.8±1.3•	92.7±1.2•	0.00±0.00
spectf	88.5±5.1	89.3±4.7	86.0±5.3	84.8±5.8	0.03±0.05
vehicle	74.9±4.1	75.0±4.5	72.4±4.5	73.4±4.2	0.15±0.09
waveform	82.6±1.8	81.8±1.9	75.3±2.0•	75.3±1.9•	0.25±0.06
wine	95.5±4.5	95.4±4.7	93.9±6.0	92.7±6.6	0.17±0.11

• denotes a statistically significant degradation compared to input smearing

Table 1. Input smearing applied to classification problems.

values 0, 0.05, 0.1, 0.15, 0.2, 0.25, and 0.3. This automated parameter estimation adds a large computational overhead, but prevents the user from bad choices, and might also provide valuable insights into both the data as well as the example generation process.

Table 1 lists the estimated classification accuracy in percent correct, obtained as averages over 100 runs of the stratified hold-out method. In each run 90% of the data was used for training and 10% was used for testing. The corrected resampled t-test [11] was used to perform pairwise comparison between algorithms.

Apart from the results for input smearing, the table also lists results for bagging, unpruned decision trees generated using REPTree, and pruned C4.5 trees. It also shows the average parameter value chosen by the internal cross-validation, and the standard deviation for each of the statistics across the 100 runs. Bagging was applied in conjunction with the same base learner and the same number of iterations as input smearing.

Analyzing the results of Table 1, we see that "input smearing" can improve the predictive accuracy of single trees for about half of all the datasets, and also significantly outperforms bagging four times. More importantly, it never performs significantly worse than any of the other algorithms. The average values chosen for p vary from 0 up to 0.29. Given that the latter value is quite close to the upper boundary of the range that we searched in our experiments, it may be possible that larger values would result in further improvements for the datasets where such a large value was chosen. For all datasets except one a non-zero parameter value is chosen, with *spambase* being the sole exception. We can only speculate why smearing does not work for this dataset. Most likely the noise generation process is not appropriate for this dataset, which consists solely of counts of word occurrences. These are non-negative and generally follow a power law [12]. A more specialized distribution like the Poisson distribution may be more appropriate for smearing in this case. Alternatively, the input variables could also be preprocessed by a logarithmic transformation, which is common practice in statistics for dealing with counts.

One method for analysing the behaviour of a modelling technique is the so-called bias-variance decomposition (see e.g. [13]), which tries to explain the total prediction error as the sum of three different sources of error: bias (i.e. how close is the average model to the actual function?), variance (i.e. how much do the models' guesses "bounce around"?), and intrinsic noise (the Bayes error).

Using the specific approach described in [13], a bias-variance decomposition was computed for all the classification datasets used above for both input smearing and bagging. We would expect that input smearing exhibits a higher bias than bagging on average, as it modifies the input distribution of all attributes. To verify this hypothesis, the relative contribution of bias compared to variance was computed for both methods on each dataset. More specifically, we computed

$$relativeBias = bias/(bias + variance).$$

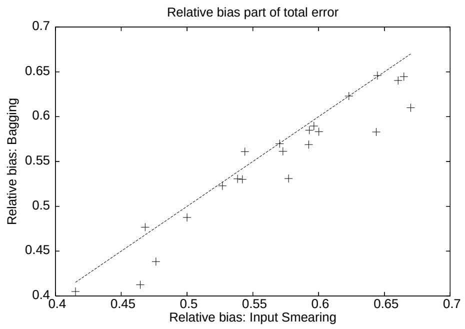

Fig. 2. Relative bias: smearing vs. bagging

In Figure 2 we plot the relative bias of bagging over the relative bias of input smearing. Points below the diagonal indicate cases where smearing exhibits a higher relative bias than bagging. This is the case for most datasets. Some points are very close to the diagonal or exactly on the diagonal. One of these points represents the spambase dataset, where the threshold value of 0.0 effectively turns input smearing into bagging.

3.2 Regression

Classification is not the only application of input smearing. In the following we investigate its performance when applied in conjunction with a state-of-the-art tree learner for regression problems. This comparison is based on a collection of 23 regression problems [14] that are routinely used as benchmarks for evaluating regression algorithms.

We employed the same evaluation framework as in the classification case: ensembles are of size ten and random train/test splits of 90%/10% are repeated

Dataset	Input smearing	Bagging	Pruned model trees	Unpruned model trees	Parameter value
2dplanes	22.9±0.3	23.2±0.3 •	22.7±0.3 ∘	23.3±0.3 •	0.30±0.00
ailerons	39.2±1.3	39.2±1.3	39.9±1.2 •	41.1±1.3 •	0.00±0.00
bank32nh	68.5±2.3	69.0±2.4 •	67.0±2.5 ∘	74.5±2.8 •	0.19±0.07
bank8FM	19.4±0.7	19.5±0.7 •	20.0±0.7 •	20.4±0.7 •	0.06±0.02
cal-housing	44.0±1.6	44.0±1.6	48.5±2.1 •	46.4±1.8 •	0.00±0.00
cpu-act	13.2±1.0	13.8±1.2 •	14.7±1.7 •	15.3±2.5 •	0.14±0.04
cpu-small	16.1±1.3	16.2±1.4	17.4±2.0 •	17.7±2.3 •	0.07±0.03
delta-ailerons	53.2±2.1	53.2±2.2	54.4±2.1 •	54.5±2.2 •	0.06±0.03
delta-elevators	59.8±1.4	60.0±1.5 •	60.1±1.4 •	61.0±1.6 •	0.18±0.05
diabetes-numeric	94.4±39.4	94.9±42.0	98.5±49.5•	96.8±44.7	0.13±0.10
elevators	34.1±6.1	33.4±1.2	32.1±1.2 ∘	35.5±1.3 •	0.01±0.02
fried	25.9±0.4	26.1±0.4 •	27.8±0.5 •	28.1±0.5 •	0.05±0.01
house-16H	62.6±4.6	62.0±4.5 ∘	68.0±3.2 •	66.7±3.6 •	0.01±0.02
house-8L	57.7±7.0	57.7±7.0	59.7±3.5 •	59.7±3.6 •	0.00±0.01
kin8nm	53.7±1.6	54.4±1.8 •	60.9±2.1 •	59.9±2.1 •	0.10±0.03
machine-cpu	36.0±12.3	35.7±11.8	40.5±18.5•	36.0±14.1	0.14±0.12
pol	13.6±1.0	13.5±1.0	15.2±1.2 •	14.8±1.1 •	0.02±0.02
puma32H	26.0±0.8	26.1±0.8 •	27.1±0.8 •	27.5±0.9 •	0.05±0.01
puma8NH	56.9±1.5	57.7±3.9 •	57.0±1.6	59.1±1.8 •	0.12±0.03
pyrim	58.5±21.3	57.3±21.7	64.9±26.2•	58.8±25.2	0.09±0.11
stock	13.9±1.9	14.2±2.5	14.4±2.5 •	14.3±2.6 •	0.07±0.03
triazines	79.8±13.9	79.6±13.9	84.0±17.4•	81.4±17.8	0.00±0.03
wisconsin	94.4±11.0	95.1±10.4	98.1±12.4•	98.7±12.8•	0.19±0.10

•/∘ denote a statistically significant degradation/improvement wrt input smearing.

Table 2. Input smearing applied to regression problems.

100 times (in this case without applying stratification, of course). Performance is measured based on the root relative squared error. A value of zero would indicate perfect prediction, and values larger than 100 indicate performance worse than simply predicting the global mean of the class-values obtained from the training data. Unpruned M5 model trees [15], generated using the M5' model tree learner in Weka [16], were used as the base learner for input smearing and bagging, and we compare to single unpruned and pruned M5 model trees. Again, the noise parameter p was determined automatically by internal five-fold cross-validation using a grid search on the values 0, 0.05, 0.1, 0.15, 0.2, 0.25, and 0.3.

Again, analyzing the results of Table 2, we see that input smearing almost always improves prediction over single model trees. However, it is significantly worse than a single pruned tree on three datasets. Compared to bagging, significant improvements are achieved 39% of the time, with only one significant loss. As with classification, the average smearing parameter values chosen by cross-validation are well below 0.3 in most cases, except for one dataset (2dplanes), where an even larger parameter value may have been chosen if it had been available. Again there is one dataset where zero is chosen consistently. As we are not familiar with the actual meaning of the attributes in this dataset (ailerons), we cannot make such strong claims as for the spambase dataset, but at least one third of all attributes in this dataset again appear to be based on counts, and another third of all attributes is almost constant, i.e. clearly not normally distributed either. Inspecting the attribute distributions for the only other two datasets with smearing parameter values close to 0 (house-8L and triazines) reveals that in both datasets a majority of attributes again is not normally distributed.

4 Related Work

In this section we discuss related work but restrict our attention to ensemble generation methods. We do not repeat the discussion of methods that have already been discussed in Section 2. In terms of ensemble generating methods we only list and discuss methods that modify the data in some way.

- Bagging [1] has its origin in bootstrap sampling in statistics, which produces robust estimates of population statistics by trying to simulate averaging over all possible datasets of a given size. Sets are generated by sampling with replacement. Bagging can reduce the variance of a learner, but it cannot reduce its bias.
- Dagging [17] is an alternative to bagging that combines classifiers induced on disjoint subsets of the data. It is especially appropriate when either the data originally comes from disjoint sources, or when data is plentiful, i.e. when the learning algorithm has reached the plateau on the learning curve. Like bagging, dagging could potentially be combined with input smearing to increase diversity.
- Output smearing [5] adds a controlled amount of noise to the output or dependent attribute only. The empirical results in [5] show that is works surprisingly well as an ensemble generator. An interesting question for future work is whether input and output smearing can be combined successfully.
- Random feature subsets [3, 18] work particularly well for so-called stable algorithms like the nearest neighbour classifier, where bagging does not achieve much improvement. Random feature projections [19] may have some potential in this setting as well.

5 Conclusions

We have described a new method for ensemble generation, called input smearing, that works by sampling from a kernel density estimator of the underlying distribution to form new training sets, in addition to resampling from the data itself like in bagging. Our experimental results show that it is possible to obtain significant improvements in predictive accuracy when applying input smearing instead of bagging (which can be viewed as a special case of input smearing in our implementation). Our results also show that it is possible to use cross-validation to determine an appropriate amount of smearing on a per-dataset basis.

Input smearing using Gaussian noise is not necessarily the best choice. An avenue for future work is to investigate the effect of other distributions in input smearing, and to choose an appropriate distribution based on the data. Such a more sophisticated approach should also make it possible to generalize input smearing to other attribute types and structured input.

References

1. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
2. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Thirteenth Int Conf on Machine Learning. (1996) 148–156
3. Bay, S.D.: Nearest neighbor classification from multiple feature subsets. Intelligent Data Analysis **3** (1999) 191–209
4. Melville, P., Mooney, R.J.: Creating diversity in ensembles using artificial data. Journal of Information Fusion (Special Issue on Diversity in Multiple Classifier Systems) **6/1** (2004) 99–111
5. Breiman, L.: Randomizing outputs to increase prediction accuracy. Machine Learning **40** (2000) 229–242
6. Breiman, L.: Random forests. Machine Learning **45** (2001) 5–32
7. T.Dietterich: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning **40** (2000) 139–157
8. Domingos, P.: Knowledge acquisition from examples via multiple models. In: Proc. 14th Int Conf on Machine Learning. (1997) 98–106
9. N.V. Chawla, K.W.Bowyer, L., W.P.Kegelmeyer: Smote: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research **16** (2002) 321–357
10. D.J. Newman, S. Hettich, C.B., Merz, C.: UCI repository of machine learning databases (1998)
11. C.Nadeau, Y.Bengio: Inference for the generalization error. Machine Learning **52** (2003) 239–281
12. Rennie, J.D.M., Shih, L., Teevan, J., Karger, D.R.: Tackling the poor assumptions of naive Bayes text classifiers. In: Proc Twentieth Int Conf on Machine Learning, AAAI Press (2003) 616–623
13. Kohavi, R., Wolpert, D.H.: Bias plus variance decomposition for zero-one loss functions. In: Proc Thirteenth Int Conf on Machine Learning. (1996) 275–283
14. Torgo, L.: Regression datasets (2005) [www.liacc.up.pt/∼ltorgo/Regression].
15. Quinlan, J.R.: Learning with Continuous Classes. In: Proc 5th Australian Joint Conf on Artificial Intelligence, World Scientific (1992) 343–348
16. Wang, Y., Witten, I.: Inducing model trees for continuous classes. In: Proc of Poster Papers, European Conf on Machine Learning. (1997)
17. Ting, K., Witten., I.: Stacking bagged and dagged models. In: Fourteenth Int Conf on Machine Learning (ICML07). (1997) 367–375
18. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 832–844
19. Achlioptas, D.: Database-friendly random projections. In: Twentieth ACM Symposium on Principles of Database Systems. (2001) 274–281