An Evaluation of Grading Classifiers

Alexander K. Seewald and Johannes Fürnkranz

Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A-1010 Wien
{alexsee, juffi}@oefai.at

Abstract. In this paper, we discuss grading, a meta-classification technique that tries to identify and correct incorrect predictions at the base level. While stacking uses the predictions of the base classifiers as meta-level attributes, we use "graded" predictions (i.e., predictions that have been marked as correct or incorrect) as meta-level classes. For each base classifier, one meta classifier is learned whose task is to predict when the base classifier will err. Hence, just like stacking may be viewed as a generalization of voting, grading may be viewed as a generalization of selection by cross-validation and therefore fills a conceptual gap in the space of meta-classification schemes. Our experimental evaluation shows that this technique results in a performance gain that is quite comparable to that achieved by stacking, while both, grading and stacking outperform their simpler counter-parts voting and selection by cross-validation.

1 Introduction

When faced with the decision "Which algorithm will be most accurate on my classification problem?", the predominant approach is to estimate the accuracy of the candidate algorithms on the problem and select the one that appears to be most accurate. [13] has investigated this approach in a small study with three learning algorithms on five UCI datasets. His conclusions are that on the one hand this procedure is on average better than working with a single learning algorithm, but, on the other hand, the cross-validation procedure often picks the wrong base algorithm on individual problems. This problem is expected to become more severe with an increasing number of classifiers.

As a cross-validation basically computes a prediction for each example in the training set, it was soon realized that this information could be used in more elaborate ways than simply counting the number of correct and incorrect predictions. One such meta-classification scheme is the family of *stacking* algorithms [19]. The basic idea of stacking is to use the predictions of the original classifiers as attributes in a new training set that keeps the original class labels.

In this paper, we investigate another technique, which we call *grading*. The basic idea is to learn to predict for each of the original learning algorithms whether its prediction for a particular example is correct or not. We therefore train one classifier for each of the original learning algorithms on a training set that consists of the original examples with class labels that encode whether

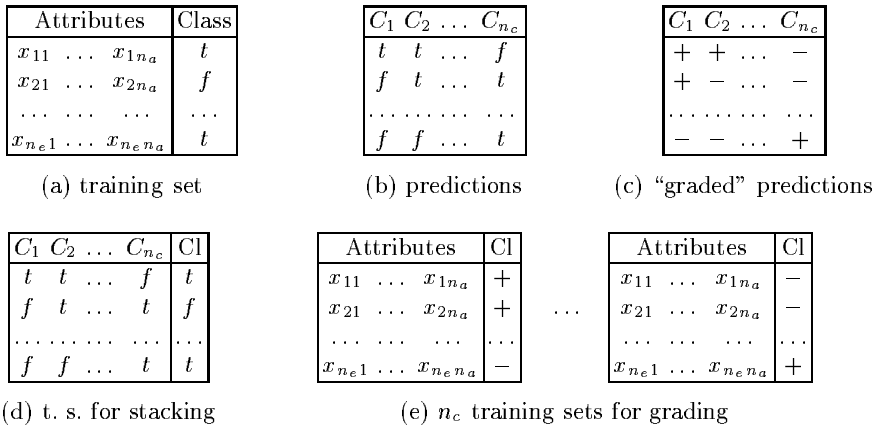

Fig. 1. Illustration of stacking and grading. In this hypothetical situation, n_c classifiers are tried on a problem with n_a attributes, n_e examples and $n_l = 2$ classes (t, f).

the prediction of this learner was correct on this particular example. Hence—in contrast to stacking—we leave the original examples unchanged, but instead modify the class labels.

The idea of grading is not entirely new. [9] and [8] independently introduce algorithms based on the same basic idea, but provide only a preliminary evaluation of the approach. We will describe this idea in more detail and compare it to cross-validation, stacking, and a voting technique.

2 Grading

Figure 1(a) shows a hypothetical learning problem with n_e training examples, each of them encoded using n_a attributes x_{ij} and a class label cl_i. In our example, the number of different class labels n_l is 2 (the values t and f) but this is no principal restriction. We are now assuming that we have n_c *base classifiers* C_k, which were evaluated using some cross-validation scheme. As cross-validation ensures that each example is used as a test example exactly once, we have obtained one prediction for each classifier and for each training example (Figure 1(b)).

A straight-forward use of this prediction matrix is to let each classifier vote for a class, and predict the class that receives the most votes. Stacking, as originally described by [19], makes a more elaborate use of the prediction matrix. It adds the original class labels cl_i to it and uses this new data set—shown in Figure 1(d)—for training another classifier[1]. Examples are classified by submitting them to each base classifier (trained on the entire training set) and using

[1] In fact, we followed [17] in using probability distributions for stacking. Instead of merely adding the classes that are considered to be most likely by each base classifier, they suggest to add the entire n_l-dimensional class probability vector P_{ik}, yielding a meta dataset with $n_l \times n_c$ attributes instead of only n_c for conventional stacking.

the predicted labels as input for the meta classifier learned from the prediction matrix. Its prediction is then used as the final prediction for the example.

By providing the true class labels as the target function, stacking provides its meta-learner with indirect feedback about the correctness of its base classifiers. This feedback can be made more explicit. Figure 1(c) shows an evaluation of the base classifiers' predictions. Each entry c_{ik} in the prediction matrix is compared to the corresponding label cl_i. Correct predictions ($c_{ik} = cl_i$) are "graded" with $+$, incorrect predictions with $-$.

Grading makes use of these graded predictions for training a set of meta classifiers that learn to predict when the base classifier is correct. The training set for each of these meta classifiers is constructed using the graded predictions of the corresponding base classifier as new class labels for the original attributes. Thus we have n_c two-class training sets (classes $+$ and $-$), one for each base classifier (Figure 1(e)). We now train n_c level 1 classifiers, everyone of which gets exactly one of these training sets, based on the assumption that different base classifiers make different errors. Thus, every one of these level 1 classifiers tries to predict when its base classifier will err.

Note that the proportion of negatively graded examples in these datasets is simply the error rate of the corresponding base classifier as estimated by the cross-validation procedure. Hence, while selection by cross-validation [13] simply picks the classifier corresponding to the dataset with the fewest examples of class $-$ as the classifier to be used for all test examples, grading tries to make this decision for each example separately by focussing on those base classifiers that are predicted to be correct on this example. In this sense, grading may be viewed as a generalization of selection by cross-validation.

At classification time, each base classifier makes a prediction for the current example. The final prediction is derived from the predictions of those base classifiers that are predicted to be correct by the meta-classification schemes. Conflicts (several classifiers with different base-level predictions are predicted to be correct) may be resolved by voting or by making use of the confidence estimates of the base classifiers.

In our implementation, the confidence[2] of the level 1 classifier will be summed per class and afterwards normalized to yield a proper class probability distribution. In the rare case that no base classifier is predicted to be correct, all base classifiers are used with ($1 - \mathit{confidence}$) as the new confidence, thus preferring those base classifiers for which the level 1 classifier is more unsure of its decision. The most probable class from the final class distribution is chosen as the final prediction. Ties are cut by choosing among all most probable classes the class which occurs more frequently in the training data.

More formally, let p_{ikl} be the class probability calculated by base classifier k for class l and example i. For simplification, we write P_{ik} to mean $(p_{ik1}, p_{ik2}, \ldots p_{ikn_l})$, i.e., the vector of all class probabilities for example i and classifier k. The predic-

[2] We measure confidence with the meta classifiers' estimate of the probability $p(+)$ that the example is classified correctly by the corresponding base classifier. Since meta datasets as defined are two-class, confidence for $-$ is $p(-) = 1 - p(+)$.

tion of the base classifier k for example i is the class l with maximum probability p_{ikl}, more formally $c_{ik} = \arg\max_l \{p_{ikl}\}$.

Grading then constructs n_c training sets, one for each base classifier k, by adding the graded predictions g_{ik} as the new class information to the original dataset (g_{ik} is 1 if the base classifier k's prediction for example i was correct (graded +) and 0 otherwise). $prMeta_{ik}$ is the probability that base classifier k will correctly predict the class of example i as estimated by meta classifier k.

From this information we compute the final probability estimate for class l and example i. In case at least one meta classifier grades its base classifier as + (i.e., $prMeta_{ik} > 0.5$), we use the following formula:[3]

$$prGrading_{il} = \sum \{prMeta_{ik} | c_{ik} = l \wedge prMeta_{ik} > 0.5\}$$

Otherwise, if no base classifiers are presumed correct by the meta classifiers, we use all base classifiers in our voting. The class with highest probability is then chosen as the final prediction.

3 Empirical Evaluation

In this section, we compare our implementation of grading to stacking, voting, and selection by cross-validation. We implemented Grading in Java within the Waikato Environment for Knowledge Analysis (WEKA).[4] All other algorithms at the base and meta-level were already available within WEKA.

For an empirical evaluation we chose twenty-six datasets from the UCI Machine Learning Repository [2]. The datasets were selected arbitrarily before the start of the experiments and include both two-class and multi-class problems with up to 2310 examples. All reported accuracy estimates are the average of ten ten-fold stratified cross validations, except when stated otherwise.

We evaluated each meta-classification scheme using the following six base learners, which were chosen to cover a variety of different biases.

- DecisionTable: a decision table learner
- J48: a Java port of C4.5 Release 8 [12]
- NaiveBayes: the Naive Bayes classifier using kernel density estimation
- KernelDensity: a simple kernel density classifier
- MultiLinearRegression: a multi-class learner which tries to separate each class from all other classes by linear regression (*multi-response linear regression*)
- KStar: the K* instance-based learner [5]

All algorithms are implemented in WEKA Release 3.1.8. They return a class probability distribution, i.e., they do not predict a single class, but give probability estimates for each possible class.

[3] Penalizing cases where the base classifier predicts class l but the meta classifier considers this prediction wrong ($c_{ik} = l \wedge prMeta_{ik} \leq 0.5$) did not work as well.

[4] The Java source code of WEKA has been made available at www.cs.waikato.ac.nz

Table 1. Accuracy (%) for all meta-classification schemes.

Dataset	Grading	X-Val	Stacking	Voting
audiology	83.36	77.61	76.02	84.56
autos	80.93	80.83	82.20	83.51
balance-scale	89.89	91.54	89.50	86.16
breast-cancer	73.99	71.64	72.06	74.86
breast-w	96.70	97.47	97.41	96.82
colic	84.38	84.48	84.78	85.08
credit-a	86.01	84.87	86.09	86.04
credit-g	75.64	75.48	76.17	75.23
diabetes	75.53	76.86	76.32	76.25
glass	74.35	74.44	76.45	75.70
heart-c	82.74	84.09	84.26	81.55
heart-h	83.64	85.78	85.14	83.16
heart-statlog	84.22	83.56	84.04	83.30

Dataset	Grading	X-Val	Stacking	Voting
hepatitis	83.42	83.03	83.29	82.77
ionosphere	91.85	91.34	92.82	92.42
iris	95.13	95.20	94.93	94.93
labor	93.68	90.35	91.58	93.86
lymph	83.45	81.69	80.20	84.05
primary-t.	49.47	49.23	42.63	46.02
segment	98.03	97.05	98.08	98.14
sonar	85.05	85.05	85.58	84.23
soybean	93.91	93.69	92.90	93.84
vehicle	74.46	73.90	79.89	72.91
vote	95.93	95.95	96.32	95.33
vowel	98.74	99.06	99.00	98.80
zoo	96.44	95.05	93.96	97.23
Avg	85.04	84.59	84.68	84.88

All base algorithms have their respective strengths and weaknesses and perform well on some datasets and badly on others. Judging by the average accuracy (a somewhat problematic measure, see below), KernelDensity and KStar seem to have the competitive edge. Since space restrictions prevent us from showing all the details, these and other experimental results can be found in [15]. On these base algorithms, we tested the following four meta-classification schemes:

- Grading is our implementation of the grading algorithms. It uses the instance-based classifier IBk with ten nearest neighbors as the meta-level classifier. Our implementation made use of the class probability distributions returned by the meta classifiers. IBk estimates the class probabilities with a Laplace-estimate of the proportion of the neighbors in each class. These estimates are then normalized to yield a proper probability distribution.[5]
- X-Val chooses the best base classifier on each fold by an internal ten-fold CV.
- Stacking is the stacking algorithm as implemented in WEKA, which follows [17]. It constructs the meta dataset by adding the entire predicted class probability distribution instead of only the most likely class. Following [17], we also used MultiLinearRegression as the level 1 learner.[6]
- Voting is a straight-forward adaptation of voting for distribution classifiers. Instead of giving its entire vote to the class it considers to be most likely, each classifier is allowed to split its vote according to its estimate of the class probability distribution for the example. It is mainly included as a benchmark of the performance that could be obtained without resorting to the expensive CV of every other algorithm.

[5] $confidence = p(+) = \frac{p+\frac{1}{n_e}}{k+\frac{2}{n_e}}$, n_e is training set size, p of k neighbors graded as $+$.

[6] Relatively global and smooth level-1 generalizers should perform well [19, 17].

Table 2. Significant wins/losses for the four meta-classification schemes against themselves and against all meta learning algorithms. The first number shows significant wins for the algorithm in the column, and the second number for the algorithm in the row.

	Grading	X-Val	Stacking	Voting
Grading	—	5/6	7/7	2/4
X-Val	6/5	—	6/3	9/7
Stacking	7/7	3/6	—	7/8
Voting	4/2	7/9	8/7	—
$\sum$ meta	17/14	15/21	21/17	18/19

	Grading	X-Val	Stacking	Voting
DecisionTable	24/0	22/0	24/0	26/0
J48	20/2	17/3	18/1	20/1
KernelDensity	21/1	18/2	20/2	23/2
KStar	19/0	17/4	17/4	19/1
MLR	17/3	12/2	13/7	14/5
NaiveBayes	13/5	14/1	14/5	17/7
$\sum$ base	114/11	100/12	106/19	119/16

As noted above, all algorithms made use of class probability distributions. This choice was partly motivated by the existing implementation of Stacking within WEKA, and partly because of the experimental evidence that shows that the use of class probability distributions gives a slightly better performance in combining classifiers with stacking [17] and ensemble methods [1]. We did some preliminary studies which seemed to indicate that this is also the case for Grading, but we did not yet attempt a thorough empirical verification of this matter.

Table 1 shows the accuracies for all meta-classification schemes w.r.t. each dataset. Not surprisingly, no individual algorithm is a clear winner over all datasets; each algorithm wins on some datasets and loses on others. On average, Grading seems to be slightly more accurate than Stacking (85.04% vs. 84.68%). Somewhat surprising is the performance of Voting: although it does not use the expensive predictions obtained from the internal cross-validation, it seems to perform no worse than the other algorithms which do use this information. Of course, a comparison of algorithms with their average accuracy over a selection of datasets has to be interpreted very cautiously because of the different baseline accuracies and variances on the different problems. In the following, we take a closer look at the performance differences.

Table 2 shows significant wins/losses of the meta-classification schemes versus themselves and versus the base classifiers. Significant differences were calculated by a t-test with 99% significance level. Positive and negative differences between classifiers were counted as wins and losses respectively. Among the meta-classification schemes, Stacking and Grading are best with Grading slightly behind ($wins - losses = 4$ vs. 3) while X-Val and Voting lag far behind (more losses than wins). Although this evaluation shows that—contrary to the average performance discussed above—Voting does not get close to Grading and Stacking, it nevertheless outperforms X-Val. Hence, adding up the predicted class probabilities of different classifiers seems to be a better decision procedure than selecting the best algorithm by cross-validation, which is consistent with other results on ensembles of classifiers [6]. Table 2 shows that all meta-classification schemes are almost always an improvement over the six base learners. Measured in terms of the differences between wins and losses, Voting and Grading are both on first

place ($wins - losses = 103$). Interestingly, Stacking's performance seems to be the worst: it has fewer wins than Voting and Grading, and the most losses of all meta-classification schemes. Still, meta-classification schemes are clearly an improvement over base classifiers in any case.

Grading is not worse than any base classifiers on seventeen datasets and better than all base classifiers on two datasets, among them the largest (*segment*). Nine times, Grading is worse than at least one classifier, two times it is worse than two, and never worse than three or more. Our meta-classification scheme can thus be considered an improvement over the base learners in the majority of cases.

However, while our results show no significant differences across the datasets we studied, they also seem to indicate that the performance of stacking and grading varies considerably across different domains. For example, our results seemed to indicate that Grading seems to be better than Stacking on smaller datasets, while it performs worse on on larger datasets.

In summary, the overall performance of Grading is comparable to that of Stacking. Although Grading has a higher average performance over all datasets and has more significant wins against base classifiers, its performance in a head-to-head comparison is equal to that of stacking.

In a second series of experiments, we wanted to test whether our intuitions were correct, and tested our original choice of IBk and the six base learners as level 1 learners. For these experiments, we used only a one-time ten-fold cross-validation all datasets (as opposed to the average of ten cross-validations shown in the previous table).

It turns out that IBk with ten nearest neighbors[7] is in fact one of the best level 1 learners among the seven tested.

The only algorithm that appears to be slightly better is NaiveBayes. However, all relative performances are within 1% of that of IBk, i.e., all algorithms perform about equally good. This came as a little surprise to us, because we had expected that there would be more differences in this wide range of algorithms. Some of the algorithms learn local models (IBk), while others always consider all examples for deriving a prediction (KStar). Likewise, some of the algorithms always use all of the features (NaiveBayes), while others try to focus on important ones (J48). Apparently, the particular type of meta-learning problem encountered in grading has some properties that make it uniformly hard for a wide variety of learning algorithms. We plan further investigations of this matter in the future.

4 Related Work

Our original motivation for the investigation of grading was to evaluate the potential of using qualitative error-characterization techniques proposed by [9] and [8] as an ensemble technique. There are some minor differences to these workshop papers with respect to how we compute the predictions at the meta level (e.g. the approach of [8] can only handle 2-class problems), but the idea

[7] After choosing IBk as the level 1 classifier, prior to the evaluation described here, we had determined the number of neighbors (10) using a ten-fold CV on all datasets.

underlying these approaches is more or less the same. This paper provides a thorough empirical evaluation of grading, and compares it to stacking, selection by cross-validation, and voting.

[4] propose the use of *arbiters* and *combiners*. A combiner is more or less identical to stacking. [4] also investigate a related form, which they call an *attribute-combiner*. In this architecture, the original attributes are not replaced with the class predictions, but instead they are added to them. As [14] shows in his paper about bi-level stacking, this may result in worse performance. On the other hand, [8] compared this approach to the above-mentioned 2-class version of grading, and found that grading performs significantly better on the KRK problem. These results have to be reconciled in future work.

An arbiter [4] is a separate, single classifier, which is trained on a subset of the original data. This subset consists of examples on which the base classifiers disagree. They also investigate *arbiter trees*, in which arbiters that specialize in arbiting between pairs of classifiers are organized in a binary decision tree. Arbiters are quite similar in spirit to grading. The main difference is that arbiters use information about the disagreement of classifiers for selecting a training set, while grading uses disagreement with the target function (estimated by a cross-validation) to produce a new training set.

Quite related to grading is also the work by [16], who proposed to use the predictions of base classifiers for learning a function that maps the algorithms' internal confidence measure (e.g., instance typicality for nearest neighbor classifiers or a posteriori probabilities for Naive Bayes classifiers) to an estimate of its accuracy on the output. The main differences to grading is that we use the original feature vectors as inputs, and select the best prediction based on the class probabilities returned by those meta classifiers that predict that their corresponding base classifiers are correct on the example.

A third approach with a similar goal, *meta decision trees* [18], aims at directly predicting which classifier is best to classify an individual example. To this end, it uses information about statistical properties of the predicted class distribution as attributes and predicts the right algorithm from this information. The approach is not modular (in the sense that any algorithm could be used at the meta-level) but implemented as a modification to the decision tree learner C4.5. Grading differs from meta decision trees in that respect, and by the fact that we use the original attributes in the datasets for learning an ensemble of classifiers which learn the errors of each base classifier.

There are many approaches to combine multiple models without resorting to elaborate meta-classification schemes. Best known are ensemble methods such as bagging and boosting, which rely on learning a set of diverse base classifiers (typically via different subsamples of the training set), whose predictions are then combined by simple voting [6]. Another group of techniques, *meta-learning*, focuses on predicting the right algorithm for a particular problem based on characteristics of the dataset [3] or based on the performance of other, simpler learning algorithms [11]. Finally, another common decision procedure (especially

with larger datasets) is to take a subsample of the entire dataset and try each algorithm on this sample. This approach was analyzed by [10].

5 Conclusions

We have examined the meta-classification scheme grading. Essentially, the idea behind grading is to train a new classifier that predicts which base classifier is correct on a given example. To that end, the original data set is transformed into a two-class dataset with new class labels that encode whether the the base classifier was able to correctly predict this example in an internal cross-validation or not. This approach may be viewed as a direct generalization of selection by cross-validation, which would always select the base classifier that corresponds to the meta dataset with the highest default accuracy.

The experimental evaluation showed that grading is slightly better than stacking according to some performance measures (like the average performance on a selection of UCI datasets or an indirect comparison to the base classifiers), but in a head-to-head comparison the differences are not statistically significant. Both algorithms, stacking and grading, perform better than voting and selection by cross-validation. We believe that both approaches are valid alternatives that should be considered when working with multiple models.

Our results also show that the method seems to be quite insensitive to the choice of a meta-learning algorithm. This is quite surprising, as we investigated a variety of algorithms with very different biases. We are yet unsure why this is the case.

A possible reason may lie in the fact that for reasonable performances of the base learning algorithms, the two-class meta data set consists of far more correct than incorrectly predicted examples. Hence the learner should be able to deal with imbalanced training sets, which none of the algorithms we tested specializes in. We have not yet investigated the influence of this issue upon the performance of the learning system.

We remain hopeful that our approach may in time become complementary to stacking, in particular if the respective strengths and weaknesses of the two approaches are better understood. To this end, we plan to investigate these issues by performing a strict empirical evaluation of the diversity of both classification schemes as well as study the influence of the diversity of the base classifiers on these meta classification schemes. Our current work concentrates on the definition of a common framework for meta-classification schemes, which allows a thorough experimental and theoretical comparison of the different approaches that have been proposed in the literature.

Acknowledgments This research is supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant no. P12645-INF and the ESPRIT LTR project METAL (26.357). The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture. We want to thank Johann Petrak and Gerhard Widmer for valuable discussions and comments.

References

1. Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning, 36*, 105–169.
2. Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html. Department of Information and Computer Science, Unifirsty of California at Irvine, Irvine CA.
3. Brazdil, P. B., Gama, J., & Henery, B. (1994). Characterizing the applicability of classification algorithms using meta-level learning. *Proceedings of the 7th European Conference on Machine Learning (ECML-94)* (pp. 83–102). Catania, Italy: Springer-Verlag.
4. Chan, P. K., & Stolfo, S. J. (1995). A comparative evaluation of voting and meta-learning on partitioned data. *Proceedings of the 12th International Conference on Machine Learning (ICML-95)* (pp. 90–98). Morgan Kaufmann.
5. Cleary, J. G., & Trigg, L. E. (1995). K*: An instance-based learner using an entropic distance measure. *Proceedings of the 12th International Conference on Machine Learning* (pp. 108–114). Lake Tahoe, CA.
6. Dietterich, T. G. (2000a). Ensemble methods in machine learning. *First International Workshop on Multiple Classifier Systems* (pp. 1–15). Springer-Verlag.
7. Kononenko, I., & Bratko, I. (1991). Information-based evaluation criterion for classifier's performance. *Machine Learning, 6*, 67–80.
8. Koppel, M., & Engelson, S. P. (1996). Integrating Multiple Classifiers By Finding Their Areas of Expertise. *Proceedings of the AAAI-96 Workshop on Integrating Multiple Models* (pp. 53–58).
9. Ortega, J. (1996). Exploiting Multiple Existing Models and Learning Algorithms *Proceedings of the AAAI-96 Workshop on Integrating Multiple Models* (pp. 101–106).
10. Petrak, J. (2000). Fast subsampling performance estimates for classification algorithm selection. *Proceedings of the ECML-00 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination* (pp. 3–14). Barcelona, Spain.
11. Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. (2000). Meta-learning by landmarking various learning algorithms. *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*. Stanford, CA.
12. Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.
13. Schaffer, C. (1993). Selecting a classification method by cross-validation. *Machine Learning, 13*, 135–143.
14. Schaffer, C. (1994). Cross-validation, stacking and bi-level stacking: Meta-methods for classification learning. In P. Cheeseman and R. W. Oldford (Eds.), *Selecting models from data: Artificial Intelligence and Statistics IV*, 51–59. Springer-Verlag.
15. Seewald, A. K., & Fürnkranz, J. (2001). *Grading classifiers* (Technical Report OEFAI-TR-2001-01). Austrian Research Institute for Artificial Intelligence, Wien.
16. Ting, K. M. (1997). Decision combination based on the characterisation of predictive accuracy. *Intelligent Data Analysis, 1*, 181–206.
17. Ting, K. M., & Witten, I. H. (1999). Issues in stacked generalization. *Journal of Artificial Intelligence Research, 10*, 271–289.
18. Todorovski, L., & Džeroski, S. (2000). Combining multiple models with meta decision trees. *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD-2000)* (pp. 54–64). Lyon, France: Springer-Verlag.
19. Wolpert, D. H. (1992). Stacked generalization. *Neural Networks, 5*, 241–260.