Efficiency Comparison of Unstable Transductive and Inductive Conformal Classifiers

Henrik Linusson[1], Ulf Johansson[1], Henrik Boström[2], and Tuve Löfström[1]

[1] School of Business and IT
University of Borås, Borås, Sweden
{henrik.linusson,ulf.johansson,tuve.lofstrom}@hb.se
[2] Dept. of Computer and Systems Sciences
Stockholm University, Kista, Sweden
henrik.bostrom@dsv.su.se

Abstract. In the conformal prediction literature, it appears axiomatic that transductive conformal classifiers possess a higher predictive efficiency than inductive conformal classifiers, however, this depends on whether or not the nonconformity function tends to overfit misclassified test examples. With the conformal prediction framework's increasing popularity, it thus becomes necessary to clarify the settings in which this claim holds true. In this paper, the efficiency of transductive conformal classifiers based on decision tree, random forest and support vector machine classification models is compared to the efficiency of corresponding inductive conformal classifiers. The results show that the efficiency of conformal classifiers based on standard decision trees or random forests is substantially improved when used in the inductive mode, while conformal classifiers based on support vector machines are more efficient in the transductive mode. In addition, an analysis is presented that discusses the effects of calibration set size on inductive conformal classifier efficiency.

1 Introduction

Conformal Prediction [1] is a machine learning framework for associating predictions for novel data with a measure of their *confidence*; whereas traditional machine learning algorithms produce *point predictions* — a single label $\hat{y}$ per test example — conformal predictors produce *prediction regions* — prediction sets $\hat{Y} \subseteq Y$ that, in the long run, contain the true labels for the test set with some predefined probability $1 - \epsilon$. Historically, such confidence predictions have relied on the Bayesian learning and Probably Approximately Correct (PAC) learning frameworks; however, the validity of Bayesian confidence predictions relies on an assumption of the *a priori* distribution, while PAC learning confidence measures apply to the entire model, and not the individual predictions [2]. In contrast, conformal predictors are able to produce confidence measures tailored for each separate prediction on novel data, and rely only on the assumption that the data is *exchangeable* — that the ordering of data points does not affect their joint distributions — which is an even weaker assumption than the i.i.d. assumption typically required by traditional machine learning algorithms.

The method of constructing prediction regions using conformal predictors relies on measuring the strangeness — the *nonconformity* — of each data point, using some (arbitrary) real-valued function, called a *nonconformity function*, that measures the degree of strangeness of an example (x_i, y_i) in relation to a bag (multiset) of examples $Z = \{(x_1, y_1), ..., (x_i, y_i), ..., (x_k, y_k)\}$. For classification problems, this nonconformity measure is often based on the predictions of a traditional machine learning algorithm, called the *underlying model* of the conformal predictor. By comparing the nonconformity of a tentative classification $(x_{k+1}, \tilde{y})$ for a novel (test) input pattern x_{k+1} to the nonconformity scores of previously seen data, a conformal predictor can make inferences as to whether $\tilde{y}$ is likely to be a correct classification for x_{k+1}, and thus decide whether or not to include $\tilde{y}$ in the prediction region $\hat{Y}_{k+1}$.

There are two major categories of conformal predictors: transductive conformal predictors (TCP) [3,4] and inductive conformal predictors (ICP) [2,5]; both variants are based on the same principles, and place the same exchangability requirement on the data. Where they differ is in their usage of the training data, and their overall training scheme. For a novel input pattern x_{k+1} and every tentative prediction $\tilde{y} \in Y$, TCP measures the nonconformity of all examples in the bag $Z' = Z \cup \{(x_{k+1}, \tilde{y})\}$ relative to each other, meaning that the instance $(x_{k+1}, \tilde{y})$ is considered when calculating the nonconformity scores for the training set Z. Hence, for every new pattern x_{k+n} and every $\tilde{y}$, the underlying model needs to be retrained, and the nonconformity scores for the training data recomputed, rendering TCP computationally intensive. In ICP, only part of the data is used to train the underlying model (once), while the remaining data (a *calibration set*) is set aside to provide an unbiased estimate of the distribution of nonconformity scores.

The predictive *efficiency* — that is, the size of the prediction regions produced — of any conformal predictor is closely tied to the nonconformity function's ability to rank examples by order of strangeness. Moreover, as noted in several papers, ICP models typically suffer a small loss in predictive efficiency compared to corresponding TCP models due to the reduced number of training and calibration examples [1, 5–8]. However, as pointed out in [1], an *unstable* nonconformity function — one that is heavily influenced by an outlier example, i.e., an erroneously labled test instance $(x_{k+1}, \tilde{y})$ — can cause TCP models to become inefficient.

So, the choice between TCP and ICP is not so clear-cut: on the one hand, ICP will surely always produce its predictions faster than TCP, and TCP is often expected to have a higher predictive power than ICP. On the other hand, the efficiency of TCP relies on the chosen nonconformity function being *stable*, meaning that the underlying model does not train outlier examples into its learned rule [1]. When choosing a conformal prediction setup, a user should thus consider not only the trade-off between predictive speed and predictive power in

TCP and ICP, but also whether the chosen nonconformity function can be used effectively in TCP.

Decision trees and random forests are commonly used model types that are able to accommodate their training examples well, and, due to their ability to near-perfectly fit their training data, these model types should be expected to function better as underlying models in ICP than in TCP. To the best of our knowledge, however, this claim has not been investigated in existing literature.

In this study, the efficiency of TCP classifiers based on off-the-shelf decision trees and random forests, implemented by the scikit-learn [9] Python package, is compared to the efficiency of corresponding ICP classifiers; the results of this comparison are juxtaposted with an identical comparison of TCP and ICP classifiers based on stable support vector machines. To allow for a straightforward comparison of TCP and ICP classifiers, an analysis is also presented that discussess the effects of calibration set size on ICP classifier efficiency.

2 Background

Given some nonconformity scoring function $\Delta(Z, x_i, y_i) \rightarrow \overline{\mathbb{R}}$, a conformal predictor can produce prediction regions that are *valid* in the sense that they contain the true target with some predefined probability $1 - \epsilon$. For classification problems, a common approach is to define a nonconformity function by combining a traditional machine learning algorithm and some error measure of the algorithm's predictions, e.g., the margin scoring function used in [10]:

$$\Delta(h, x_i, y_i) = P(y_i \mid h(x_i)) - \underset{y' \neq y_i}{\arg\max}\, P(y' \mid h(x_i)), \tag{1}$$

where h is a predictive model trained on some set Z, i.e., h represents a generalization of Z. Given the nature of classification problems in general, it is to be expected that a test pattern that deviates from the examples found in Z is likely to be misclassified by h, i.e., examples that are not conforming to Z are likely to be assigned large nonconformity scores.

When making a prediction using a conformal classifier, the nonconformity function is applied to a set of examples with known labels, resulting in a set of nonconformity scores $\alpha_1, ..., \alpha_k$ that represents a sample which is to be used for statistical inference (we here refer to this as the *nonconformity baseline*). The test pattern x_{k+1} is then assigned a tentative classification $(x_{k+1}, \tilde{y})$, where $\tilde{y} \in Y$, and a nonconformity score $\alpha_{k+1} = \Delta(Z, x_{k+1}, \tilde{y})$. Using a form of hypothesis testing (described in Sections 2.1 and 2.2), the conformal classifier attempts to reject the null hypothesis that $(x_{k+1}, \tilde{y})$ is conforming with Z, i.e., if the nonconformity score α_{k+1} is higher than for most examples in Z, as estimated by the nonconformity baseline, then $\tilde{y}$ is considered to be an unlikely classification for x_{k+1} and can be excluded from the final prediction set.

2.1 Transductive Conformal Predictors

A transductive conformal classifier uses the full training set Z to establish the nonconformity baseline. Due to the exchangeability assumption, this requires that the test pattern $(x_{k+1}, \tilde{y})$ is considered when calculating the nonconformity scores $\alpha_1, ..., \alpha_k$, i.e., if the nonconformity function Δ is based on an inductive model h, then $(x_{k+1}, \tilde{y})$ must be included in the training set for h. If it is not, then there is a possibility that h will have a larger bias towards its training examples than towards $(x_{k+1}, \tilde{y})$, meaning that the training examples and the test pattern might have different expected nonconformity values (which is a direct violation of the exchangeability assumption[1]). TCP thus requires that the underlying model h is trained $n\,|Y|$ times, where n is the number of test patterns, using the following scheme:

1. Assume a label $\tilde{y}$ and form the set $Z' = Z \cup \{(x_{k+1}, \tilde{y})\}$.
2. Use Z' to train a model h.
3. For each $(x_i, y_i) \in Z'$ calculate $\alpha_i = \Delta(h, x_i, y_i)$.
4. Calculate the p-value for $(x_{k+1}, \tilde{y})$ as

$$p(x_{k+1}, \tilde{y}) = \frac{|\{z_i \in Z' \mid \alpha_i \geq \alpha_{k+1}\}|}{|Z'|} \,. \tag{2}$$

5. If $p(x_{k+1}, \tilde{y}) > \epsilon$, include $\tilde{y}$ in the prediction region $\hat{Y}_{k+1}$.

2.2 Inductive Conformal Predictors

Inductive conformal predictors instead use only part of the training data to fit the underlying model h, setting aside a calibration set that is later used to establish the nonconformity baseline. Since the proper training set $Z^t \subset Z$ used to fit h and the calibration set $Z^c \subset Z$ are disjoint, the nonconformity scores $\alpha_1, ..., \alpha_{k-t}$ are exchangeable (unbiased) with the nonconformity score α_{k+1} without the need for including $(x_{k+1}, \tilde{y})$ in the training of h; thus, an ICP only needs to be trained once:

1. Divide Z into two disjoint subsets Z^t and Z^c.
2. Use Z^t to train a model h.
3. For each $(x_i, y_i) \in Z^c$, calculate $\alpha_i = \Delta(h, x_i, y_i)$.

For a novel test instance x_{k+1}:

1. Assume a label $\tilde{y}$ and calculate $\alpha_{k+1} = \Delta(h, x_{k+1}, \tilde{y})$.
2. Calculate the p-value for $(x_{k+1}, \tilde{y})$ as

$$p(x_{k+1}, \tilde{y}) = \frac{|\{z_i \in Z^c \mid \alpha_i \geq \alpha_{k+1}\}| + 1}{|Z^c| + 1} \,. \tag{3}$$

3. If $p(x_{k+1}, \tilde{y}) > \epsilon$, include $\tilde{y}$ in the prediction region $\hat{Y}_{k+1}$.

[1] If the model is able to better predict the correct output for the training examples than the test example, then the p-values for the true targets will no longer be uniformly distributed as required by the conformal prediction framework [1].

3 Method

The main point of interest of this study is the efficiency comparison of TCP and ICP classifiers based on decision tree, random forest and support vector machine models, but, to provide a straightforward comparison between the two, it is also necessary to find a suitable choice of calibration set size for the ICP classifiers. To the best of our knowledge, no thorough investigation has been published that discusses the effects of calibration set size on ICP classifier efficiency, and so the results presented in this paper contain first a discussion of ICP classifier setup, and second a comparison of ICP and TCP classifier efficiency.

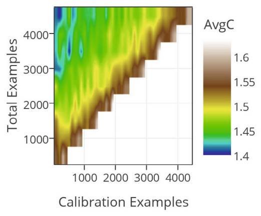
To investigate what effect the calibration set size has on the efficiency of ICP classifiers using various types of underlying models, several ICP models were trained on five binary classification sets from the LIBSVM website [11] (a9a, covertype, cod-rna, ijcnn1 and w8a), using different amounts of training and calibration data. For each dataset, stratified random samples of $s = 500, 1000, ..., 4500$ examples were drawn, and for each s, several ICP models were applied to the same test set of 100 examples, each ICP model using $c = 100, 200, ..., s - 100$ calibration examples and $t = s - c$ training examples.

To compare the efficiency of the ICP and TCP variants, both types of classifiers were trained on 19 binary classification sets from the UCI repository [12]. The ICP models used a suitable calibration size as suggested by the results from the first experiment (20%). To limit the training time required for the TCP models, results from the larger datasets (kr-vs-kp, sick and spambase) were obtained using 1x10-fold cross-validation, while the results from the remaining sets were obtained using 5x10-fold cross-validation.

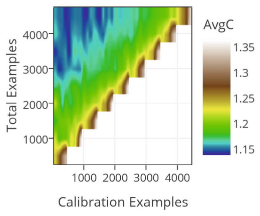
In both experiments, three different types of underlying models were used: decision trees (CART) [13], random forests (RF) [14] and support vector machines (SVM) [15]. The CART models used relative frequency probability estimates and no pruning, making them highly susceptible to overfitting noisy data. RF models (here consisting of 100 trees) are relatively robust to noise, but since the ensemble members are simple CART models, noisy examples are still fit into the learned rule to some extent. SVM models are, in contrast, stable to isolated noisy data points [16]. The scikit-learn library [9] for Python, using default settings, was used to train the underlying models. The margin nonconformity function (1) was used to construct the conformal classifiers, and predictions were made in the off-line (batch) mode.

4 Results

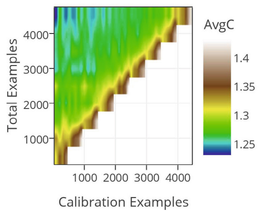
Figure 1 shows the relationships between training set size, calibration set size and efficiency (measured as AvgC — the average number of labels per prediction). The size of the calibration set is expressed as a portion of the full dataset, i.e., the point $(500, 1000)$ uses 500 of the total (1000) available examples as calibration data, and the remaining 500 examples as training data. The results are averaged over all five datasets and all five iterations. Clearly, the more data that is made available to the ICP classifier, the more efficient its predictions are, but only if the data is used sensibly. If more data is made available, and all newly available examples are added to the calibration set (any line $y = x + k$ in the six plots), the efficiency remains virtually unchanged at $\epsilon = 0.05$. If instead all newly available examples are added to the proper training set (any line parallel to the Y axis) the efficiency of the ICP model increases greatly, whereas, naturally then, the efficiency decreases if a larger calibration set is used while the total amount of available data remains unchanged (any line parallel to the X axis). The chosen (absolute) size of the calibration set thus has a small effect on ICP classifier efficiency, while the size of the proper training set is more strongly correlated with efficiency (cf. the correlations, r_t for training set size and r_c for calibration set size, listed below the plots).



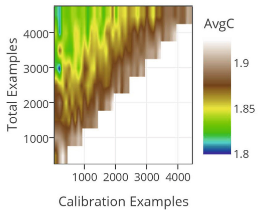
(a) CART, $\epsilon = 0.05$
$r_t = -0.106, r_c = 0.005$



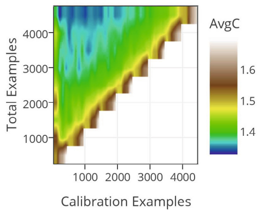
(b) RF, $\epsilon = 0.05$
$r_t = -0.146, r_c = -0.004$



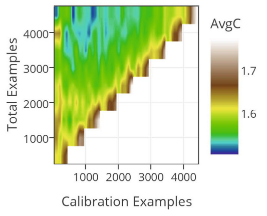
(c) SVM, $\epsilon = 0.05$
$r_t = -0.097, r_c = -0.010$



(d) CART, $\epsilon = 0.01$
$r_t = -0.135, r_c = -0.001$



(e) RF, $\epsilon = 0.01$
$r_t = -0.205, r_c = -0.023$



(f) SVM, $\epsilon = 0.01$
$r_t = -0.103, r_c = -0.010$

Fig. 1. ICP AvgC based on portion of examples used in the calibration set. The y-axis represents a growing dataset, while the x-axis represents a growing calibration set (relative to the full dataset). Blue areas represent the most efficient combinations of training and calibration set sizes, green and yellow areas are moderately efficient, while brown areas are the most inefficient.

As illustrated by Figures 1d, 1e and 1f, the size of the calibration set plays a larger role in determining efficiency when $\epsilon = 0.01$ than it does when $\epsilon = 0.05$, although the training set size is clearly the most important factor for maximizing efficiency at both significance levels. The best performing ICP classifiers are those using $15 - 30\%$ of the full training set as their calibration set. Notably, the performance of ICP classifiers using a calibration set containing less than 500 examples is quite poor when $\epsilon = 0.01$, regardless of training set size, suggesting that at least a few hundred examples should be used for calibration unless this leaves too few examples in the proper training set. The coverage (i.e. the 'empirical validity') of the conformal classifiers is not tabulated, however, all tested conformal classifiers show a coverage at or near the expected error rates on average. Note though, that the conformal classifiers using a very small calibration set (100 examples) displayed a larger variance in their coverage than the conformal classifiers using 500 calibration examples; in turn, using 500 calibration examples showed no substantial increase in variance compared to using a larger calibration set.

Table 1. AvgC of TCP and ICP based on CART, RF and SVM, $\epsilon = 0.05$

			CART		RF		SVM	
dataset	#f	#ex	ICP	TCP	ICP	TCP	ICP	TCP
balance-scale	4	577	1.571	1.891	1.037	1.044	0.957	0.951
breast-cancer	9	286	1.878	1.859	1.719	1.809	1.711	1.733
breast-w	10	699	1.259	1.520	0.973	0.976	0.987	0.993
credit-a	15	690	1.747	1.902	1.284	1.328	1.766	1.774
credit-g	20	1000	1.845	1.902	1.548	1.596	1.821	1.849
diabetes	20	769	1.842	1.896	1.541	1.601	1.822	1.861
haberman	3	306	1.849	1.843	1.649	1.773	1.680	1.607
heart-c	14	303	1.796	1.903	1.402	1.452	1.885	1.849
heart-h	14	264	1.808	1.898	1.402	1.460	1.838	1.836
heart-s	13	270	1.813	1.896	1.445	1.460	1.904	1.900
hepatitis	19	155	1.821	1.885	1.400	1.350	1.844	1.820
ionosphere	34	351	1.588	1.896	1.057	1.047	1.059	1.040
kr-vs-kp	36	3196	0.949	1.895	0.953	0.949	1.036	1.016
labor	16	57	1.719	1.909	1.400	1.115	1.525	1.039
liver-disorders	7	345	1.859	1.880	1.753	1.740	1.857	1.846
sonar	30	208	1.829	1.904	1.373	1.401	1.720	1.648
tic-tac-toe	9	958	1.255	1.906	0.960	0.963	1.161	1.045
sick	30	3772	0.964	1.886	0.954	0.950	1.062	1.036
spambase	57	4601	1.370	1.549	1.021	1.028	1.050	1.052
mean rank			**1.105**	1.895	1.316	1.684	1.684	1.316

Tables 1 and 2 show a comparison of the efficiency of ICP and TCP classifiers based on CART, random forest and SVM models. Note that the ranks are computed only within each TCP-ICP pair, i.e., the different underlying model types are not compared to each other in terms of efficiency ranks. Mean ranks in bold indicate significant differences at $\alpha = 0.05$ as reported by a two-tailed Wilcoxon signed-ranks test.

The stable SVM models do indeed appear to gain from being able to use the full dataset for both training and calibration when used in the TCP mode. At $\epsilon = 0.01$, the SVM-based TCP classifiers are significantly more efficient that SVM-based ICP classifiers; at $\epsilon = 0.05$ the SVM-TCP classifiers are more efficient on

Table 2. AvgC of TCP and ICP based on CART, RF and SVM, $\epsilon = 0.01$

			CART		RF		SVM	
dataset	#f	#ex	ICP	TCP	ICP	TCP	ICP	TCP
balance-scale	4	577	1.911	1.979	1.175	1.217	1.039	1.022
breast-cancer	9	286	1.977	1.976	1.931	1.965	1.940	1.972
breast-w	10	699	1.843	1.589	1.178	1.187	1.256	1.228
credit-a	15	690	1.951	1.982	1.806	1.810	1.934	1.936
credit-g	20	1000	1.969	1.981	1.838	1.871	1.959	1.972
diabetes	20	769	1.970	1.978	1.798	1.875	1.964	1.979
haberman	3	306	1.971	1.959	1.894	1.936	1.935	1.922
heart-c	14	303	1.966	1.981	1.800	1.815	1.982	1.976
heart-h	14	294	1.963	1.980	1.814	1.821	1.950	1.933
heart-s	13	270	1.953	1.979	1.837	1.827	1.984	1.979
hepatitis	19	155	1.960	1.983	1.832	1.758	1.976	1.966
ionosphere	34	351	1.917	1.974	1.589	1.454	1.543	1.330
kr-vs-kp	36	3196	1.067	1.980	1.008	1.000	1.203	1.258
labor	16	57	1.952	1.987	1.845	1.683	1.914	1.615
liver-disorders	7	345	1.978	1.959	1.945	1.929	1.969	1.948
sonar	30	208	1.968	1.980	1.804	1.672	1.922	1.872
tic-tac-toe	9	958	1.828	1.978	1.059	1.062	1.488	1.328
sick	30	3772	1.356	1.957	1.013	1.021	1.749	1.621
spambase	57	4601	1.875	1.673	1.272	1.623	1.352	1.344
mean rank			**1.263**	1.737	1.368	1.632	1.737	**1.263**

average, although the difference is not significant. It is evident, however, that the unpruned CART models — the most noise sensitive of the three model types — struggle with rejecting erroneous class labels in a TCP setting, likely due to the noisy test examples $(x_i, \tilde{y} \neq y_i)$ being fitted into the learned rules. TCP models based on CART are significantly less efficient than ICP-CART, both at $\epsilon = 0.05$ and $\epsilon = 0.01$. As expected, the RF models fare better in terms of TCP efficiency; the overfitting of the underlying CART models is counteracted by the smoothing effect of the ensemble constellation, however, not to such an extent that the TCP-RF models are more efficient than ICP-RF. ICP-RF is more efficient than TCP-RF on average, but the difference is not significant, neither at $\epsilon = 0.05$ nor at $\epsilon = 0.01$.

5 Related Work

As noted in [1], TCP classifiers using unstable nonconformity functions can be made more efficient through a slight modification of the transductive procedure. This modification involves calculating the nonconformity scores in a leave-one-out manner (LOOTCP), where the nonconformity of an example $(x_i, y_i) \in Z'$ is calculated from the set $Z' \setminus (x_i, y_i)$. Hence, in LOOTCP, the underlying model needs to be retrained not only for every test example and every tentative classification, but also for every training example. In principle, LOOTCP should indeed be expected to produce more efficient predictions than ICP, however, as LOOTCP increases the computational complexity of the (already computationally intensive) TCP classifier by a factor k, it is questionable whether it is truly applicable in practice when the nonconformity function requires training.

In [17], random forests are used to construct conformal classifiers run in the transductive mode. Rather than using the entire forest when calculating the nonconformity scores for calibration and test examples, the error of the out-of-bag prediction is used, i.e., to calculate the nonconformity score for a specific example, only ensemble members not trained on that particular example are considered (similar to LOOTCP). Although not explicitly stated by the authors, this effectively results in a random forest TCP that is not affected by the potential issues of strange examples being trained into the model.

In [18], a method for estimating the prediction certainty in decision trees, similar to conformal prediction, is proposed. The authors note that the default tree induction algorithm requires a modification for the predictions to be useful. Since the goal is to identify examples that are difficult to predict, the test instance (which is, as in TCP, included in the training data) is only allowed to affect node splitting to a limited extent, to avoid fitting strange examples into the models.

6 Concluding Remarks

This study shows that inductive conformal classifiers based on standard decision tree and random forest models can be more efficient than corresponding transductive conformal classifiers, while the opposite is true for support vector machines. This is contrary to the commonly accepted claim that transductive conformal predictors are by default more efficient than inductive conformal predictors. It has also been shown that to maximize the efficiency of inductive conformal classifiers, the calibration set should be kept small relative to the amount of available data ($15 - 30\%$). At the same time, if the training set is large enough, it appears favourable to let the calibration set contain at least a few hundred examples.

For future work, we suggest that transductive and inductive conformal classifiers based on other types of classification models should be compared, to provide guidelines for designing conformal classification systems. Similarly, the efficiency of specialized transductive models, such as those proposed in [17], should be contrasted to the efficiency of standard transductive and inductive variants.

Acknowledgements. This work was supported by the Swedish Foundation for Strategic Research through the project High-Performance Data Mining for Drug Effect Detection (IIS11-0053) and the Knowledge Foundation through the project Big Data Analytics by Online Ensemble Learning (20120192).

References

1. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic learning in a random world. Springer Verlag, DE (2006)
2. Papadopoulos, H.: Inductive conformal prediction: Theory and application to neural networks. Tools in Artificial Intelligence 18, 315–330 (2008)
3. Gammerman, A., Vovk, V., Vapnik, V.: Learning by transduction. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 148–155. Morgan Kaufmann Publishers Inc. (1998)
4. Saunders, C., Gammerman, A., Vovk, V.: Transduction with confidence and credibility. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI 1999), vol. 2, pp. 722–726 (1999)
5. Papadopoulos, H., Proedrou, K., Vovk, V., Gammerman, A.: Inductive confidence machines for regression. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) ECML 2002. LNCS (LNAI), vol. 2430, pp. 345–356. Springer, Heidelberg (2002)
6. Papadopoulos, H.: Inductive conformal prediction: Theory and application to neural networks. Tools in Artificial Intelligence 18(315-330), 2 (2008)
7. Papadopoulos, H., Vovk, V., Gammerman, A.: Conformal prediction with neural networks. In: 19th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2007, vol. 2, pp. 388–395. IEEE (2007)
8. Balasubramanian, V.N., Ho, S.S., Vovk, V.: Conformal prediction for reliable machine learning: theory, adaptations, and applications. Elsevier, Waltham (2013) (to appear)
9. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. The Journal of Machine Learning Research 12, 2825–2830 (2011)
10. Johansson, U., Boström, H., Löfström, T.: Conformal prediction using decision trees. In: IEEE International Conference on Data Mining (2013)
11. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST), Software, available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
12. Bache, K., Lichman, M.: UCI machine learning repository (2013), `http://archive.ics.uci.edu/ml`
13. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and regression trees. CRC press (1984)
14. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
15. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297 (1995)
16. Buciu, I., Kotropoulos, C., Pitas, I.: Demonstrating the stability of support vector machines for classification. Signal Processing 86(9), 2364–2380 (2006)
17. Devetyarov, D., Nouretdinov, I.: Prediction with confidence based on a random forest classifier. In: Papadopoulos, H., Andreou, A.S., Bramer, M. (eds.) AIAI 2010. IFIP AICT, vol. 339, pp. 37–44. Springer, Heidelberg (2010)
18. Costa, E.P., Verwer, S., Blockeel, H.: Estimating prediction certainty in decision trees. In: Tucker, A., Höppner, F., Siebes, A., Swift, S. (eds.) IDA 2013. LNCS, vol. 8207, pp. 138–149. Springer, Heidelberg (2013)