Shapley-Value based Inductive Conformal Prediction

William Lopez Jaramillo WILLIAM.LOPEZ-JARAMILLO@TATASTEELEUROPE.COM
Tata Steel Europe, The Hague, The Netherlands
Evgueni Smirnov SMIRNOV@MAASTRICHTUNIVERSITY.NL
Maastricht University, Maastricht, The Netherlands

Editor: Lars Carlsson, Zhiyuan Luo, Giovanni Cherubin and Khuong An Nguyen

Abstract

Shapley values of individual instances were recently proposed for the problem of data valuation. They were defined as the average marginal instance contributions to the performance of a given predictor. In this paper we propose to use Shapley values of individual instances as conformity scores. To compute these values efficiently and exactly we employ a standard algorithm based on nearest neighbor classification and propose a variant of this algorithm for clustered data. Both variants are used for computing Shapley conformity scores for inductive conformal predictors. The experiments show that the Shapley-value conformity scores result in smaller prediction sets for significance level $\epsilon \leq 0.1$ compared with those produced by standard conformity scores (i.e. similarity between true and predicted output values).

Keywords: Inductive Conformal Prediction, Shapley values, Conformity Scores

1. Introduction

Most of the classification methods are focused on point estimation when a new test instance x receives "the best guess", i.e. one class estimate (Johansson et al., 2013; Vanderlooy et al., 2006). In critical-domain applications, however, when the cost of misclassification is prohibitively high we need to estimate instead a prediction set Γ that contains possible classes for x given the data. There exist several types of set predictors proposed. The two desired properties of these predictors are validity and informational efficiency. A set predictor is valid if the probability that the set Γ does not contain the class for the test instance x is at most ϵ, the chosen significance level, and a set predictor is efficient if the set Γ is non-empty and small.

The conformal framework proposed in (Vovk et al., 2005) allows us designing set predictors that are automatically valid. To decide whether to include class y in the set Γ for a test instance x the framework suggests that we first compute conformity scores of the training instances and provisionally labeled instance (x, y). Then, the p-value p_y for class y is calculated as the proportion of the instances whose conformity scores are smaller than or equal to that of the instance (x, y). If $p_y > \epsilon$ for a chosen significance level ϵ, the class y is added to the set Γ for the instance x.

A conformity score for a labeled instance is a score that indicates how typical is the instance against the remaining instances in the data. Conformity functions usually are designed for concrete point predictors that are used. There exist several schemes to construct conformity functions: a general scheme and several specific based on concrete types of point predictors.

In this paper we propose a new scheme to construct conformity functions based on the concept of the Shapley value (Shapley, 1953). The conformity score of an instance is defined as a data Shapley value (Jia et al., 2019b), (Ghorbani and Zou, 2019): the average marginal contribution of the instance for the performance of a point predictor used. Since the performance is measured on validation instances, the instance Shapley value indicates how typical this instance is for these instances, i.e. it is indeed a conformity score.

The proposed scheme for conformity functions has two main advantages. First, it is a general scheme: it can be applied for any type of point predictors. This is due to the fact that there exists a long repertoire of general purpose Shapley-value algorithms: from brute-force to Monte-Carlo approximations. Second, our scheme allows computing instance Shapley values (conformity scores) using different metrics for predictor performance like accuracy rate, ROC-AUC, F-score etc. Thus, we can compute the conformity scores w.r.t. different aspects of the classification tasks in hand and predictors used[1].

To compute Shapley-value conformity scores efficiently and exactly we employ in this paper a standard algorithm based on nearest neighbor classification (Jia et al., 2019b). We realize that this algorithm has tendency to produce extreme Shapley values in the presence of clustered data. To avoid this problem we propose a new variant of this algorithm. We use both algorithms for computing Shapley conformity scores for inductive conformal predictors (Papadopoulos et al., 2002). The experiments show that the Shapley-value conformity scores result in smaller prediction sets for significance level $\epsilon \leq 0.1$ compared with those produced by standard conformity scores[2] (Vovk et al., 2005).

The rest of the paper is organized as follows. In Section 2 we formalize the classification task in the context of point estimation and prediction-set estimation. The conformal prediction and its basic set predictors are presented in Section 3. Section 4 provides related work in conformity function design. Shapley values for data valuation are introduced in Section 5 together with the algorithm for exact Shapley values based on nearest neighbor classification (Jia et al., 2019a). In Section 6 we propose a new variant of this algorithm for clustered data. The Shapley-value scheme for conformity functions is introduced in Section 7. We show how to use the scheme for the inductive conformal predictors using both algorithms for exact Shapley values. The experiments are provided in Section 8. Section 9 concludes the paper.

2. Classification

Let X be an instance space, Y be a finite discrete class variable, and P be a probability distribution over $X \times Y$. The training data set T is a multi set of M instances $(x_m, y_m) \in X \times Y$ drawn from the distribution P under the randomness assumption. In this context, we can define two possible tasks: point classification task and prediction-set classification task.

The point classification task is to find an estimate $\hat{y} \in Y$ of the class for a test instance $x \in X$ according to P. To solve the task we first learn a point predictor h in a hypothesis

1. For example, if we use AUC as a metric for predictor's performance to compute Shapley-value conformity scores, then we can use any standard conformal predictor for class imbalanced problem; i.e. we will not need the Mondrian conformal prediction.
2. We use a similarity between true and predicted output values for standard conformity scores.

space H using the training data set T. The predictor h is a function of type $h : X \rightarrow \mathbb{R}^{|Y|}$. It outputs for the test instance x a posterior distribution of scores $\{s_y\}_{y \in Y}$ over all the classes in Y. The class y with the highest posterior score s_y is the estimated class $\hat{y}$ for the instance x.

The prediction-set classification task is to estimate a prediction set $\Gamma(T, x) \subseteq Y$ that contains possible classes for a test instance $x \in X$ according to P. To solve the task we need a set predictor. The two most desired properties of a predictor are validity and informational efficiency. A set predictor is said to be valid iff the probability that the prediction set $\Gamma(T, x) \subseteq Y$ does not contain the class for the test instance x is at most the chosen significance level $\epsilon \in [0, 1]$. A set predictor is said to be informationally efficient if the prediction set $\Gamma(T, x) \subseteq Y$ is non-empty and small. In the next section we briefly introduce the conformal framework that is used for designing valid set predictors (Shafer and Vovk, 2008; Vovk et al., 2005).

3. Conformal Prediction

The conformal framework allows designing set predictors that are automatically valid when the training data is drawn under the exchangeability assumption [3]. Given a point predictor h, a conformal set predictor computes the prediction set $\Gamma^{\epsilon}(T, x) \subseteq Y$ for any test instance x and any significance level $\epsilon \in [0, 1]$ as follows. To decide whether to include class $y \in Y$ in $\Gamma^{\epsilon}(T, x)$, it first provisionally considers the labeled instance (x, y). Then, the conformal set predictor computes conformity scores of the instances in training data set T and test instance (x, y). It computes the p-value p_y of class y for x as the proportion of the instances in $T \cup \{(x, y)\}$ of which the conformity scores are smaller than or equal to that of the instance (x, y). The class y is added to the final prediction-set for the instance x if $p_y > \epsilon$, where ϵ is a chosen significance level in $[0, 1]$.

A conformity score for a labeled instance is a score that indicates how typical is the instance w.r.t. the instances in the data. To compute the score, we need a conformity function B. Formally, a conformity function is of type $B : (X \times Y)^{(*)} \times (X \times Y) \rightarrow \mathbb{R}^{+} \cup \{+\infty\}$ [4]. Given a data set $T \in (X \times Y)^{(*)}$ and an instance $(x, y) \in (X \times Y)$, it returns a conformity score $\beta \in \mathbb{R}^{+} \cup \{+\infty\}$ indicating how typical the instance (x, y) is for the instances in T. Usually if $(x, y) \in T$, the conformity score for that instance is computed for the instances in $T \setminus \{(x, y)\}$. This means that when we compute the p-value p_y of class y for a test instance x the conformity scores for all the instances from $T \cup \{(x, y)\}$ are computed in a leave-one-out manner. This computation is implemented in the transductive conformal predictor (TCP) (Vovk et al., 2005).

Conformity functions usually are designed w.r.t. the concrete point predictors that are used (see next Section for details). The validity of TCP is proven in (Vovk et al., 2005) to be insensitive to the type of the conformity functions used.

We note that often it is more convenient to compute nonconformity scores instead of the conformity ones. They are defined in an opposite manner: a nonconformity score for a labeled instance is a score that indicates how untypical is the instance w.r.t. the instances in the data. In this case, the prediction sets $\Gamma^{\epsilon}(T, x)$ are computed analogously: the only

3. The exchangeability assumption is weaker than the randomness assumption.
4. $(X \times Y)^{(*)}$ denotes the set of all multi sets defined over $X \times Y$.

difference is that the p-value p_y of class y for any test x is calculated as the proportion of the instances in $T \cup \{(x, y)\}$ of which the nonconformity scores are greater than or equal to that of the instance (x, y).

The TCP predictor described above is very accurate since it is essentially a transductive predictor. However, due to the leave-one manner of computing the conformity scores, the computationally complexity is high: it is proportional to the product of the number of training instances and the complexity of training predictor h. Papadopoulos et al. (2002) addressed this computational problem by developing inductive conformal predictor (ICP). This predictor in addition to the training data set T employs a validation data set V with size N. The training data is used to train a point predictor that is used for implementing the conformity function B. The validation data is used to computing conformity scores for the validation instances only. In this way the overall computational complexity decreases (see below).

The ICP predictor has a training algorithm and a classification algorithm. The training algorithm is presented in Algorithm 1. The algorithm input consists of a training data set T, a validation data set V, and a point predictor h. The training algorithm first trains h on T and builds the conformity function B based on h (line 1). Then, it computes the conformity score β_n for each instance (x_n, y_n) in the validation data set V using the function B (lines 2-4).

The classification algorithm is presented in Algorithm 2. The algorithm input consists of a significance level $\epsilon \in [0, 1]$, a point predictor h, a conformity function B for h, conformity scores $\{\beta(n)\}_{n=1}^{N}$, and a test instance $x \in X$. The algorithm constructs a prediction set $\Gamma^{\epsilon}(T, x) \subseteq Y$ for the test instance x as follows. To decide whether to include a class $y \in Y$ in the prediction set $\Gamma^{\epsilon}(T, x)$, the instance x and class y are first combined into a labeled instance (x, y). Then, the predictor computes the conformity score β_{N+1} for (x, y) w.r.t. the training data set T using the conformity function B for the point predictor $h(x)$ (line 3). The conformity score is used for computing the p-value p_y of the class y for the instance x (line 5). More precisely, p_y is computed as the proportion of the instances in the validation data set V of which the conformity scores β_n are smaller or equal to that of the instance (x, y). Once p_y has been set, the algorithm includes the class y in the prediction set $\Gamma^{\epsilon}(T, x)$ if $p_y > \epsilon$ (line 7).

Algorithm 1 Inductive Conformal Predictor: Training

```
Input:   Training data set $T$ of size $M$, Validation data set $V$ of size $N$,
         Point predictor $h \in H$.
Output:  Point predictor $h$ trained on $T$, Conformity function $B$ for $h$,
         Conformity scores $\{\beta(n)\}_{n=1}^{N}$.
1: Train predictor $h$ on the training data set $T$ and build conformity function $B$ for $h$;
2: for $n := 1$ to $N$ do
3:    Compute conformity score $\beta_n$ for validation instance $(x_n, y_n) \in V$ equal to
      $B(T, (x_n, y_n))$;
4: Output predictor $h$, conformity function $B$, and conformity scores $\{\beta(n)\}_{n=1}^{N}$.
```

Algorithm 2 Inductive Conformal Predictor: Classification

Input: Significance level ϵ, Point predictor $h \in H$, Conformity function B for h, Conformity scores $\{\beta(n)\}_{n=1}^{N}$, Test instance x.

Output: prediction set $\Gamma^{\epsilon}(T, x)$ for test instance x.

```
1: $\Gamma^{\epsilon}(T, x) := \emptyset$;
2: **for each** class $y \in Y$ **do**
3:     Compute conformity score $\beta_{N+1}$ for the test instance $(x, y)$ equal to $B(T, (x, y))$;
4:     $p_y := \frac{\#\{n=1,\ldots,N+1 | \beta_n \leq \beta_{N+1}\}}{N+1}$;
5:     Include $y$ in $\Gamma^{\epsilon}(T, x)$ if $p_y > \epsilon$;
6: **Output** prediction set $\Gamma^{\epsilon}(T, x)$.
```

The classification algorithm of ICP is fast. Its computational complexity is $O(C(B+N))$ where C is the number of classes and B is the complexity of the conformity function.

4. Related Work on Conformity Functions

There exist several schemes to construct conformity functions. We consider a general scheme and several specific based on concrete types of point predictors.

The general scheme to construct a conformity function B for a point predictor $h(x)$ was provided in (Vovk et al., 2005). Given an instance (x, y), the function outputs the value $\Delta(y, h(x))$ for (x, y) where $\Delta : Y \times Y \to \mathbb{R}$ is a similarity function. For example, in our experiments the general conformity function B is defined to output the probability estimate $\hat{p}(y|x)$ provided by h for x. It is employed in ICPs used as baselines.

The Lagrange multipliers α of support vector machines were proposed for non-conformity scores in (Saunders et al., 1999). The higher the value of the multiplier of an instance, the closer the instance is to the SVM hyperplane. Thus, instances with higher (lower) Lagrange multipliers, are less (more) typical which means that $-\alpha$ can be used as a conformity score.

The nearest-neighbor conformity function is one of the most used predictor dependent schemes (Shafer and Vovk, 2008). It outputs for an instance (x, y) a conformity score β equal to $\frac{D_K^{-y}}{D_K^{y}}$, where D_K^{y} is the sum of distances between x and K nearest neighbors of x that belong to class y, and D_K^{-y} is the sum of distances between x and K nearest neighbors of x that do not belong to class y.

The boosting conformity function was proposed in (Moed and Smirnov, 2009; Smirnov et al., 2009; Zhou et al., 2015). It outputs for an instance (x, y) a conformity score β equal to $-w$ where w is the weight of (x, y) in the final iteration of a boosting algorithm. We note that w indicates the classification difficulty of (x, y). Hence, $-w$ indicates how typical (x, y) for the remaining instances.

The random-forest conformity function was proposed in (Devetyarov and Nouretdinov, 2010). It outputs for an instance (x, y) a conformity score β equal to the proportion of the correct out-of-bag classifications that this instance receives.

In this paper we propose a new general scheme to construct conformity functions. The scheme is based on Shapley values of instances and can be computed w.r.t. different performance criteria.

5. Shapley Values for Data Valuation

The Shapley value is an important concept in cooperative game theory (Shapley, 1953) that received a lot of attention in machine learning (Strumbelj and Kononenko, 2014; Lundberg and Lee, 2017). Recently, the Shapley value was employed for the problem of data valuation: to determine the contribution of any training instance for the performance of a given point predictor h (Jia et al., 2019a). The key idea is to consider a cooperative game specified by "players", i.e. all the instances in the training data T, and a characteristic function $v(V)$ that estimates the performance of the predictor h on a validation set V when trained on T [5]. In this context, the Shapley value $s(m)$ for any training instance $(x_m, y_m) \in T$ is defined as follows:

$$s(m) = \frac{1}{M} \sum_{S \subseteq T \setminus \{(x_m, y_m)\}} \frac{1}{\binom{M-1}{|S|}} \big(v(S \cup \{(x_m, y_m)\}) - v(S)\big). \tag{1}$$

The Shapley value is essentially the average marginal contribution of the training instance (x_m, y_m) for the performance of predictor h estimated by the characteristic function $v(T)$ when trained over all possible subsets $S \subseteq T$. It has three desirable properties:

- Group Rationality: $v(T) = \sum_{m=1}^{M} s(m)$.
- Fairness: $(\forall S \subseteq T)\big(v(S \cup \{(x_m, y_m)\}) = v(S \cup \{(x_n, y_n)\}) \to s(m) = s(n)\big)$.
- Additivity: $(\forall m)(s_1(m) + s_2(m) = s_{12}(m))$, where
  - $s_1(m)$ is Shapley value for instance $(x_m, y_m) \in T$ if we use characteristic function v_1,
  - $s_2(m)$ is Shapley values for instance $(x_m, y_m) \in T$ if we use characteristic function v_2,
  - $s_{12}(m)$ is Shapley values for instance $(x_m, y_m) \in T$ if we use characteristic function $v_1 + v_2$.

The characteristic function $v(V)$ can be any function that estimates the predictor performance. It can output, for example, accuracy rate, ROC-AUC, TPr, F1 score, etc. This means that Shapley values of the training instances can be related to different aspects of the classification tasks in hand and predictors used. In this context we note that these values can be positive or negative. Ghorbani and Zou (2019) proposed to remove instances with negative Shapley values to boost the predictors' performance.

Computing Shapley values for all the instances in the training data set T can be realized in numerous ways. The brute-force algorithm follows formula (1) and thus is computationally expensive: its time complexity is $O(2^M)$. There exist several approximation algorithms

5. Our notation for the characteristic function v assumes that the predictor h and validation set V are fixed. However, if this is not the case, the characteristic function can be given by $v(h, T, V)$.

based on Monte-Carlo simulation (Jia et al., 2019a)(Ghorbani and Zou, 2019). Although they have an acceptable computational complexity, they do not guarantee to find exact Shapley values.

Recently, (Jia et al., 2019a) proposed an algorithm for computing exact Shapley values for nearest neighbor classification (ESVNN). *The algorithm was designed under the assumption that the instance space X is a metric space.* It was shown to be more computationally efficient than any of the predecessors.

To introduce the ESVNN algorithm let us consider a validation instance $(x_n, y_n) \in V$. The training instances $(x_m, y_m) \in T$ can be ordered in an increasing order of their distance to the instance (x_n, y_n). Let us consider the first K of the ordered training instances, i.e. instances $(x_{\pi(1)}, y_{\pi(1)}), (x_{\pi(2)}, y_{\pi(2)}), \ldots, (x_{\pi(K)}, y_{\pi(K)})$. They can be used to estimate the probability $p(y_n|x_n)$ as $\frac{1}{K}\sum_{k=1}^{K} \mathbb{1}[y_{\pi(k)} = y_n]$, where $\mathbb{1}$ is the indicator function. This estimated probability can be viewed as the characteristic value $v_n(T)$ of the K-nearest neighbor classifier for the nth validation instance. More formally,

$$v_n(T) = \frac{1}{K} \sum_{k=1}^{min(K,|V|)} \mathbb{1}[y_{\pi(k)} = y_n]. \tag{2}$$

If we sum the characteristic values $v_n(T)$ over all the validation instances, then we receive the characteristic value $v(T)$ of the K-nearest neighbor classifier for the whole validation set V of size N:

$$v(T) = \frac{1}{N} \sum_{n=1}^{N} v_n(T). \tag{3}$$

The ESVNN algorithm is an algorithm for computing exact Shapley values for the characteristic function defined in formula (3). It is presented in Algorithm 3. The algorithm first computes Shapley value $s_n(m)$ of every training instance $(x_m, y_m) \in T$ for every validation instance $(x_n, y_n) \in V$. This is realized as follows: for any validation instance (x_n, y_n) the algorithm sorts the training instances (x_m, y_m) in an increasing order of their distance to (x_n, y_n) (lines 1-2). Then (in lines 3-7) it visits each instance of the sorted sequence $(x_{\pi(1)}, y_{\pi(1)}), (x_{\pi(2)}, y_{\pi(2)}), \ldots, (x_{\pi(M)}, y_{\pi(M)})$ in reverse order and assigns the Shapley values $s_n(m)$ according to the following recursive rule:

$$\begin{aligned} s_n(\pi(M)) &= \frac{\mathbb{1}[y_{\pi(M)} = y_n]}{M}, & (4) \\ s_n(\pi(m)) &= s_n(\pi(m+1)) + \frac{\mathbb{1}[y_{\pi(m)} = y_n] - \mathbb{1}[y_{\pi(m+1)} = y_n]}{K} \frac{min(K, m)}{m}. & (5) \end{aligned}$$

Once Shapley values $s_n(m)$ of all the training instances $(x_m, y_m) \in T$ have been computed for all validation instances $(x_n, y_n) \in V$, the algorithm computes (in lines 8-10) the final Shapley value $s(m)$ of any training instance $(x_m, y_m) \in T$. The latter is computed following the additive property as the average of the Shapley values $s_n(m)$ over all the validation instances $(x_n, y_n) \in V$.

The ESVNN algorithm is a computationally efficient algorithm. *It computes Shapley values independently on the order of the instances in the training and validation data sets T*

and V. The time complexity is $O(NM\log(M))$. This is due to the most expensive operation (line 2, Algorithm 3) when we sort the training instances for each validation instance.

Algorithm 3 Algorithm for Exact Shapley Values based on Nearest Neighbor Classification

```
Input:   Training data set $T$ and Validation data set $V$,
Output:  Shapley values for all the training instances: $\{s(m)\}_{m=1}^{M}$.

1: for $n := 1$ to $N$ do
2:     Find permutation $(\pi(1), \pi(2), \ldots, \pi(M))$ that defines a sorted sequence of the training
       instances in an increasing order of their distance to the validation instance $(x_n, y_n)$;
3:     $s_n(M) := \frac{\mathbb{1}[y_{\pi(M)}=y_n]}{M}$;
4:     for $m := M-1$ to $1$ do
5:         $s_n(\pi(m)) := s_n(\pi(m+1)) + \frac{\mathbb{1}[y_{\pi(m)}=y_n]-\mathbb{1}[y_{\pi(m+1)}=y_n]}{K}\frac{min(K,m)}{m}$;
6: for $m := 1$ to $M$ do
7:     $s(m) := \frac{1}{M}\sum_{n=1}^{N} s_n(m)$;
8: Output $\{s(m)\}_{m=1}^{M}$.
```

6. Computing Local Exact Shapley Values for Nearest Neighbor Classification

The ESVNN algorithm is a global algorithm for computing Shapley values despite being a nearest-neighbour-type algorithm. This is due to the fact that the Shapley value for any training instance is computed w.r.t. all the validation instances (lines 8-10). Thus, any training instance that belongs to a relatively small (big) but important cluster of a class will receive a relatively low (high) Shapley value. This means that the ESVNN algorithm has a tendency to produce extreme Shapley values for clustered data.

To introduce locality in computing Shapley values we introduce a new version of the ESVNN algorithm. The new algorithm is an algorithm for computing local exact Shapley values for nearest neighbor classification (LESVNN). It has an additional parameter L that sets the local neighborhoods for computing Shapley values as presented in Algorithm 4. The key feature of the algorithm is that the Shapley values $s_n(m)$ for any training instance $(x_m, y_m) \in T$ are computed only for those validation instances $(x_n, y_n) \in V$ whose L-neighborhood contains (x_m, y_m). Thus, the final Shapley value $s(m)$ for (x_m, y_m) is computed as the average of the Shapley values $s_n(m)$ for those validation instances only. In this context, we note that the training instances that are absolute outliers (i.e. they do not belong to the L-neighborhood of any validation instance $(x_n, y_n) \in V$) receive a NaN value.

The time complexity of the LESVNN algorithm is derived analogously to that of the ESVNN algorithm. It is $O(NL\log(L))$. Since the locality parameter L is usually much smaller than M, the LESVNN algorithm is faster in practice. *We note that LESVNN computes Shapley values independently on the order of the instances in the training and validation data sets* T *and* V.

Algorithm 4 Algorithm for Local Exact Shapley Values based on Nearest Neighbor Classification

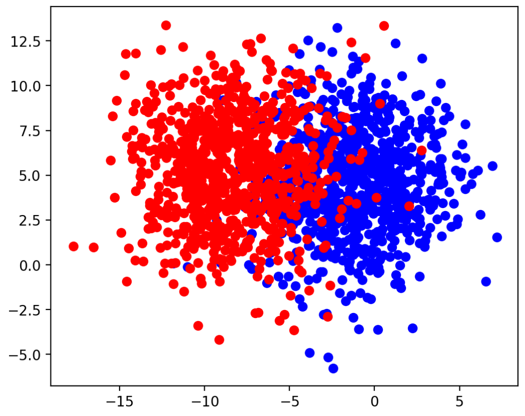
```
Input:  Training data set $T$ and Validation data set $V$, Locality parameter $L$.
Output: Shapley values for all the training instances: $\{s(m)\}_{m=1}^{M}$.

1:  for $n := 1$ to $N$ do
2:      Save the closest $L$ training instances $(x_m, y_m) \in T$ to the validation instance $(x_n, y_n) \in V$ in a training set $T_L$.
3:      Find permutation $(\pi(1), \pi(2), \ldots, \pi(L))$ that defines a sequence of the training instances of $T_L$ in increasing order of their distance to the validation instance $(x_n, y_n)$;
4:      $s_n(L) := \frac{\mathbb{1}[y_{\pi(L)} = y_n]}{L}$;
5:      for $m := L - 1$ to $1$ do
6:          $s_n(\pi(m)) := s_n(\pi(m+1)) + \frac{\mathbb{1}[y_{\pi(m)} = y_n] - \mathbb{1}[y_{\pi(m+1)} = y_n]}{K} \frac{min(K,m)}{m}$;
7:      Set the Shapley values $s_n(m)$ of training instances $(x_m, y_m) \in T \setminus T_L$ equal to $NaN$;
8:  for $m := 1$ to $M$ do
9:      $N_L := \sum_{n=1}^{N} \mathbb{1}[s_n(m) \neq NaN]$;
10:     if $N_L \neq 0$ then
11:         $s(m) := \frac{1}{N_L} \sum_{n=1}^{N} \mathbb{1}[s_n(m) \neq NaN] s_n(m)$;
12:     else
13:         $s(m) := NaN$;
14: Output $\{s(m)\}_{m=1}^{M}$.
```

7. Shapley Values as Conformity Scores

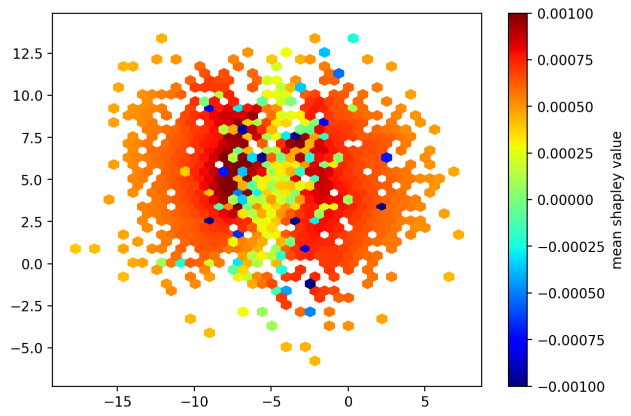
In this paper we propose to use Shapley values of the training instances as conformity scores for conformal prediction. The setting is as follows. Assume that we have a point predictor h built on a training data set T and tested on a validation data set V using a characteristic function $v(T)$. We choose the function to be kind of "positive": it outputs higher values when the performance of the predictor h gets better. In this context, the higher the Shapley value $s(m)$ is for a training instance $(x_m, y_m) \in T$ the more it contributes to correctly predicting the labels of the validation instances $(x_n, y_n) \in V$ drawn from the distribution P. This means that Shapley values indicate typicalness of the training instances for the distribution P. Thus, we can conclude that they can be used as conformity scores.

Figures 1(*a*) and 1(*b*) present the distributions of the Shapley values for a 2-dimensional data set with two classes s.t. each class is given by a Gaussian distribution and the distance between the class means is one standard deviation. The Shapley values were computed by the ESVNN algorithm in a leave-one-out manner for $K = 11$ [6]. From the figure we observe that instances that are closer to the class border have lower Shapley values than the instances closer to the class centers. Similarly the instances that are surrounded by instances of the opposite class receive the lowest Shapley values.

6. This is done s.t. for any $m \in \{1, \ldots, M\}$ instance $(x_m, y_m) \in T$ is considered as a validation instance and the remaining instances in T as training instances.



(a) Two-Class Data



(b) Shapley Values for $K = 11$

Figure 1: Distribution of Shapley Values for Two-Class Data

Shapley-value conformity scores can be used for any type of conformal predictors. Due to the time complexity of computing Shapley values, in this paper we consider Shapley values for the ICP predictors only.

The Shapley-value based inductive conformal predictor (S-ICP) is given in Algorithm 5. Given a significance level ϵ, a training data set T, a validation data set V, a point predictor $h \in H$, a Shapley-value algorithm for h, and a test instance x, the predictor constructs a prediction set $\Gamma^{\epsilon}(T, x) \subseteq Y$ for the instance x as follows. To decide whether to include a class $y \in Y$ in the prediction set $\Gamma^{\epsilon}(T, x)$, the instance x and class y are first combined into a labelled instance (x, y) (line 3). The set T and the instance (x, y) with index $M + 1$ form a new set T'. A Shapley-value algorithm computes the Shapley values for all the instances in T' w.r.t. the validation set V under the assumption that the characteristic function $v(T)$ outputs higher values when the performance of the predictor h gets better (line 4). The Shapley values are considered as conformity scores used for computing the p-value p_y of the class y for the instance x (line 5). More precisely, p_y is computed as the proportion of the instances in the training data set T' of which the Shapley values $\beta(n)$ are smaller or equal to that of the instance (x, y). Once p_y has been set, the predictor includes the class y in the prediction set $\Gamma^{\epsilon}(T, x_{M+1})$ if $p_y > \epsilon$ (line 6).

S-ICP operates with the training data set T and validation data set V in a manner opposite to that of ICP. S-ICP computes the conformity scores for the training data set T while ICP computes the conformity scores for the validation data set V. This means that we expect (1) S-ICP to outperform ICP for large validation sets, and (2) ICP to outperform S-ICP for large training sets. Another difference is that S-ICP is partially transductive; i.e. it recomputes the conformity scores (Shapley values) for any new test instance. This is due to the fact that the existing Shapley value algorithms are batch; i.e. they need to recompute all the Shapley values whenever any new instance is added.

We propose to implement S-ICP using the nearest-neighbor based Shapley-value algorithms from the previous section. When we use the ESVNN algorithm we have global S-ICP predictor (GS-ICP) due to the global manner of computing the Shapley values. When we

Algorithm 5 Shapley-value based Inductive Conformal Predictor

```
Input:    Significance level $\epsilon$,
          Training data set $T$ of size $M$, Validation data set $V$ of size $N$,
          Point predictor $h \in H$, Shapley-value algorithm for $h$,
          Test instance $x$.
Output:   prediction set $\Gamma^{\epsilon}(T, x)$.

1: $\Gamma^{\epsilon}(T, x) := \emptyset$;
2: for each class $y \in Y$ do
3:     $T' := T \cup \{(x, y)\}$; /// The test instance $(x, y)$ receives index of $M + 1$.
4:     Compute the Shapley values $s(m)$ of the instances $(x_m, y_m) \in T'$ w.r.t. data set $V$;
5:     $p_y := \frac{\#\{m=1,\ldots,M+1 | s(m) \leq s(M+1)\}}{M+1}$;
6:     Include $y$ in $\Gamma^{\epsilon}(T, x_{M+1})$ if $p_y > \epsilon$;
7: Output prediction set $\Gamma^{\epsilon}(T, x)$.
```

use the LESVNN algorithm we have local S-ICP predictor (LS-ICP) due to the local manner of computing the Shapley values [7]. We note that both predictors are sensitive to the parameter K of ESVNN and LESVNN: according to formula (5) the Shapley values decrease with K. In addition, LS-ICP is senstive to the locality parameter L of LESVNN: the Shapley values increase with L (check the for loop lines 5-7, Algorithm 4), having a higher range as observed in Figure 2.

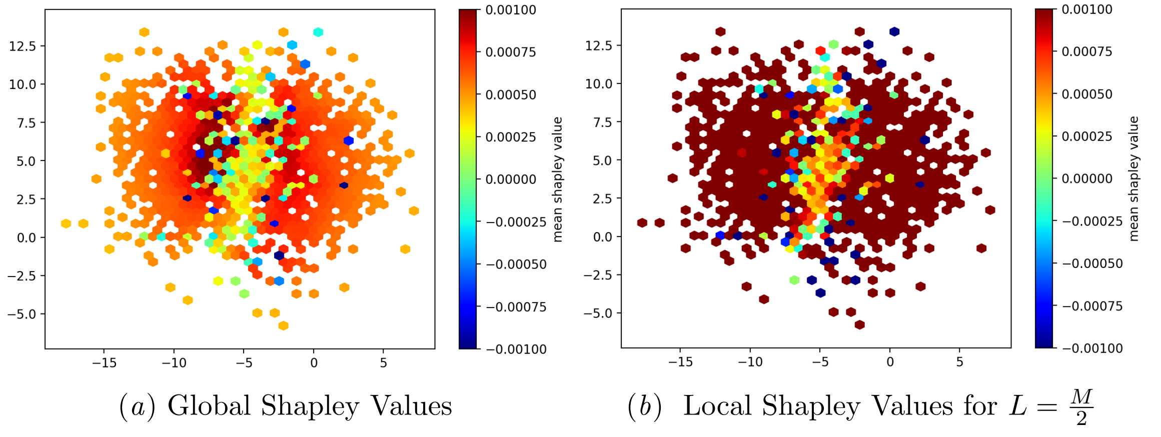


Figure 2: Distribution of Global and Local Shapley Values for Two-Class Data

The GS-ICP and LS-ICP predictors differ in terms of computational complexity and generalization performance. The computational of complexity of GS-ICP is $O(CNM\log(M))$ and that of LS-ICP is $O(CNL\log(L))$ where C is the number of classes. Since the locality parameter L is usually much smaller than M, LS-ICP is faster in practice.

7. LS-ICP computes p-value $p_y = \frac{1}{M+1}$ for any instance (x, y) if LESVNN outputs NaN for that instance.

Dataset	Instances	Attributes	Classes	Majority class
Heartc	303	14	2	54%
Colic	299	26	2	60%
Hepatitis	155	19	2	81%
Ionosphere	351	34	2	65%
Sonar	208	60	2	53%
Glass	214	9	6	35%
Wine	177	13	3	40%
Iris	150	4	3	30%
$Colic_{og}$	299	26	3	60%

Table 1: Dataset characteristics

At the end of this section we would like to note that Shapley values can be used as nonconformity scores as well if we choose the characteristic function $v(T)$ to be kind of “negative”: it outputs higher values when the performance of the predictor h gets worse. Examples include: error rate, FPr, FNr etc. In addition, Shapley values as (non)conformity scores can be estimated w.r.t. different performance metrics like accuracy rate, ROC-AUC, F-score etc. Thus, we can compute the (non)conformity scores w.r.t. different aspects of predictors. For example, if the data is class-imbalanced, we can use ROC-AUC instead of the accuracy rate. Thus, we can use any standard conformal predictor and we will not need the Mondrian conformal prediction.

8. Experiments, Results, and Analysis

This section presents our experimental set-up, results, and analysis. The data sets under study are described in Subsection 8.1. The experimental setup is provided in Subsection 8.2. In Subsection 8.3, the results are analyzed and compared.

8.1. Data Sets

In the experiments, we consider nine data sets provided by the UCI machine learning repository [8] (Dua and Graff, 2017). The sets are summarized in Table 1. We note they are pre-processed: missing values are replaced by mean (mode) for numeric (discrete) features.

8.2. Experimental Settings

We experiment with three types of conformal set predictors: ICP, GS-ICP, and LS-ICP. All the three predictors are considered in a nearest-neighbor setting. More precisely,

- ICP employs the K-nearest-neighbor classifier as an underlying model (check Algorithms 1 and 2). The K-nearest-neighbor implementation is taken from the scikit-learn library (Pedregosa et al., 2011). The conformity function B is the general one: it is defined to output the probability estimate $\hat{p}(y_n|x_n)$ for any validation instance

8. $Colic_{og}$ refers to the original 3 class Horse Colic dataset

$(x_n, y_n) \in V$ equal to $\frac{1}{K}\sum_{k=1}^{K} \mathbb{1}[y_k = y_n]$ were y_k is the class label for the kth closest neighbor to the instance x_n.

- GS-ICP employs the ESVNN algorithm for computing Shapley values.
- LS-ICP employs the LESVNN algorithm for computing Shapley values.

We note that the general conformity function B employed in ICP is the core element of the characteristic functions employed by the ESVNN and LESVNN algorithms. Thus, due to the opposite manner of treating the training and validation datasets in ICP and S-ICP, the settings of ICP, GS-ICP, and LS-ICP ensure a fair comparison.

The set predictors ICP, GS-ICP, and LS-ICP are tested for three random data splits:

- $33\% - 67\%$: training data set T is 33% of the data and validation data set V is 67% of the data;
- $50\% - 50\%$: training data set T is 50% of the data and validation data set V is 50% of the data;
- $67\% - 33\%$: training data set T is 67% of the data and validation data set V is 33% of the data;

The set predictors are tested using a stratified 5-fold cross validation procedure. We employ several metrics to estimate the performance of the models. To test experimentally the validity of a conformal set predictor we use the error rate e. The error rate e for a significance level ϵ is defined as proportion of test instances whose predicted prediction-sets Γ^{ϵ} do not contain the correct class. To show experimentally that a conformal set predictor is valid, we need to show that for any significance level $\epsilon \in [0, 1]$ we have $e \leq \epsilon$.

To test experimentally the informational efficiency of a set predictor for a given significance level ϵ we employ three main metrics: the rate r^e of empty prediction sets, the rate r^s of single prediction sets, and the rate r^m of multiple prediction sets. The empty prediction sets, single prediction sets, and multiple prediction sets can be characterized by their own errors. The rate r^e of empty prediction sets is essentially an error, since the correct classes are not in the prediction-sets. The error rate e^s on single prediction sets is defined as the proportion of the single prediction sets that do not contain the correct classes. Analogously, the error rate e^m on multiple prediction sets is defined as the proportion of the multiple prediction sets that do not contain the correct classes. The error rates r^e, e^s, and e^m are related to the error rate e. More precisely, it is easy to show that $e = r^e + e^s r^s + e^m r^m$. In addition, we use the average size N of the prediction-sets Γ^{ϵ} (N criterion) and average p-value sum $S = \sum_{y \in Y} p_y$ (S criterion) introduced in (Fedorova et al., 2013) and (Johansson et al., 2013).

8.3. Results

8.3.1. Experiments with the Ionosphere Data

We start presenting the experimental results for ICP, GS-ICP, and LS-ICP on a particular data set Ionosphere. The Ionosphere data is chosen because it is a relatively difficult data in a 34 dimensional space with non-linearly separable classes. The nearest-neighbor parameter

K is set to 5, the locality parameter L of LS-ICP is set to 50% of the training set size, and the random data split is $33\% - 67\%$ (i.e. T is 33% and V is 67% of the data).

Figure 3(a) presents the error rate e of the three predictors in functions of the significance level $\epsilon \in [0, 1]$. It shows that the predictors are valid. However, ICP is conservatively valid while GS-ICP and LS-ICP are almost exactly valid. This is an interesting observation taking into account that all the three predictors are not smoothed (Vovk et al., 2005).

Figure 3(c) and Figure 3(e) present the rates r^s and errors e^s of single prediction sets in functions of the significance level $\epsilon \in [0, 1]$. They show that LS-ICP and GS-ICP have higher rate r^s of single prediction sets and lower rates r^e and r^m of empty and multiple prediction sets for significance level $\epsilon \leq 0.2$ when compared with ICP (see Figure 3(d)) [9]. Thus, LS-ICP and GS-ICP are more informationally efficient than ICP on the Ionopshere data. If we compare ICP and GS-ICP with each other, we note that ICP surpasses GS-ICP. The superiority of LS-ICP and ICP can be explained by the fact that both predictors are local and the Ionosphere data has two non-linearly separable classes.

We measure the time complexity of the predictors on a computer with 8Gb of memory and a dual-core Intel core i5 3.1 GHZ. The running times are 0.0048, 35.8782 and 26.385 seconds for ICP, GS-ICP, and LS-ICP, respectively, for one cross-validation fold. These results are in accordance with the time complexities of the predictors derived in the previous sections.

In Appendix A we present the error and prediction-set size plots of ICP, GS-ICP, and LS-ICP for the remaining 8 data sets from Table 1 [10]. The behavior of the predictors on these sets are similar to that presented in this section.

8.3.2. Experiments with All the Data

In Tables 2 and 3 we present the experimental results for ICP, GS-ICP, and LS-ICP on all the data sets from Table 1 for random splits $33\% - 67\%$, $50\% - 50\%$, and $67\% - 33\%$. The parameter settings of the predictors are those for which their performance is maximized. From the tables we observe that for significance level $\epsilon \in \{0.01, 0.05, 0.1\}$:

- the error rate e is smaller than or equal to ϵ for ICP, GS-ICP, and LS-ICP up to some statistical fluctuations;
- the rates r^s of single prediction sets for GS-ICP and LS-ICP are usually higher than those of ICP;
- the rates r^m of multiple prediction sets for GS-ICP and LS-ICP are usually lower than those of ICP;
- the rates r^e of empty prediction sets for ICP are lower than those for GS-ICP and LS-ICP;
- the rates r^s, r^m, and r^e are better for LS-ICP than for LS-ICP when the data set is more clustered;

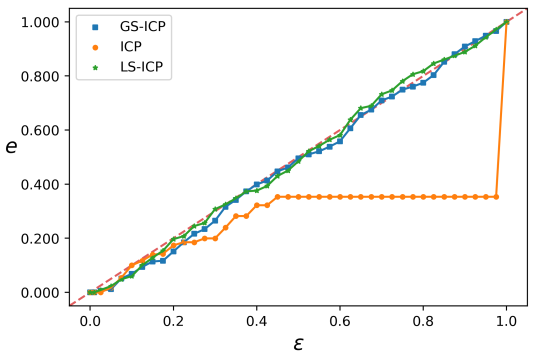
9. We note that the error rate e^m of multiple prediction sets is 0.0 in Figure 3(f) due to the fact that the Ionopshere data has two classes.
10. The appendix and code implementation can be found at https://github.com/w1ll1a9m/ShapleyConfomalPrediction

Set	ϵ	**V %**	**ICP**							**GS-ICP**							**LS-ICP**						
			e	r^e	r^m	r^s	e^s	N	S	e	r^e	r^m	r^s	e^s	N	S	e	r^e	r^m	r^s	e^s	N	S
Ionosphere	0.01	33	0	0	1	0	0	2	0.913	0.006	0	0.561	0.439	0.013	1.561	0.589	0.006	0.003	0.632	0.365	0.006	1.63	0.537
		50	0	0	1	0	0	2	0.907	0.009	0	0.561	0.439	0.019	1.561	0.586	0.003	0	0.61	0.39	0.003	1.61	0.548
		66	0	0	1	0	0	2	0.863	0	0	1	0	0	2	0.606	0	0	1	0	0	2	0.552
	0.05	33	0	0	1	0	0	2	0.913	0.043	0.003	0.254	0.744	0.054	1.251	0.589	0.054	0.009	0.103	0.889	0.054	1.094	0.537
		50	0.006	0	0.849	0.151	0.038	1.849	0.907	0.057	0	0.242	0.758	0.075	1.242	0.586	0.048	0	0.105	0.895	0.048	1.105	0.548
		66	0	0	1	0	0	2	0.863	0.094	0	0.191	0.809	0.116	1.191	0.606	0.04	0	0.103	0.897	0.04	1.103	0.552
	0.1	33	0.066	0	0.165	0.835	0.078	1.165	0.913	0.094	0.006	0.165	0.829	0.107	1.16	0.589	0.097	0.037	0.028	0.934	0.097	0.991	0.537
		50	0.08	0	0.162	0.838	0.095	1.162	0.907	0.097	0.003	0.154	0.843	0.111	1.151	0.586	0.091	0.026	0.017	0.957	0.091	0.991	0.548
		66	0.071	0	0.276	0.724	0.098	1.276	0.863	0.094	0	0.191	0.809	0.116	1.191	0.606	0.088	0.028	0.02	0.952	0.088	0.991	0.552
Colic	0.01	33	0	0	1	0	0	2	0.93	0.003	0	0.866	0.134	0.025	1.866	0.736	0.007	0	0.903	0.097	0.069	1.903	0.719
		50	0	0	1	0	0	2	0.89	0.009	0	0.561	0.439	0.019	1.561	0.702	0.003	0	0.87	0.13	0.026	1.87	0.683
		66	0	0	1	0	0	2	0.902	0	0	1	0	0	2	0.695	0.003	0	0	0.13	0.026	1.87	0.653
	0.05	33	0.013	0	0.853	0.147	0.091	1.853	0.93	0.057	0	0.629	0.371	0.153	1.629	0.736	0.057	0	0.629	0.371	0.153	1.629	0.719
		50	0.006	0	0.849	0.151	0.038	1.849	0.89	0.057	0	0.242	0.758	0.075	1.242	0.702	0.043	0	0.642	0.358	0.121	1.642	0.683
		66	0.017	0	0.846	0.154	0.109	1.846	0.902	0.07	0	0.495	0.505	0.139	1.495	0.695	0.043	0	0	0.358	0.121	1.642	0.653
	0.1	33	0.033	0	0.656	0.344	0.097	1.656	0.93	0.08	0	0.495	0.505	0.159	1.495	0.736	0.08	0	0.495	0.505	0.159	1.495	0.719
		50	0.08	0	0.162	0.838	0.095	1.162	0.89	0.097	0.003	0.154	0.843	0.111	1.151	0.702	0.087	0	0.472	0.528	0.165	1.472	0.683
		66	0.067	0	0.582	0.418	0.16	1.582	0.902	0.07	0	0.495	0.505	0.139	1.495	0.695	0.087	0	0	0.528	0.165	1.472	0.653
Heartc	0.01	33	0	0	1	0	0	2	0.84	0.003	0	0.904	0.096	0.034	1.904	0.643	0.007	0	0.822	0.178	0.037	1.822	0.627
		50	0	0	1	0	0	2	0.859	0.003	0	0.95	0.05	0.067	1.95	0.625	0.007	0	0.868	0.132	0.05	1.868	0.612
		66	0	0	0	0	0	2	0.829	0	0	1	0	0	2	0.639	0	0	1	0	0	2	0.661
	0.05	33	0.003	0	0.904	0.096	0.034	1.904	0.84	0.04	0	0.446	0.554	0.071	1.446	0.643	0.046	0	0.429	0.571	0.081	1.429	0.627
		50	0.043	0	0.607	0.393	0.109	1.607	0.859	0.053	0	0.469	0.531	0.099	1.469	0.625	0.04	0	0.442	0.558	0.071	1.442	0.612
		66	0.036	0	0	0.323	0.112	1.677	0.829	0.079	0	0.261	0.739	0.107	1.261	0.639	0.033	0	0.578	0.422	0.078	1.578	0.661
	0.1	33	0.04	0	0.528	0.472	0.084	1.528	0.84	0.079	0	0.261	0.739	0.107	1.261	0.643	0.083	0	0.231	0.769	0.107	1.231	0.627
		50	0.046	0	0.531	0.469	0.099	1.531	0.859	0.092	0	0.231	0.769	0.12	1.231	0.625	0.099	0	0.241	0.759	0.13	1.241	0.612
		66	0.04	0	0	0.396	0.1	1.604	0.829	0.079	0	0.261	0.739	0.107	1.261	0.639	0.079	0	0.261	0.739	0.107	1.261	0.661
Hepatitis	0.01	33	0	0	1	0	0	2	0.873	0	0	1	0	0	2	0.639	0	0	1	0	0	2	0.617
		50	0	0	1	0	0	2	0.907	0	0	1	0	0	2	0.63	0	0	1	0	0	2	0.592
		66	0	0	1	0	0	2	0.88	0	0	1	0	0	2	0.669	0	0	1	0	0	2	0.613
	0.05	33	0	0	1	0	0	2	0.873	0.021	0	0.451	0.549	0.038	1.451	0.639	0.021	0	0.507	0.493	0.043	1.507	0.617
		50	0	0	1	0	0	2	0.907	0.049	0	0.408	0.592	0.083	1.408	0.63	0.042	0	0.345	0.655	0.065	1.345	0.592
		66	0	0	1	0	0	2	0.88	0.092	0	0.155	0.845	0.108	1.155	0.669	0.021	0	0.563	0.437	0.048	1.563	0.613
	0.1	33	0.042	0	0.535	0.465	0.091	1.535	0.873	0.12	0	0.176	0.824	0.145	1.176	0.639	0.141	0.007	0.106	0.887	0.151	1.099	0.617
		50	0.021	0	0.746	0.254	0.083	1.746	0.907	0.106	0	0.197	0.803	0.132	1.197	0.63	0.085	0	0.197	0.803	0.105	1.197	0.592
		66	0.056	0	0.275	0.725	0.078	1.275	0.88	0.092	0	0.155	0.845	0.108	1.155	0.669	0.099	0	0.141	0.859	0.115	1.141	0.613
Sonar	0.01	33	0	0	1	0	0	2	0.903	0.01	0	0.822	0.178	0.054	1.822	0.787	0.019	0.005	0.827	0.168	0.086	1.822	0.792
		50	0	0	1	0	0	2	0.893	0	0	1	0	0	2	0.778	0	0	1	0	0	2	0.774
		66	0	0	0.952	0.048	0	1.952	0.936	0	0	1	0	0	2	0.743	0	0	1	0	0	2	0.739
	0.05	33	0.01	0	0.841	0.159	0.061	1.841	0.903	0.043	0.005	0.654	0.341	0.113	1.649	0.787	0.043	0.005	0.702	0.293	0.131	1.697	0.792
		50	0.019	0	0.808	0.192	0.1	1.808	0.893	0.038	0	0.663	0.337	0.114	1.663	0.778	0.029	0.005	0.707	0.288	0.083	1.702	0.774
		66	0.019	0	0.856	0.144	0.133	1.856	0.936	0.072	0	0.558	0.442	0.163	1.558	0.743	0.043	0	0.688	0.313	0.138	1.688	0.739
	0.1	33	0.048	0	0.644	0.356	0.135	1.644	0.903	0.087	0.01	0.534	0.457	0.168	1.524	0.787	0.115	0.005	0.514	0.481	0.23	1.51	0.792
		50	0.048	0	0.716	0.284	0.169	1.716	0.893	0.106	0.005	0.495	0.5	0.202	1.49	0.778	0.106	0.005	0.514	0.481	0.21	1.51	0.774
		66	0.029	0	0.779	0.221	0.13	1.779	0.936	0.072	0	0.558	0.442	0.163	1.558	0.743	0.106	0	0.438	0.563	0.188	1.438	0.739

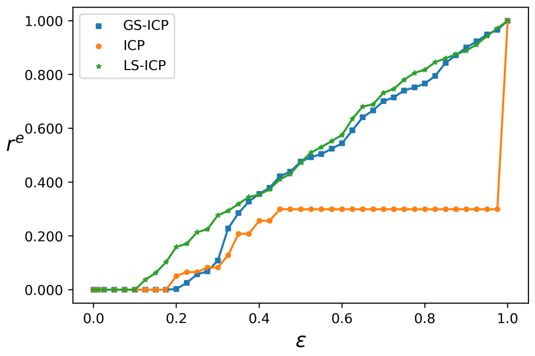
Table 2: Results on binary classification tasks. V% is the percentage of the validation set in the random data split.

Set	ϵ	V %	ICP								GS-ICP								LS-ICP							
			e	r^e	r^m	r^s	e^s	e^m	N	S	e	r^e	r^m	r^s	e^s	e^m	N	S	e	r^e	r^m	r^s	e^s	e^m	N	S
Wine	0.01	33	0.021	0	0.746	0.254	0.083	1.746	0.907	0.106	0	0.197	0.803	0.132	1.197	0.63	0.085	0	0.197	0.803	0.105	1.197	0.592	0	0	0
		50	0.056	0	0.275	0.725	0.078	1.275	0.88	0.092	0	0.155	0.845	0.108	1.155	0.669	0.099	0	0.141	0.859	0.115	1.141	0.613	0	0	0
		66	0	0	1	0	0	2	0.903	0.01	0	0.822	0.178	0.054	1.822	0.787	0.019	0.005	0.827	0.168	0.086	1.822	0.792	0	0	0
	0.05	33	0	0	1	0	0	2	0.893	0	0	1	0	0	2	0.778	0	0	1	0	0	2	0.774	0	0	0
		50	0	0	0.952	0.048	0	1.952	0.936	0	0	1	0	0	2	0.743	0	0	1	0	0	2	0.739	0	0	0
		66	0.01	0	0.841	0.159	0.061	1.841	0.903	0.043	0.005	0.654	0.341	0.113	1.649	0.787	0.043	0.005	0.702	0.293	0.131	1.697	0.792	0	0	0
	0.1	33	0.019	0	0.808	0.192	0.1	1.808	0.893	0.038	0	0.663	0.337	0.114	1.663	0.778	0.029	0.005	0.707	0.288	0.083	1.702	0.774	0	0	0
		50	0.019	0	0.856	0.144	0.133	1.856	0.936	0.072	0	0.558	0.442	0.163	1.558	0.743	0.043	0	0.688	0.313	0.138	1.688	0.739	0	0	0
		66	0.048	0	0.644	0.356	0.135	1.644	0.903	0.087	0.01	0.534	0.457	0.168	1.524	0.787	0.115	0.005	0.514	0.481	0.23	1.51	0.792	0	0	0
Glass	0.01	33	0	0	1	0	0	0	6	1.644	0.014	0	0.949	0.051	0	0.015	5.154	1.06	0	0	1	0	0	0	6	1.065
		50	0	0	1	0	0	0	6	1.578	0	0	1	0	0	0	6	1.076	0	0	1	0	0	0	6	1.014
		66	0	0	1	0	0	0	6	1.673	0	0	1	0	0	0	6	1.077	0	0	1	0	0	0	6	1.008
	0.05	33	0	0	1	0	0	0	6	1.644	0.056	0	0.846	0.154	0.061	0.055	3.678	1.06	0.037	0.019	0.841	0.14	0	0.022	3.813	1.065
		50	0	0	1	0	0	0	6	1.578	0.056	0	0.888	0.112	0.083	0.053	3.85	1.076	0.056	0.009	0.762	0.229	0.061	0.043	3.107	1.014
		66	0	0	1	0	0	0	6	1.673	0.103	0	0.743	0.257	0.109	0.101	2.215	1.077	0.047	0	0.827	0.173	0.108	0.034	3.28	1.008
	0.1	33	0	0	1	0	0	0	6	1.644	0.098	0	0.668	0.332	0.127	0.084	2.589	1.06	0.103	0.019	0.706	0.276	0.085	0.086	2.481	1.065
		50	0	0	1	0	0	0	6	1.578	0.112	0	0.659	0.341	0.096	0.121	2.322	1.076	0.112	0.009	0.664	0.327	0.114	0.099	2.248	1.014
		66	0	0	1	0	0	0	6	1.673	0.103	0	0.743	0.257	0.109	0.101	2.215	1.077	0.107	0	0.738	0.262	0.107	0.108	2.065	1.008
$Colic_{og}$	0.01	33	0	0	1	0	0	0	3	1.123	0.01	0	0.9	0.1	0.033	0.007	2.736	0.858	0.01	0	0.843	0.157	0.01	0	2.666	0.815
		50	0	0	1	0	0	0	3	1.086	0.013	0	0.88	0.12	0.056	0.008	2.669	0.859	0.013	0	1	0.154	0.013	0.008	2.656	0.788
		66	0	0	1	0	3	0	0	1.095	0	0	1	0	3	0	0	0.867	0	0	1	0	3	0	0	0.757
	0.05	33	0	0	1	0	0	0	3	1.123	0.054	0	0.672	0.328	0.133	0.015	2.171	0.858	0.04	0	0.659	0.341	0.04	0.02	2.094	0.815
		50	0	0	1	0	0	0	3	1.086	0.057	0	0.669	0.331	0.141	0.015	2.174	0.859	0.03	0	1	0.318	0.03	0.02	2.181	0.788
		66	0	0	1	0	3	0	0	1.095	0.104	0	0.545	0.455	1.833	0.169	0.169	0.867	0.047	0	0.639	0.361	2.043	0.047	0.047	0.757
	0.1	33	0.047	0	0.816	0.184	0.036	0.049	2.378	1.123	0.104	0	0.552	0.448	0.172	0.048	1.839	0.858	0.124	0	0.525	0.475	0.124	0.07	1.742	0.815
		50	0.064	0	0.759	0.241	0.111	0.048	2.164	1.086	0.097	0	0.575	0.425	0.15	0.058	1.863	0.859	0.094	0	0.816	0.468	0.094	0.057	1.779	0.788
		66	0.09	0	0.776	0.224	2.184	0.075	0.075	1.095	0.104	0	0.545	0.455	1.833	0.169	0.169	0.867	0.12	0	0.502	0.498	1.682	0.12	0.12	0.757
Iris	0.01	33	0	0	1	0	0	0	3	0.974	0	0	1	0	0	0	3	0.588	0	0	1	0	0	0	3	0.54
		50	0	0	1	0	0	0	3	0.955	0	0	1	0	0	0	3	0.542	0	0	1	0	0	0	3	0.533
		66	0	0	1	0	0	0	3	0.907	0	0	1	0	0	0	3	0.517	0	0	1	0	0	0	3	0.548
	0.05	33	0.02	0	0.027	0.973	0.021	0	1.027	0.974	0.053	0.007	0.08	0.913	0.051	0	1.093	0.588	0.033	0.007	0.013	0.98	0.027	0	1.007	0.54
		50	0.04	0	0	1	0.04	0	1	0.955	0.033	0.007	0.053	0.94	0.028	0	1.047	0.542	0.053	0.02	0.06	0.92	0.036	0	1.04	0.533
		66	0.027	0	0.127	0.873	0.031	0	1.127	0.907	0.107	0.08	0.007	0.913	0.029	0	0.927	0.517	0.053	0.02	0.04	0.94	0.035	0	1.02	0.548
	0.1	33	0.087	0.047	0	0.953	0.042	0	0.953	0.974	0.087	0.027	0.007	0.967	0.062	0	0.98	0.588	0.1	0.073	0.007	0.92	0.029	0	0.933	0.54
		50	0.073	0.033	0	0.967	0.041	0	0.967	0.955	0.093	0.04	0.007	0.953	0.056	0	0.967	0.542	0.1	0.08	0.007	0.913	0.022	0	0.927	0.533
		66	0.067	0	0	1	0.067	0	1	0.907	0.107	0.08	0.007	0.913	0.029	0	0.927	0.517	0.1	0.073	0	0.927	0.029	0	0.927	0.548

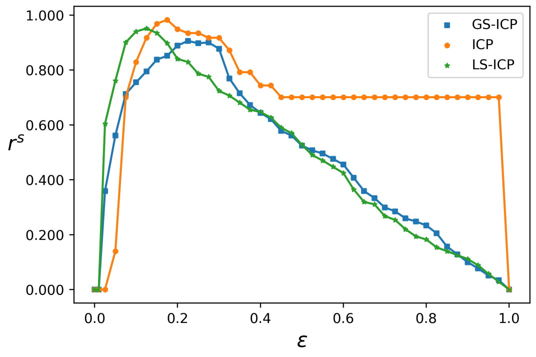
Table 3: Results on multi class classification tasks. V% is the percentage of the validation set in the random data split.



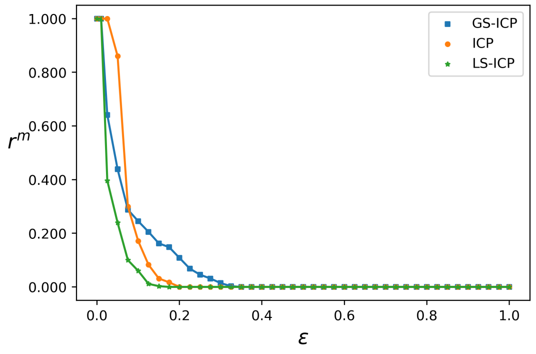
(a) Error rates



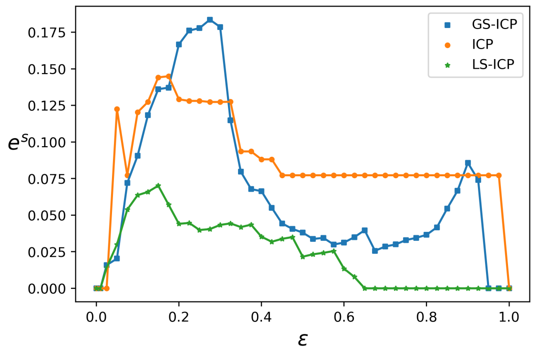
(b) Empty prediction-set rates



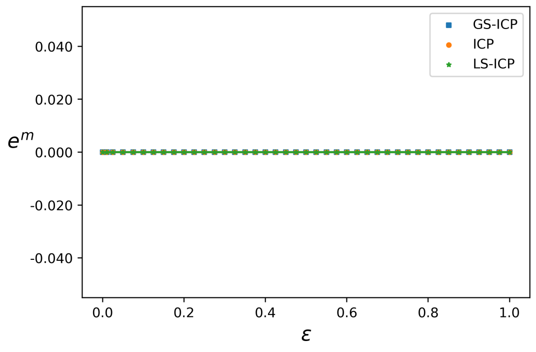
(c) Single prediction-set rates



(d) Multiple prediction-set rates



(e) Single prediction-set error rates



(f) Multiple prediction-set error rates

Figure 3: Error and Prediction-Set Size Plots for the Ionosphere Data

- the error rate e^s on single prediction sets and error rate e^m on multiple prediction sets ICP, GS-ICP, and LS-ICP do not show a pattern.

From the above we may conclude that for significance level $\epsilon \in \{0.01, 0.05, 0.1\}$ on the experimental data :

- ICP, GS-ICP, and LS-ICP are valid set predictors.
- GS-ICP and LS-ICP are more informationally efficient than ICP, and
- LS-ICP is more informationally efficient than GS-ICP for clustered data.

9. Conclusion

In this paper we proposed a new Shapley-value scheme to construct (non)conformity functions. The new scheme has two main advantages. First, it is a general scheme: it can be applied for any type of point predictors. Second, it allows computing conformity scores w.r.t. different aspects of the classification tasks/predictors.

Our Shapley-value scheme for conformity functions has to be used with care. First, it is not recommended to remove negative Shapley values when the p-values are being computed since it may result in invalid prediction sets. Second, our Shapley-value scheme is still computationally inefficient. This is due to the fact that all the available algorithms for Shapley-values are batch; i.e. they recompute Shapley values whenever an instance is added/removed.

Future research will focus on two research directions. The first one is developing incremental and kernel-based Shapley-value algorithms that will help speeding our Shapley-value scheme for conformity functions (Nalbantov and Smirnov, 2010). The second direction is combining standard conformal predictors and Shapley-value based conformal predictors, for example ICP and S-ICP predictors. As it is stated above these predictors use the validation and training instances in opposite manners: ICP computes conformity scores for the validation instances while S-ICP computes conformity scores for the training instances. Thus, if we combine the predictors we receive conformity scores for all the instances which can greatly improve the informational efficiency.

Acknowledgments

This work was done as part of a master's thesis research at the Department of Data Science and Knowledge Engineering (DKE) at Maastricht University. We are grateful to the COPA 2021 reviewers for their thoughtful comments and suggestions.

References

Dmitry Devetyarov and Ilia Nouretdinov. Prediction with confidence based on a random forest classifier. In Harris Papadopoulos, Andreas S. Andreou, and Max Bramer, editors, *Artificial Intelligence Applications and Innovations - 6th IFIP WG 12.5 International Conference, AIAI 2010, Larnaca, Cyprus, October 6-7, 2010. Proceedings*, volume 339 of

IFIP Advances in Information and Communication Technology, pages 37–44. Springer, 2010. URL https://doi.org/10.1007/978-3-642-16239-8_8.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Valentina Fedorova, Alex J. Gammerman, Ilia Nouretdinov, and Vladimir Vovk. Conformal prediction under hypergraphical models. In Andrew V. Jones and Nicholas Ng, editors, *2013 Imperial College Computing Student Workshop, ICCSW 2013, September 26/27, 2013, London, United Kingdom*, volume 35 of *OASICS*, pages 27–34. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2013. URL https://doi.org/10.4230/OASIcs.ICCSW.2013.27.

Amirata Ghorbani and James Y. Zou. Data shapley: Equitable valuation of data for machine learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251. PMLR, 2019. URL http://proceedings.mlr.press/v97/ghorbani19c.html.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gürel, Bo Li, Ce Zhang, Costas J. Spanos, and Dawn Song. Efficient task-specific data valuation for nearest neighbor algorithms. *Proc. VLDB Endow.*, 12(11):1610–1623, 2019a. URL http://www.vldb.org/pvldb/vol12/p1610-jia.pdf.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards efficient data valuation based on the shapley value. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 1167–1176. PMLR, 2019b. URL http://proceedings.mlr.press/v89/jia19a.html.

Ulf Johansson, Rikard König, Tuve Löfström, and Henrik Boström. Evolved decision trees as conformal predictors. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013*, pages 1794–1801. IEEE, 2013. URL https://doi.org/10.1109/CEC.2013.6557778.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4765–4774, 2017. URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.

Matthijs Moed and Evgueni N. Smirnov. Efficient adaboost region classification. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition, 6th International Conference, MLDM 2009, Leipzig, Germany, July 23-25, 2009. Proceedings*, volume 5632 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 2009. URL https://doi.org/10.1007/978-3-642-03070-3_10.

Georgi I. Nalbantov and Evgueni N. Smirnov. Soft nearest convex hull classifier. In *Proceedings of the 19th European Conference on Artificial Intelligence, ECAI 2010*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 841–846. IOS Press, 2010. URL https://doi.org/10.3233/978-1-60750-606-5-841.

Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alexander Gammerman. Inductive confidence machines for regression. In *Proceedings of 13th European Conference on Machine Learning, ECML 02*, volume 2430 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2002. URL https://doi.org/10.1007/3-540-36755-1_29.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Craig Saunders, Alexander Gammerman, and Volodya Vovk. Transduction with confidence and credibility. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 722–726. Morgan Kaufmann, 1999. URL http://ijcai.org/Proceedings/99-2/Papers/010.pdf.

Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9:371–421, 2008.

Lloyd Shapley. A value for n-person games. *Annals of Mathematical Studies*, 28:307–317, 1953.

Evgueni N. Smirnov, Georgi I. Nalbantov, and A. M. Kaptein. Meta-conformity approach to reliable classification. *Intell. Data Anal.*, 13(6):901–915, 2009. URL https://doi.org/10.3233/IDA-2009-0400.

Erik Strumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowl. Inf. Syst.*, 41(3):647–665, 2014. URL https://doi.org/10.1007/s10115-013-0679-x.

Stijn Vanderlooy, Ida Sprinkhuizen-Kuyper, and Evgueni Smirnov. An analysis of reliable classifiers through ROC isometrics. In *Proceedings of the ICML 2006 Workshop on ROC Analysis (ROCML 2006)*, pages 55–62, 2006.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

Shuang Zhou, Evgueni Nikolaevich Smirnov, and Ralf Peeters. Conformal region classification with instance-transfer boosting. *International Journal on Artificial Intelligence Tools*, 24(6):1560002:1–1560002:25, 2015. URL https://doi.org/10.1142/S0218213015600027.