Inductive Venn–Abers prediction for regression

Ivan Petej

July 31, 2018

Abstract

This brief report outlines the proposed algorithm of Inductive Venn–Abers prediction applied to regression with some preliminary results.

1 Introduction

Venn predictors produce probability-type predictions for the labels of test objects which are guaranteed to be well calibrated under the standard assumption that the observations are generated independently from the same distribution. Recently a new class of Venn predictors were introduced, called Venn–Abers predictors [6], based on the idea of isotonic regression. with promising empirical results.

Traditionally the Venn–Abers method has been applied to the problem of classification. More recently, the framework has been extended to regression [4]. This method can be applied on top of any regression method to obtain calibrated predictive distributions, without requiring assumptions beyond i.i.d. of calibration and test sets.

This report outlines an alternative algorithm based on the Inductive version of Venn–Abers predictors (IVAP) [7], which just like standard Venn–Abers predictors automatically enjoy the property of validity but are computationally more efficient. In this report we consider a novel version of the IVAP algorithm modified for the use in regression. Our hope is that this new algorithm can offer a complementary method to the one described in [4].

2 Venn–Abers predictors

Many standard machine-learning algorithms for classification are in fact *scoring classifiers*: when trained on a training sequence of observations and fed with a test object x,

they output a *prediction score* $s(x)$; we will call $s : \mathbf{X} \to \mathbb{R}$ the *scoring function* for that training sequence. The actual classification algorithm is obtained by fixing a threshold c and predicting the label of x to be 1 if and only if $s(x) \geq c$ (or if and only if $s(x) > c$). Alternatively, one could apply an increasing function g to $s(x)$ in an attempt to "calibrate" the scores, so that $g(s(x))$ can be used as the predicted probability that the label of x is 1.

Fix a scoring classifier and let $(z_1, \dots, z_l)$ be a training sequence of observations $z_i = (x_i, y_i)$, $i = 1, \dots, l$. The most direct application [9] of the method of isotonic regression [1] to the problem of score calibration is as follows. Train the scoring classifier on the training sequence and compute the score $s(x_i)$ for each training observation (x_i, y_i), where s is the scoring function for $(z_1, \dots, z_l)$. Let g be the increasing function on the set $\{s(x_1), \dots, s(x_l)\}$ that maximizes the likelihood

$$\prod_{i=1}^{l} p_i, \quad \text{where } p_i := \begin{cases} g(s(x_i)) & \text{if } y_i = 1 \\ 1 - g(s(x_i)) & \text{if } y_i = 0. \end{cases} \tag{1}$$

Such a function g is indeed unique [1, Corollary 2.1] and can be easily found using the "pair-adjacent violators algorithm" (PAVA, described in detail in the summary of [1] and in [2, Section 1.2]. We will say that g is the *isotonic calibrator* for $((s(x_1), y_1), \dots, (s(x_l), y_l))$. To predict the label of a test object x, the direct method finds the closest $s(x_i)$ to $s(x)$ and outputs $g(s(x_i))$ as its prediction. We will refer to this as the *direct isotonic-regression* (DIR) method.

The direct method is prone to overfitting as the same observations $z_1, \dots, z_l$ are used both for training the scoring classifier and for calibration without taking any precautions. The *Venn–Abers predictor* corresponding to the given scoring classifier is the multiprobabilistic predictor that is defined as follows. Try the two different labels, 0 and 1, for the test object x. Let s_0 be the scoring function for $(z_1, \dots, z_l, (x, 0))$, s_1 be the scoring function for $(z_1, \dots, z_l, (x, 1))$, g_0 be the isotonic calibrator for

$$\big((s_0(x_1), y_1), \dots, (s_0(x_l), y_l), (s_0(x), 0)\big), \tag{2}$$

and g_1 be the isotonic calibrator for

$$\big((s_1(x_1), y_1), \dots, (s_1(x_l), y_l), (s_1(x), 1)\big). \tag{3}$$

The multiprobabilistic prediction output by the Venn–Abers predictor is (p_0, p_1), where $p_0 := g_0(s_0(x))$ and $p_1 := g_1(s_1(x))$. (And we can expect p_0 and p_1 to be close to each other unless DIR overfits grossly.) The Venn–Abers predictor is described as Algorithm 1.

Algorithm 1 Venn–Abers predictor

Input: training sequence $(z_1, \dots, z_l)$

Input: test object x

Output: multiprobabilistic prediction (p_0, p_1)

for $y \in \{0, 1\}$ **do**

set s_y to the scoring function for

$(z_1, \dots, z_l, (x, y))$

set g_y to the isotonic calibrator for

$(s_y(x_1), y_1), \dots, (s_y(x_l), y_l), (s_y(x), y)$

set $p_y := g_y(s_y(x))$

The intuition behind Algorithm 1 is that it tries to evaluate the robustness of the DIR prediction. To see how sensitive the scoring function is to the training set we extend the latter by adding the test object labelled in two different ways. And to see how sensitive the probabilistic prediction is, we again consider the training set extended in two different ways (if it is sensitive, the prediction will be fragile even if the scoring function is robust). For large data sets and inflexible scoring functions, we will have $p_0 \approx p_1$, and both numbers will be close to the DIR prediction. However, even if the data set is very large but the scoring function is very flexible, p_0 can be far from p_1 (the extreme case is where the scoring function is so flexible that it ignores all observations apart from a few that are most similar to the test object, and in this case it does not matter how big the data set is). We rarely know in advance how flexible our scoring function is relative to the size of the data set, and the difference between p_0 and p_1 gives us some indication of this. Since Venn–Abers predictors output pairs of probabilities rather than point probabilities, we need to fit them in the standard framework extracting one probability p from p_0 and p_1. Using log loss as a calibration metric, this is shown [6] to be given by:

$$p = \frac{p_1}{1 - p_0 + p_1}. \tag{4}$$

3 Inductive Venn–Abers predictors

Venn–Abers predictors were originally introduced in [6] which also showed that Venn–Abers predictors are Venn predictors and, therefore, inherit all properties of validity of the latter. Despite the promising empirical results demonstrated in improving probabilistic class membership classification one of the drawbacks of the Venn–Abers

method is in its computational inefficiency (as an example in Algorithm 1 the underlying algorithm needs to be retrained twice for each new test example).

Inductive Venn–Abers predcitors (IVAPs), introduced in [7] overcome the computational inefficiency of the standard Venn–Abers method. Consider data sequences (usually loosely referred to as sets) consisting of *observations* $z = (x, y)$, each observation consisting of an *object* x and a *label* $y \in \{0, 1\}$; we only consider binary labels. We are given a training set whose size will be denoted l. Just like VAPs described in the previous section, IVAP are defined in terms of a scoring algorithm. he IVAP computational steps are as follows:

- Divide the training set of size l into two subsets, the *proper training set* of size m and the *calibration set* of size k, so that $l = m + k$.
- Train the scoring algorithm on the proper training set.
- Find the scores $s_1, \dots, s_k$ of the calibration objects $x_1, \dots, x_k$.
- When a new test object x arrives, compute its score s. Fit isotonic regression to $(s_1, y_1), \dots, (s_k, y_k), (s, 0)$ obtaining a function f_0. Fit isotonic regression to $(s_1, y_1), \dots, (s_k, y_k), (s, 1)$ obtaining a function f_1. The multiprobability prediction for the label y of x is the pair $(p_0, p_1) := (f_0(s), f_1(s))$ (intuitively, the prediction is that the probability that $y = 1$ is either $f_0(s)$ or $f_1(s)$).

Notice that the multiprobability prediction (p_0, p_1) output by an IVAP always satisfies $p_0 < p_1$, and so p_0 and p_1 can be interpreted as the lower and upper probabilities, respectively; in practice, they are close to each other for large training sets.

The main step within th inductive Venn–Abers framework is the precompuation of f_0 and f_1 for any possible value of the score using the calibration set. This is done efficiently using the geometric representation of isotonic regression as the slope of the GCM (greatest convex minorant) of the CSD (cumulative sum diagram): see [2], pages 9–13 (especially Theorem 1.1). Given the scores $s_1, \dots, s_k$ of the calibration objects, the prediction rule for computing the IVAP's predictions can be computed in time $O(k \log k)$ and space $O(k)$. Its application to each test object takes time $O(\log k)$. Given the sorted scores of the calibration objects, the prediction rule can be computed in time and space $O(k)$.

4 Inductive Venn–Abers method applied to regression

Traditionally the Venn–Abers method has been applied to the problem of classification. More recently, the framework has been extended to regression [4]. As the authors show in moving from the binary classification (where the labels $y \in (0, 1)$) to the regression problem (where $y \in (-\infty, \infty)$), the approach is to map the original regression setting onto a binary classification setting on which we can apply Venn-Abers prediction.

In this report we apply a similar method and assume that within the regression setting the scores s are given by the underlying algorithm as predictions for the true labels y. Let us assume we have a training set with a total of n examples given by $(s_1, \ldots, s_n, y_1, \ldots, y_n)$ (and for simplicity, let us assume that the n examples consist of unique scores s). Next let us select a threshold y_i where i corresponds to the chosen integer i in $i = 1, \ldots, n$. We then map the original labels $y_1, \ldots, y_n$ into a set of labels $y'_1, \ldots, y'_n$ such that $y'_n = 0$ if $y_n =< y_i$ and 1 otherwise. This transforms a regression problem into a binary classification problem where $y'_1, \ldots, y'_n \in (0, 1)$ and which allows us to then apply the standard IVAP method (as summarised in Algorithms 2-6 in [7]) to the set $(s_1, \ldots, s_n, y'_1, \ldots, y'_n)$ in order to precompute a set of functions $(f_0(s_1, \ldots, s_n), f_1(s_1, \ldots, s_n))_i$ for a given threshold y_i above. After applying this step recursively a total of n times for each y_i we can precompute the full probability distribution $(f_0(s_1, \ldots, s_n), f_1(s_1, \ldots, s_n))_{i=1,\ldots,n}$.

When a new example is presented to us from the test set with a score s_t our goal is to find a probability distribution for its label $P(y_t)$. In order to do so we use a binary search tree algorithm (as in [7]) to derive the corresponding values $(p_0, p_1)_{i=1,\ldots,n} = (f_0(s_t), f_1(s_t))_{i=1,\ldots,n}$. The corresponding probability P_i (derived using the minimax procedure as described by Equation 4) gives the cumulative probability $P(y_t > y_i)$, for each y_i in the calibration set.

Within the standard machine learning framework the n examples described above can originate either from the training or the calibration set. To keep this new algorithm in the same spirit of the work described in [7] we assume that the we are presented with both a training and a calibration set and the latter is used to precompute $(f_0(s_1, \ldots, s_n), f_1(s_1, \ldots, s_n))_{i=1,\ldots,n}$ where the indices i now refer to the individual examples in the calibration set. This procedure is summarised in Algorithm 2 (referred to as the IVAPR algorithm).

Algorithm 2 Inductive Venn–Abers regression (IVAPR) algorithm

Input: calibration sequence consisting of scores and labels $(z_1, \ldots, z_n) = (s_1, \ldots, s_n; y_1, \ldots, y_n)$ for the underlying algorithm

Input: test sequence of scores $(s_1, \ldots, s_t)$ for the underlying algorithm

Output: a set of probabilistic outputs $P_j(y > y_i)$ for each element j in the test set and each y_i in the calibration set $(y_1, \ldots, y_n)$

1: order the calibration sequence $(z_1, \ldots, z_n)$ by values of $(s_1, \ldots, s_n)$ in increasing order such that $(z_1, \ldots, z_n) \longrightarrow (z_{\pi(1)}, \ldots, z_{\pi(n)})$ with $s_{\pi(1)} < s_{\pi(2)}, \ldots < s_{\pi(n)}$

2: order the calibration sequence $(z_1, \ldots, z_n)$ by values of $(y_1, \ldots, y_n)$ in decreasing order such that $(z_1, \ldots, z_k) \longrightarrow (z_{\mu(1)}, \ldots, z_{\mu(n)})$ with $y_{\mu(1)} > y_{\mu(2)}, \ldots > y_{\mu(n)}$ and remember the mapping $\pi_i \longrightarrow \mu_j$

3: map the real valued set $(y_{\mu(1)}, \ldots, y_{\mu(n)})$ to new binary set set $(y'_{\mu(1)}, \ldots, y'_{\mu(n)}) = 0$

4: generate new calibration set $(s_{\pi(1)}, \ldots, s_{\pi(n)}; y'_{\pi\to\mu(1)}, \ldots, y'_{\pi\to\mu(n)})$

5: **for** each $y'_{\mu(i)} \in (y'_{\mu(1)}, \ldots, y'_{\mu(n)})$ **do**

6: set $'y_{\mu(i)} = 1$

7: set $(s_{\pi(1)}, \ldots, s_{\pi(n)}; y_{\pi\to\mu(1)}, \ldots, y_{\pi\to\mu(n)})$ as an input into the standard IVAP algorithm described in [7]

8: derive $(f_0, f_1)_i$ for $y_i \longleftarrow y_\pi \longleftarrow y_\mu$

9: **for** each s_j in the test set $(s_1, \ldots, s_t)$ **do**

10: find the corresponding payload using the binary search tree algorithm given by Algorithm 6 in [7]

11: use $(p_0, p_1)_{i=1,\ldots,n} = (f_0(s_t), f_1(s_t))_{i=1,\ldots,n}$to generate $P_j(y > (y_1, \ldots, y_n))$

4.1 Computational efficiency

The key steps in Algorithm 2 are precomputation which consists of sorting (steps 1 and 2) which compute in time $O(nlog(n))$ each followed by a loop (steps 5-8) in which the IVAP algorithm is recursively applied for each possible threshold given by values of y in the calibration step (which computes in time $O(n) \times O(n) = O(n^2)$) where n is the total number of examples in the calibration set. Given the sorted scores of the calibration objects, its application to each test object takes time $O(\log t)$ where t is the number of examples in the test set.

5 Empirical results

An example of Algorithm 2, developed in Python, uses a training dataset with $n = 100$ training points generated from a known probability distribution where $(s_1, \ldots, s_n) = (1, \ldots, 100)$ and $y = s$, i.e. the underlying algorithm is perfectly calibrated. This very preliminary test forms a "sanity check" to see whether the IVAPR derived distributions for different test points corresponds to the one expected by the (known) probability distribution generating the examples. Figure 1 shows the derived cumulative probability distributions $P(y_t > y_n)$ and the probability density functions ($\frac{dP(y_t>y)}{dy}$) for test points $s_t = 30$ and $s_t = 60$ respectively calculated using Algorithm 2. We can see that the maxima correspond to the values of $s_t = y$ as expected which suggests that the algorithm likely produces valid prediction sets (needs further proof). The results above have been compared with those derived using the less computationally efficient PAVA Venn–Abers algorithm described in [6] and they are identical.

In order to test the Algorithm further on real-life datasets a set of experiments was run similar to those reported in [4]. In particular the datasets are:

- Facebook Metrics [3] dataset contains information about 500 Facebook posts. The goal is to predict the number of Total Interactions depending on the user behavior in a social network.
- Energy Efficiency [5] dataset contains energy analysis using 12 different simulated building shapes (totally 768 instances). The predicted value is Heating Load.

Each dataset was split into a training set (which was used to train the underlying algorithm and calibrate the scores) and a test set in a ratio of $2 : 1$. The duplicate test scores

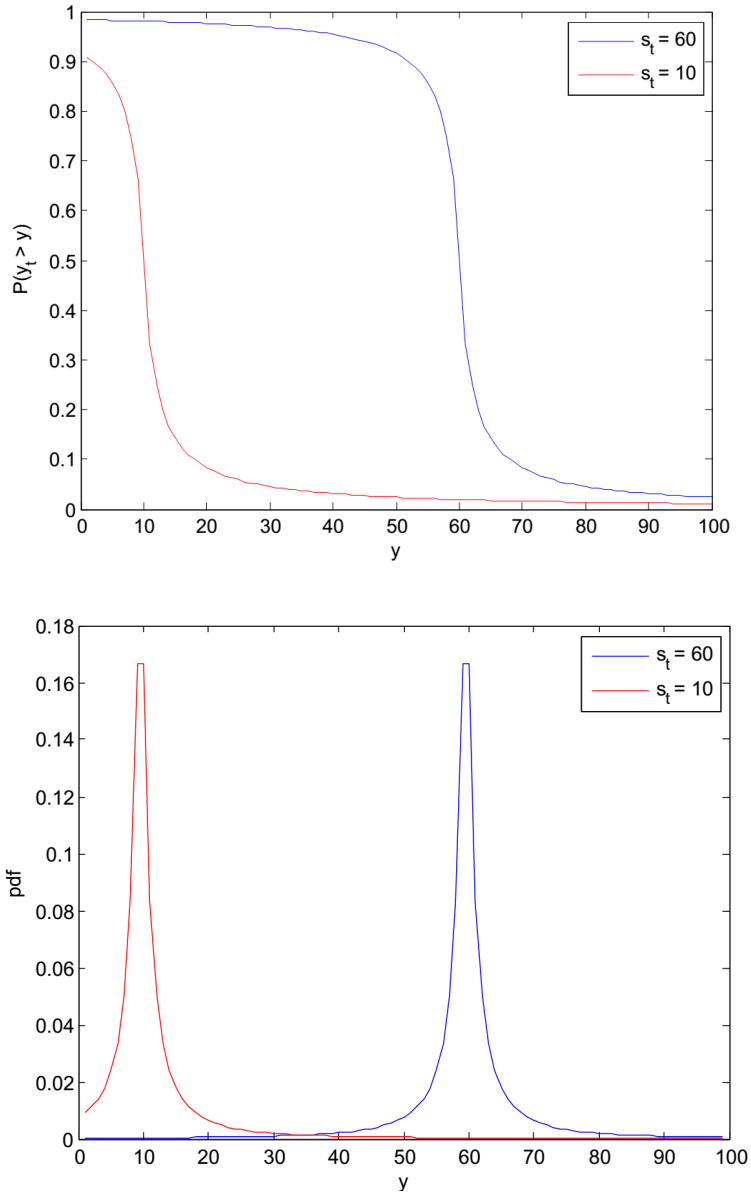

Figure 1: (a) Cumulative probability distributions and (b) probability density functions for an artificial dataset with known $y = s$ and test examples $s_t = 10$ and $s_t = 60$ respectively

were processed using the method analogous to the one reported in [7] Section 2. Each dataset was randomly permuted and the experiments repeated a total of 100 times.

The results of validity tests are summarised in Figure 2. The coverage on y axis means the percentage of test objects for which the predicted confidence covers the real value. The confidence level is the complement to 1 of the significance level and expresses the (chosen) probability that the confidence interval covers the true value of the label. As an example for a significance level of 90% and a corresponding confidence level of 10% (which corresponds to the point 0.1 on the x axis in Figure 2) the coverage is the percentage of all test points for which the cumulative probability $P(y > y_t)$ where y_t is the true test label. For reference, the points along the diagonal correspond to ideal validity. We can ee that the IVAPR algorithm generates on average values close to the ideal for both data sets.

A further experiment was performed where the true labels were set as a randomly permuted artificial dataset with 500 elements $(1, \ldots, 500)$ and the predicted labels were set to the true label plus an added element of Gaussian noise of $\mathcal{N}(0, 1)$. The dataset was split in a ratio of $3 : 2$ and the aim of the IVAPR algorithm was to predict the most likely value for the label for the test point s_j chosen as y_t such that $P_j(y > y_t)$ in line 11 of Algorithm 2 was closest to the median value of 0.5. Figure 3 shows the scatter plot of the estimated and true labels for the 200 points in the test set. We can see that the algorithm seems to output values very close to the true label with little or no bias.

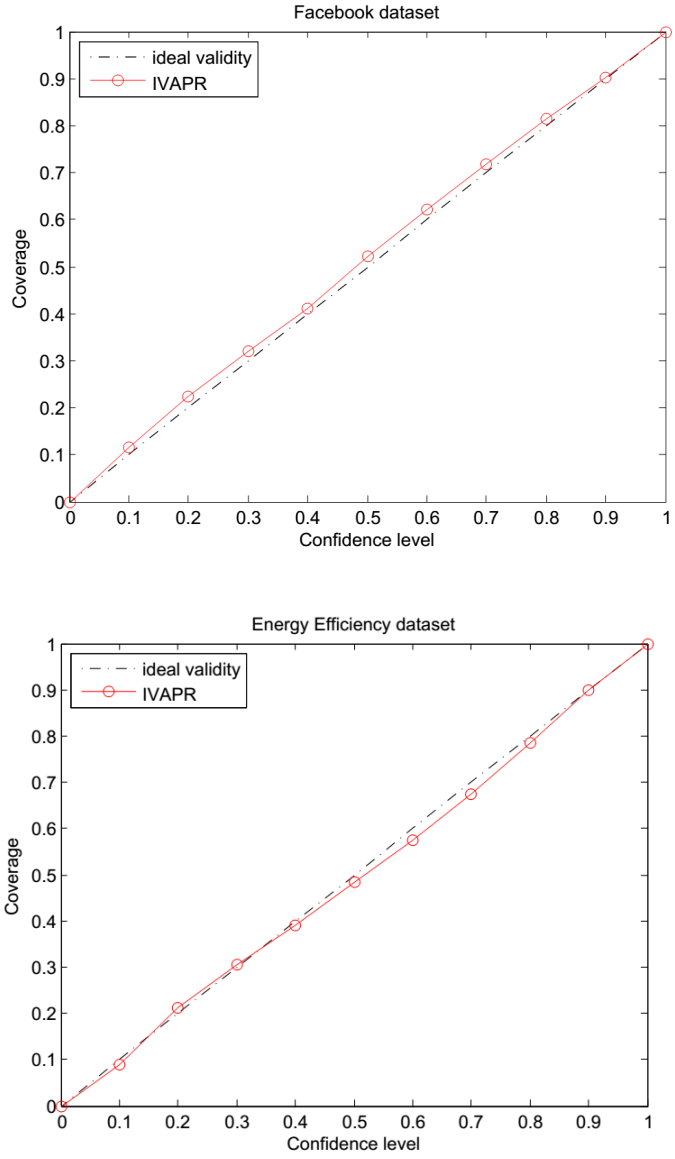

Figure 2: Coverage for different confidence levels produced by IVAPR on for (a) Facebook and (b) Energy Efficiency datasets

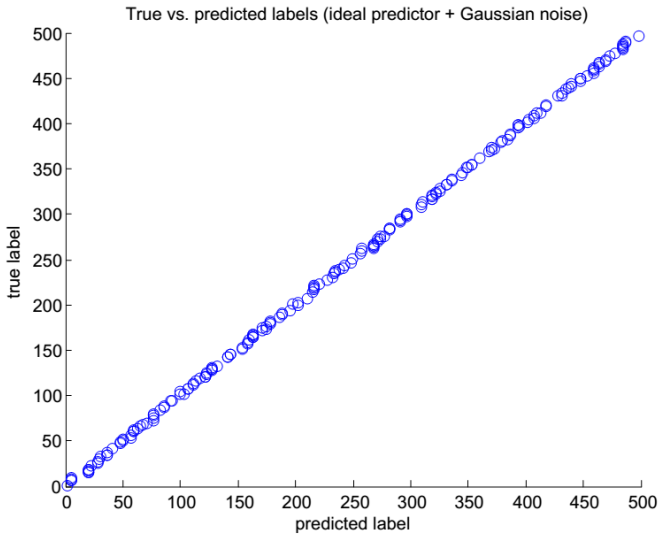

Figure 3: Estimated and true labels for the 200 points in the test set for an artificial dataset where the estimated label is equal to the true label with added Gaussian noise

In order to test the IVAPR algorithm on furthers, a total of 5 publicly available regression problem datasets from the UCI Repository was obtained: Machine Data (Machine), Concrete Data (Concrete), Forest Fires (Forest), Facebook Metric (Facebook) and Energy Efficiency (Energy). Five different algorithms were chosen as a basis for comparison: linear regression (Linear), polynomial regression (Poly), decision tree regression (Tree), random forest regression (RF) and support vector regression (SVR). Each dataset was randomly permuted and divided into a training set (70%) and test set (30%). Each underlying algorithm was then trained on the training set using 3 fold cross-validation with a range of hyperparameters and the optimal model then used to make the predictions on the test set. The predicted and the actual values for each test set and algorithm were then fed into the IVAPR algorithm which generated a set of probabilistic predictions for each dataset/algorithm combination. The results were compared using the *continuous ranked probability score* (CRPS) [8] given by:

$$CRPS(F, y_i) = \int_{-\infty}^{\infty} (F(y) - \mathbf{1}_{y>y_i})^2 dy$$

where $\mathbf{1}$ refers to the indicator function.

The results are summarised in Table 1. We can see that the IVAPR algorithm performs better on average than the simple point prediction of the underlying algorithms.

Table 1: Continuous ranked probability scores for five underlying algorithms (A) and the IVAPR method (IVAPR) applied to five different datasets. The lowest values for each algorithm/dataset are shown in bold

	Linear		Poly		Tree		RF		SVR	
	A	*IVAPR*	*A*	*IVAPR*	*A*	*IVAPR*	*A*	*IVAPR*	*A*	*IVAPR*
Machine	0.069	**0.052**	0.245	**0.217**	0.033	**0.030**	0.026	**0.022**	0.318	**0.279**
Concrete	0.140	**0.095**	0.105	**0.076**	0.089	**0.083**	0.065	**0.051**	0.075	**0.059**
Forest	0.326	**0.136**	0.430	**0.171**	0.399	**0.362**	0.214	**0.139**	0.231	**0.144**
Facebook	**0.005**	0.007	0.075	**0.046**	0.078	**0.069**	0.013	**0.010**	0.041	**0.032**
Energy	0.069	**0.039**	0.029	**0.018**	0.023	**0.022**	0.019	**0.015**	0.018	**0.016**

6 Conclusion

This report described a preliminary version into an alternative Inductive Venn–Abers algorithm applied to the problem of regression. A preliminary empirical study using artificial and real datasets suggests that the method could be complementary to the one reported in [4].

References

[1] Miriam Ayer, H Daniel Brunk, George M Ewing, William T Reid, and Edward Silverman. "An empirical distribution function for sampling with incomplete information". In: *The Annals of Mathematical Statistics* 26.4 (1955), pp. 641–647.

[2] Daniel H Brunk, Richard E Barlow, Daniel J Bartholomew, and James M Bremner. *Statistical inference under order restrictions (the theory and application of isotonic regression)*. Tech. rep. DTIC Document, 1972.

[3] Sérgio Moro, Paulo Rita, and Bernardo Vala. "Predicting social media performance metrics and evaluation of the impact on brand building: A data mining approach". In: *Journal of Business Research* 69.9 (2016), pp. 3341–3351.

[4] Ilia Nouretdinov, Denis Volkhonskiy, Pitt Lim, Paolo Toccaceli, and Alexander Gammerman. "Inductive Venn-Abers Predictive Distribution". In: *Proceedings of Machine Learning Research* 91 (2018), pp. 1–22.

[5] Athanasios Tsanas and Angeliki Xifara. "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools". In: *Energy and Buildings* 49 (2012), pp. 560–567.

[6] Vladimir Vovk and Ivan Petej. "Venn-Abers predictors". In: *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2014, pp. 829–838.

[7] Vladimir Vovk, Ivan Petej, and Valentina Fedorova. "Large-scale probabilistic predictors with and without guarantees of validity". In: *Advances in Neural Information Processing Systems*. 2015, pp. 892–900.

[8] Vladimir Vovk, Ilia Nouretdinov, Valery Manokhin, and Alexander Gammerman. "Cross-conformal predictive distributions". In: *Conformal and Probabilistic Prediction and Applications*. 2018, pp. 37–51.

[9] Bianca Zadrozny and Charles Elkan. "Transforming classifier scores into accurate multiclass probability estimates". In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2002, pp. 694–699.