Conformal Association Rule Mining (CARM): A novel technique for data error detection and probabilistic correction

Ilia Nouretdinov I.R.NOURETDINOV@RHUL.AC.UK
Centre for Reliable Machine Learning,
Royal Holloway University of London
James Gammerman JGAMMERMAN@GMAIL.COM
Centre for Reliable Machine Learning,
Royal Holloway University of London

Editor: Harris Papadopoulos, Khuong An Nguyen, Henrik Boström and Lars Carlsson

Abstract

Conformal prediction (CP) is a modern framework for reliable machine learning. It is most commonly used in the context of supervised learning, where in combination with an underlying algorithm it generates predicted labels for new, unlabelled examples and complements each of them with an individual measure of confidence. Conversely, association rule mining (ARM) is an unsupervised learning technique for discovering interesting relationships in large datasets in the form of rules. In this work, we integrate CP and ARM to develop a novel technique termed Conformal Association Rule Mining (CARM). The technique enables the identification of probable errors within a set of binary labels. Subsequently, these probable errors are analysed using another modern framework called Venn-ABERS prediction to correct the value in a probabilistic way.

Keywords: Conformal prediction, error correction, association rules.

1. Introduction

Conformal Prediction (CP) is a modern technique for reliable machine learning. It makes individual, confidence-based predictions that are *valid* under weak assumptions about the data distribution (such as i.i.d. or exchangeability). It is a broad framework that can be integrated with a standard prediction algorithm through its core component called the *non-conformity measure*, which transforms the output of the algorithm into a more reliable form.

This study focuses on applying CP to the problem of data correction, which involves identifying instances where the data contains errors and eliminating them where possible. In the context of machine learning, this can be seen as a unique task in which an algorithm is trained on the same data to which it is applied. It therefore does not fall neatly into the domain of supervised, semi-supervised or unsupervised learning. In this application, the validity property of CP provides bounds on the probability of a "false alarm" - an unnecessary correction suggestion. The upper limit for this probability is related to the *significance level*, which is a pre-selected parameter at the input stage.

For this task, we refine the concept of the non-conformity measure by linking it to a set of *association rules* which can be derived from the data. Using association rules as the underlying algorithm enhances the error detection process's transparency and explainability. By integrating CP and association rules we develop a novel technique called *Conformal Association Rule Mining (CARM)*.

To demonstrate the technique's effectiveness we apply it to three real data sets. We impute errors into binary labels to evaluate how accurately the erroneous examples are identified and explained. Then, to achieve error correction, we apply another technique from the domain of reliable machine learning called the *Venn-ABERS machine* for probabilistic prediction.

2. Background and Related Work

2.1. Data error detection and correction

In this work, we adopt the conventional understanding of a dataset for machine learning, namely that it is represented as a sequence of *feature vectors* x_i accompanied by *labels* y_i. Our use case is data error detection and correction, which may involve correcting feature values or label values. Here we focus on the labels, with feature correction left for future work.

Recent label correction approaches include instance-dependent label-noise learning (Berthon et al., 2021); combining principal component analysis (PCA), independent component analysis (ICA) and autoencoders (Salekshahrezaee et al., 2021), and confident learning based on pruning noisy data principles (Northcutt et al., 2021). A disadvantage of these approaches is their reliance on data generation models, which may or may not be accurate for a new dataset with unknown properties, while reliable machine learning techniques are valid under weak (exchangeability) assumptions. Some other methods are even more data-specific, having been developed for specific data types like text (Wang et al., 2022), (Chong et al., 2022) or images (Rottmann et al., 2022).

There are also approaches for special settings where the label correction task is modified, including partial label learning (Yan et al., 2020) studying label relationships (Cui et al., 2020); weak (noisy) supervision (Zheng et al., 2021); active label correction (Kremer et al., 2018); and label correction in federated learning (Zeng et al., 2022). These methods are designed for cases when datasets contain non-standard elements in their structure. The task of label correction should be differentiated from the closely related topic of making machine learning methods *robust* (tolerable) to label noise (Chen et al., 2022), without the aim of correcting and explaining of errors.

2.2. Association rules

Many machine learning models face criticism for their lack of transparency (Zhou and Kantarcioglu, 2020), which poses a significant obstacle to explaining and correcting errors, even when they have been detected. To address this issue, we examine the technique of association rule mining (ARM), which is a rule-based ML method for discovering interesting relationships between variables in datasets, first introduced in (Agrawal et al., 1993).

A common application for ARM is to identify patterns between items in transaction databases collected by supermarkets. For example, the rule $\{\text{bread}\} \rightarrow \{\text{butter}\}$ indicates that when customers buy bread, they are also likely to buy butter. Rules consist of two distinct sets of items, known as *itemsets*, A and B, where A is referred to as the *antecedent* and B is the *consequent*. In the above example, $\{\text{bread}\}$ would be the antecedent and $\{\text{butter}\}$ the consequent. Transposing these concepts to a typical ML dataset, an example of a simple association rule would be: "If feature F has a value of f, then feature G has a value of g", or more formally:

$$\text{IF } F = f \text{ THEN } G = g$$

where $F = f$ is the antecedent and $G = g$ is the consequent. Note that F or G could alternatively refer to a label rather than a feature, which is what we examine in this work. More complex rules involving multiple features are also possible. Armed with a set of these rules, there is an implication that errors can be identified as deviations from rules that hold true for the majority of the data.

Two fundamental concepts in ARM are the *support* and *confidence*. Support refers to the proportion of examples in a dataset where the rule holds, while confidence[1] is the conditional probability of the rule's consequent given its antecedent. For example, if the rule "IF $F = f$ THEN $G = g$" has a support of 70% and a confidence of 90%, this means that the rule holds true for 70% of the examples, and that when feature F has a value of f, feature G has a value of g in 90% of the cases.

We can formalise these definitions as follows. For a given rule R of the form $F = f \rightarrow G = g$:

$$\text{Support}(R) = P(F = f \cap G = g)$$

$$\text{Confidence}(R) = P(G = g | F = g) = \frac{\text{Support}(R)}{\text{Support}(F = f)}$$

where Support$(F = f)$ is the probability $P(F = f)$. Given user-defined thresholds $\min_{supp}$, $\min_{conf} \in [0, 1]$, a rule R is selected as an association rule if:

1. Support(R) $\geq min_{supp}$, and
2. Confidence(R) $\geq min_{conf}$

The first condition requires that the rule should be common enough and the second requires that it should be strong enough.

This framework underpins the most popular ARM method, called the *apriori* algorithm, which generates rules in a bottom-up manner. We refer the reader to (Agrawal et al., 1993) for more details, but in short the algorithm starts by taking user-defined minimum support and confidence thresholds. It then generates candidate itemsets, starting with single items and then iteratively creating new itemsets by combining smaller itemsets together. It then discounts or 'prunes' those itemsets with inadequate support. Next, it forms association rules by splitting frequent itemsets into antecedent and consequent parts, and calculating a confidence for each rule, keeping only those rules which have a confidence above the minimum threshold.

Association rules have been utilised in previous studies as a means to draw conclusions about suspected errors in data, such as in (Marcus et al., 2001), (Joshi, 2014), (Bohmer et al., 2020). However, as noted in (Hämäläinen and Nykänen, 2009), this traditional approach has notable flaws. The primary problem is that it does not capture the idea of *statistical dependence* between items. This often leads to identified rules that are spurious (i.e. false positives, or type 1 errors) and missed rules that are statistically significant (i.e. false negatives, or type 2 errors). There is an inevitable trade-off involved in these types of errors brought about by the fact that the user has to pick thresholds for the minimum support and confidence: if the thresholds are set too low, the algorithm may generate spurious rules resulting in more type 1 errors. But if the thresholds are too high, then it may miss statistically significant rules that are below the thresholds, leading to type 2 errors.

1. This should not be confused with the conception of confidence used in CP literature, with a different meaning.

A commonly proposed solution to this problem first suggested in (Patetsky-Shapiro, 1991) is to measure the *lift* instead of the confidence of a rule, which is the ratio of the observed support for both $\mathbf{F}$ and $\mathbf{G}$ occurring together to the expected joint probability of $\mathbf{F}$ and $\mathbf{G}$ assuming they are independent events, where $\mathbf{F}$ denotes $F = f$ and $\mathbf{G}$ denotes $G = g$. More formally:

$$\text{Lift}(\mathbf{F} \to \mathbf{G}) = \frac{P(\mathbf{F} \cap \mathbf{G})}{P(\mathbf{F}) \times P(\mathbf{G})}$$

A lift value greater than 1 indicates that the occurrence of X and Y together is more likely than if they were independent. This approach yields more statistically robust results, but doesn't eliminate the type 1 and 2 errors. It doesn't directly account for the statistical significance of a rule, so it does not guarantee that these rules will pass a rigorous statistical test. Furthermore, even if lift is used in the apriori algorithm, the algorithm remains sensitive to the choice of support and confidence thresholds.

To rectify this lack of statistical rigor, we follow the approach used in (Hämäläinen and Nykänen, 2009) and subject any proposed rule to a binomial test, enabling us to estimate exactly the significance of the rule by the binomial distribution. In this context, each example in the dataset corresponds to an independent Bernoulli trial, whose outcome is either 1 (i.e. $F = f \to G = g$ occurs) or 0 (i.e. $F = f \to G = g$ does not occur). The null hypothesis is that there is no association between the antecedent and the consequent. A small-enough p-value for any given rule provides evidence in favour of the alternative hypothesis, namely that there is such an association. Hence we can intepret these p-values as a measure of the 'weight' of a given rule.

Once every possible rule has been evaluated and weighted according to a binomial test, we then turn to the problem of quantifying how deviant a given example is from the given set of rules. CP offers a mathematically rigorous framework for doing this, allowing the number of suspected errors to be adjusted according to a significance level.

2.3. Conformal prediction

Conformal prediction (CP) (Vovk et al., 2005) is a framework for producing reliable predictions without relying on complex probabilistic models. Instead of generating single point values, it produces predictions as sets of varying sizes. It also comes with theoretically proven guarantees of validity, with the only assumption being that the data is *i.i.d.* CP has also been applied to anomaly detection (Balasubramanian et al., 2014) and clustering (Cherubin, 2014), (Cherubin et al., 2015).

Within the standard ML context, CP is applied to a sequence of objects $z_1, \dots, z_l$ and the aim is to provide a prediction for a new example z_{l+1}. All examples belong to an object space Z. The framework can be employed for both supervised and unsupervised learning tasks. In supervised learning, each $z_i \in Z$ is a pair (x_i, y_i) consisting of a feature vector x_i and a label y_i, making Z the product of a vector space X and a label set Y. In unsupervised learning, which includes tasks such as clustering and anomaly detection, z_i typically refers only to the feature vector $x_i \in X$.

Conformal predictors can operate on top of another ML algorithm (referred to as the underlying algorithm). This can be any classification or regression algorithm - the only requirement is that a 'score' can be extracted from it, from which we can develop a *nonconformity measure (NCM)*. The NCM quantifies how different a new example (i.e. a feature vector - label relationship in the supervised learning setting, or just a feature vector in the case of unsupervised learning) is from a set of previous examples: in other words, how non-conforming it is. In this sense, the NCM serves as an indicator of how unusual or 'strange' a new example is given prior data, allowing us to exclude a label from the prediction set if it generates new examples with high NCM values.

Formally, an NCM is a function $A : Z^{(*)} \times Z \to \mathbb{R}$ where Z denotes the set of all possible examples, and $Z^{(*)}$ denotes the set of all bags of examples of Z. The function A tells us how different an example $z_i \in Z$ is from a bag (multi-set) $Z^{(*)}$ by assigning it a non-conformity score.

An example of a possible NCM can be given by considering simple underlying algorithm, such as *k-nearest neighbours*. In the supervised learning setting, the strangeness of an example with respect to a bag might be the proportion of its k nearest neighbours with differing labels. Intuitively, an example with one label surrounded by examples with a different one would be considered strange or anomalous. In the unsupervised learning setting, the NCM can be simplified to the average distance between the examples and their nearest k neighbours. This means that an example is strange/anomalous if it is located in a low-density area with high average distance to its neighbours.

More formally, let us consider a new example z_{l+1}. Our null hypothesis is that z_{l+1} was drawn *i.i.d.* from the same distribution as the observed examples $z_1, \ldots, z_l$. Adding this new example z_{l+1} to our dataset results in an extended bag $\{z_1, ..., z_l, z_{l+1}\}$. We can then use an NCM to compute the non-conformity score for each example:

$$\alpha_i = A(\{z_1, ..., z_{i-1}, z_{i+1}, ..., z_{l+1}\}, z_i) \tag{1}$$

for each $i = 1, ..., l + 1$. Note that the NCM is applied to all examples, not just the new one. The scores produced by the NCM for each example are not particularly informative on their own. However, by comparing α for the new example z_{l+1} against that of all other examples, we can quantify how unusual it is using the notion of a p-value:

$$p(z_{l+1}) = \frac{\#\{i = 1, ..., l+1 : \alpha_i \geq \alpha_{l+1}\}}{l+1} \tag{2}$$

In this work we require a modification of this formula which accounts for imbalanced labels in the data. This is known as *label-conditional* CP (or also Mondrian CP):

$$p(z_{l+1}) = \frac{\#\{i = 1, ..., l+1 : \alpha_i \geq \alpha_{l+1}, y_i = y_{l+1}\}}{\#\{i = 1, ..., l+1 : y_i = y_{l+1}\}} \tag{3}$$

This p-value tests the null hypothesis stated earlier, namely that z_{l+1} was drawn *i.i.d.* from the same distribution as the observed examples $z_1, \ldots, z_l$. A threshold or *significance level* ε is usually selected, and if the p-value of example z_{l+1} is higher than this threshold, then we fail to reject the null (i.i.d.) hypothesis.

All conformal predictors automatically come with the property of *validity*, which states that the probability of incorrectly rejecting the null hypothesis is at most ε. That is, if the whole data sequence $z_1, z_2, \ldots, z_l, z_{l+1}$ indeed meets the *i.i.d.* assumption, then the p-value of z_{l+1} will be below this threshold with probability ε, or in other words it will be above the threshold with probability $1 - \varepsilon$. For example, if the threshold is $\varepsilon = 0.05$, then there is a 95% probability that $p(z_{l+1})$ will be above this threshold, confirming the *i.i.d.* hypothesis.

In the context of error detection, ε becomes an upper bound on the probability of a false positive (i.e. a false alarm). Hence the validity property prevents us from identifying an error when there is no need for it: by setting ε at a sufficiently low level, we can limit the number of examples which fail to meet this threshold and hence are flagged as suspicious because their inclusion in the data indicated a violation of the i.i.d. hypothesis. It is important to note that statistical analysis cannot eliminate such errors, but does allow us to regulate their occurrence.

Having a large pool of features (or, in our case, association rules derived from them) expands the potential for defining strangeness, which has only been partially examined previously. Something similar to this can be found in (Nouretdinov et al., 2011) where the non-conformity of an example with respect to the set was estimated based on the consistency (regularity) of the remainder of the set. In that work, the consistency was measured by the proportion of features that displayed very significant separability between the two classes. The accuracy of this method was driven by the large number of features making the strangeness metric sensitive enough. This motivated us to make a similar construction based on a pool of association rules that will be described later in this paper.

2.4. Probabilistic prediction

When a data example is identified as an error, it is important to conduct a thorough investigation that includes not only the probability of the error, but also a suggested correction for it. For this purpose, we can employ the Venn-ABERS framework, first described in (Vovk and Petej, 2014). This is a subset of the *Venn prediction* framework which is specifically designed for reliable probabilistic prediction and, similar to CP, can serve as an enveloping framework for an underlying algorithm. Notably, the Venn-ABERS framework includes a component called a *scoring function*, which plays a role analagous to the NCM in the CP framework.

An approach for integrating the Venn prediction framework with association rules has previously been presented in (Alkhatib et al., 2022), although this work was concerned with quantifying the uncertainty of explanations generated by ARM (and comparing it with other methods in explainable ML such as Anchors), rather than detecting and correcting label errors.

2.4.1. VENN PREDICTOR

Assume we are given training samples $(z_1, \ldots, z_n)$ where $z_i = (x_i, y_i)$, $x_i \in R^d$, and $y_i \in 0, 1$ for $i = 1 \ldots n$. Given a test object x_{n+1}, a Venn Predictor provides a probabilistic prediction in the form of a probability distribution over the possible label values. To achieve this, we introduce the concept of a *Venn taxonomy*. A Venn taxonomy A is a measurable function that assigns an equivalence relation $\sim$ to each $n \in 2, 3, \ldots$ and each sequence $(z_1, \ldots, z_n)$. This relation must be equivariant, meaning that for each n and any permutation π of $1, \ldots, n$:

$$(i \sim j | z_1, \ldots, z_n) \Rightarrow (\pi(i) \sim \pi(j) | z_{\pi(1)}, \ldots, z_{\pi(n)}),$$

where $(i \sim j | z_1, \ldots, z_n)$ denotes that i is equivalent to j under the relation assigned by A to $(z_1, \ldots, z_n)$. We define the equivalence class of j as:

$$A(j | z_1, \ldots, z_n) := \{i \in \{1, \ldots, n\} | (i \sim j | z_1, \ldots, z_n)\}.$$

A Venn predictor with a Venn taxonomy A outputs the pair (p_0, p_1), where

$$p_y = \frac{|\{i \in A(n+1|z_1, \ldots, z_n, (x_{n+1}, y))|y_i = 1\}|}{|A(n+1|z_1, \ldots, z_n, (x_{n+1}, y))|} \tag{4}$$

Note that both p_0 and p_1 express the probability of the test object having 1 as its label. $y \in \{0, 1\}$ will be referred to here as the *version* of Venn prediction.

Venn predictors provide a calibration guarantee under the assumption that the observations are *i.i.d.* A general, but complex, game-theoretic form of Venn machine validity is provided in (Vovk et al., 2005). Following (Vovk and Petej, 2014) we can follow a simplified definition of calibration. For a single probability distribtuion P_a, calibration implies that:

$$\mathbb{P}\left[Y = a \mid P_a\right] = P_a \qquad \text{a.s.} \tag{5}$$

This means that the probability of the label being a given the predicted probability of P_a is indeed P_a. In practice, this corresponds to the relative frequency of the label being a tending to P_a when computed only on the objects for which the probability of label a is P_a.

2.4.2. VENN-ABERS PREDICTOR

The Venn Prediction framework does not prescribe any one method for constructing the taxonomy. The Venn-ABERS prediction framework offers a simple way to establish a taxonomy with desirable properties using a scoring classifier as the underlying ML method. It requires only that the score $s(x_{n+1})$ output by the underlying algorithm is directly related to the probability of label y_{n+1} being 1. Simply put, the higher the score, the higher the probability of the label being positive.

The key distinction between an NCM and a scoring function is that the latter evaluates the likelihood of an example belonging to the positive class, rather than assessing its strangeness. A scoring function can be a probabilistic score from standard prediction algorithms, such as a neural networks or random forests. In this case, Venn-ABERS calibrates the predictions under the weak *i.i.d.* assumption. However, a scoring function can also use any ranking method, and in this study we use association rules for this purpose.

The taxonomy built by Venn-ABERS predictors (a partition of $\mathbb{R}$ into intervals) is designed to ensure that the predicted probabilities are non-decreasing (as a function of s) and maximise the likelihood of the training set. This is achieved by a calibrator $g(s)$ that maximises:

$$\prod_{i=1,2,\ldots,n} q_i$$

where $q_i = g(s(x_i))$ if $y_i = 1$ and $q_i = 1 - g(s(x_i))$ if $y_i = 0$.

Based on results in (Ayer et al., 1955), (Brunk, 1955), (Zadrozny and Elkan, 2002), the desired $g(s)$ can be obtained using isotonic regression (i.e. order-preserving regression) on (s_i, y_i), defined as the non-decreasing function that minimizes the sum of square residuals:

$$\sum_{i=1}^{n} (g(s(x_i)) - y_i)^2$$

Note that the isotonic regression is piecewise constant. The intervals of x where $(g(s(x)))$ is constant are the categories of the Venn taxonomy. The multi-probabilistic prediction for x is the pair

$$(p_0, p_1) = (g_0(s_0(x)), g_1(s_1(x)))$$

3. Proposed method

In this section we present a method for integrating the three frameworks (association rule mining, conformal prediction, and Venn-ABERS prediction) involved in this study.

3.1. Association Rule Non-Conformity Measure

In Algorithm 1, we first evaluate each possible association rule for its statistical significance using a binomial test, generating a p-value which can be interpreted as its weight as explained in Section 2.2. Next, the algorithm compares a given data example z to the set of rules and calculates a non-conformity score for it, which indicates the degree to which it deviates from these rules. A high score suggests the example may be an error, as it either violates multiple rules or just a few rules with large weights. It is important to note that the actual implementation of this algorithm extends its application to the entire dataset, however here we focus only on example z for clarity.

Algorithm 1 Association Rule Non-Conformity Measure using rule-exceptions

```
INPUT: example $z = (x_{l+1}, y_{l+1})$ where $x \in \{0,1\}^m$, $y \in \{0,1\}$
INPUT: bag of (same kind) examples $Z = \{(x_1, y_1), \ldots, (x_l, y_l)\}$
INPUT: pool (set) $P$ of association rules $r$ =“IF ... THEN $y = Y$”
$y_a = \frac{\sum_{i=1}^{l+1} y}{l+1}$
$\alpha := 0$
for $r \in P$ do
    support base $S := \{i = 1, \ldots, l+1 :$ (conditions of $r$) are true $\}$
    support base size $s := |S|$
    $p_r$ := p-value of the empirical distribution of $y_i$ within the support base, tested against the Bernoulli distribution with probability $y_a$;
    calculate average of $y_i$ over the support base: $y'_a := \frac{\sum_{i \in S} y_i}{s}$
    if $y'_a > y_a$ then
        $C := 1$
    else
        $C := 0$
    end if
    if $y_{l+1} \neq C$ then
        $\alpha := \alpha - \log p_r$
    end if
end for
OUTPUT: non-conformity score $\mathcal{A}(z, Z) = \alpha$
```

Let us examine the algorithm in more detail. Its inputs are:

1. A pool P of association rules r of the form "IF [antecedent] THEN $y = Y$. In our approach, the antecedent could refer to one feature on its own or a combination of two.

2. An example $z = (x_{l+1}, y_{l+1})$ where $x \in \{0,1\}^m$, $y \in \{0,1\}$,

3. A bag of similar data examples $Z = \{(x_1, y_1), \ldots, (x_l, y_l)\}$

Note that the numbering of the data examples in this algorithm may differ from their order in the dataset when the NCM is applied. Also, unlike in standard CP, z is not a test example with an unknown label as opposed to bag Z with known labels; all labels are known but under question as to whether they are potentially erroneous.

For each rule r we then identify a support base S which is the set of examples that satisfy the conditions of r, i.e. where the antecedent conditions are true. We then use a binomial test to calculate a p-value p_r, assessing the statistical significance of r with respect to the empirical distribution of y_i ($i \in S$) within the support base. The p-value compares this distribution to a reference distribution, namely the Bernoulli distribution with probability y_a, which represents the expected distribution of the label y_i under the null hypothesis of no association between the rule's antecedent and the label. The p-value provides a measure of the strength of the rule's association with y_i. A smaller p-value suggests that the empirical distribution of y_i within S is significantly different from the expected distribution under the null hypothesis. In other words it indicates a stronger rule with a larger "weight". Again, these p-values should not be confused with those generated by CP.

After determining the rule weights, we calculate the non-conformity score for example z. To do this, we compute a value y'_a which is the average label value y_i within the support base S. By comparing y'_a to y_a, we can determine whether the rule is more closely associated with label $Y = 1$ or $Y = 0$ within the support base S. If $y'_a > y_a$, we expect the label for z to be 1, and vice versa. If the expected label differs from the actual label, it means that example z is an exception to the rule. The non-conformity score α is then updated by adding the negative logarithm of p_r to its current value, which starts at zero[2]. Thus α accumulates the differences between the expected and actual outcomes, representing the degree of non-conformity for the example z.

3.2. Converting non-conformity scores to conformal p-values

Algorithm 2 outlines the process of converting non-conformity scores to p-values (as understood in the CP context) using label-conditional CP. The algorithm aims to identify if an example is a suspected error based on the calculated p-value and a pre-defined threshold.

The inputs for this algorithm are a data sequence containing N examples; a non-conformity measure (function) $\mathcal{A}$ that calculates the non-conformity scores for each example, which is defined in Algorithm 1; and a threshold ε which determines whether an example should be reported as an anomaly. The algorithm iterates through each example in the sequence, computing the non-conformity scores using the provided NCM. Subsequently, the p-value for the last example (x_N, y_N) is computed by dividing the count of examples with the same label and a non-conformity score greater than or equal to the current example's score by the total number of examples sharing the same label. If the calculated p-value is less than the threshold, the algorithm reports the example as a suspected error. Algorithms 1 and 2 form the crux of conformal association rule mining (CARM).

2. We transform the p_r value to a logarithmic scale in order to scale the deviations in a more interpretable way, penalise exceptions from stronger rules and improve numerical stability.

Algorithm 2 Label-conditional data processing step

```
INPUT: data sequence $Z' = ((x_1, y_1), \dots, (x_N, y_N))$
INPUT: non-conformity measure (function) $\mathcal{A}$
INPUT: threshold $\varepsilon$
for $i := 1, \dots, N$ do
    $\alpha_i := \mathcal{A}((x_i, y_i), Z' \setminus \{(x_i, y_i)\})$
end for
$p_N := \frac{|\{j : y_j = y_N, \alpha_j \geq \alpha_N\}|}{|\{j : y_j = y_N\}|}$
if $p_N < \varepsilon$ then
    OUTPUT: report $(x_N, y_N)$ as an anomaly (suspected error)
end if
```

3.3. Probabilistic prediction for correcting suspicious labels using Venn-ABERS prediction

Algorithm 3 demonstrates an approach for probabilistic prediction of labels after detecting a suspected error. As a reminder the Venn-ABERS framework, similar to CP, includes a core element known as the scoring function which connects it to the underlying ML algorithm. For a dataset with binary labels, CP can tell us that a given label looks suspicious, while Venn-ABERS can give us a probability score for the alternative label. While this may seem redundant ("if a label of 0 looks suspicious then surely it must be 1?"), there is always the possibiity that neither label looks plausible for a given data example. In this case, the p-value computed by CP would be small for both labels. We remind the reader that these p-values are not probabilities of the labels *per se*. Venn-ABERS allows us to compute them.

Algorithm 3 presents an example of a scoring function, with output scores denoted as $\Sigma_1, \dots, \Sigma_N$. This scoring function is analogous to the NCM used in Algorithm 1: Rule weights are assigned based on their statistical significance, and both 'positive' and 'negative' rules are handled separately to calculate their difference. The scoring function compares the amount of evidence (in the form of positive and negative rules) for the example being part of the positive class. Once the scores are calculated, the general methodology for the Venn-ABERS framework is applied as outlined in (Vovk and Petej, 2014). This process incorporates an isotonic approximation step to convert the labels into probabilistic values. The outcome consists of a pair of predicted probabilities $\hat{p}_0$ and $\hat{p}_1$, rather than a single value. This is a known limitation of Venn machines, and the true probability is one of these two predicted probabilities. When an appropriate scoring function is selected, these probabilities are typically close to one another.

4. Results and Discussion

In this section we start by analysing the results of CARM on the three datasets, and then focus only on the Mushrooms data to investigate a) the rules broken by the most suspicious examples and b) the results of probabilistic prediction, to show how our approach could be used in practice for error correction.

4.1. Datasets

For our experiments, we used the following public datasets:

Algorithm 3 Label-error correction by Venn-ABERS prediction

```
INPUT: data sequence $Z' = (x_1, y_1), \ldots, (x_N, y_N)$ with a suspected mistake in $(x_i, y_i)$
for $h := 0, 1$ do
  $\hat{y}_1 := y_1, \ldots, \hat{y}_n := y_n$
  $\hat{y}_i := h$
  $y_a := \frac{\sum_{i=1}^{N} \hat{y}_N}{N}$
  $\Sigma_1 := \Sigma_2 := \ldots \Sigma_N := 0$
  for $q := 1, ..., m$ do
    for $r := 1, ..., m$ do
      support base $S := \{n = 1, \ldots, N : x_n^q = x_n^r = 1\}$
      $p :=$ p-value of the empirical distribution of $y_k$ within the support base, tested against the
      Bernoulli distribution with probability $y_a$;
      if average $y_k$ within $k \in S$ is above $y_a$ then
        for $k \in S$ do
          $\Sigma_k := \Sigma_k - log(p)$
        end for
      else
        for $k \in S$ do
          $\Sigma_k := \Sigma_k + log(p)$
        end for
      end if
    end for
  end for
  Let $(\Sigma'_j, v'_j, w'_j)$ be triple vectors $(\Sigma_j, \hat{y}_j, j)$ sorted by the first dimension $\Sigma_j$.
  while $\exists j : v'_j > v'_{j+1}$ do
    take the smallest $j$ s.t. $v'_j > v'_{j+1}$
    let $j^-$ be the smallest $j^- \leq j$ s.t. $v'_{j^-} = \cdots = v'_{j-1} = v'_j$
    let $j^+$ be the largest $j^+ \geq j + 1$ s.t. $v'_{j+1} = v'_{j+2} = \cdots = v'_{j^+}$
    replace $v'_{j^-}, v'_{j^-+1}, \ldots, v'_{j^+-1}, v'_{j^+}$ with their average;
  end while
  $\hat{p}_h := v'_j$ s.t. $w'_j = i$
end for
OUTPUT: $(\hat{p}_0, \hat{p}_1)$
```

1. Mushrooms (UCI, 1987)
2. Wine Quality (Cortez, 2009)[3];
3. Adult Income (Becker, 1996).

We chose a binary feature from each dataset to serve as the label: edible/poisonous for Mushrooms; white/red for Wine; income greater/less than $\$50K$ for Adult Income. Then, we introduced known errors into the data by flipping the value to its opposite for 1% of the data examples.

3. Originally, the dataset was presented as two separate files for red wine and for white wine. For our purposes, we combined them and randomly shuffled the order.

The remaining features are converted to binary form by one-hot encoding. For categorical features, a new feature is created for each possible value. For example, if the values of a feature are a, b, c, this feature is replaced by three features according to the rule: $a \to (1, 0, 0)$; $b \to (0, 1, 0)$; $c \to (0, 0, 1)$. This transformation is applied even to binary features, e.g. $0 \to (1, 0)$, $1 \to (0, 1)$. This is necessary because, in the defined pool of association rules, all the conditions are positive (IF $F = f \dots$), so additional features make negative conditions (IF $F = 0 \dots$) possible.

The Mushroom dataset included only categorical features. In the Wine and Adult Income datasets, there are also numerical features. For them, we applied binning[4] as an extra pre-processing step.

Figure 1: CARM p-values (base-10 log scale) for Mushrooms. Imputed errors in red.

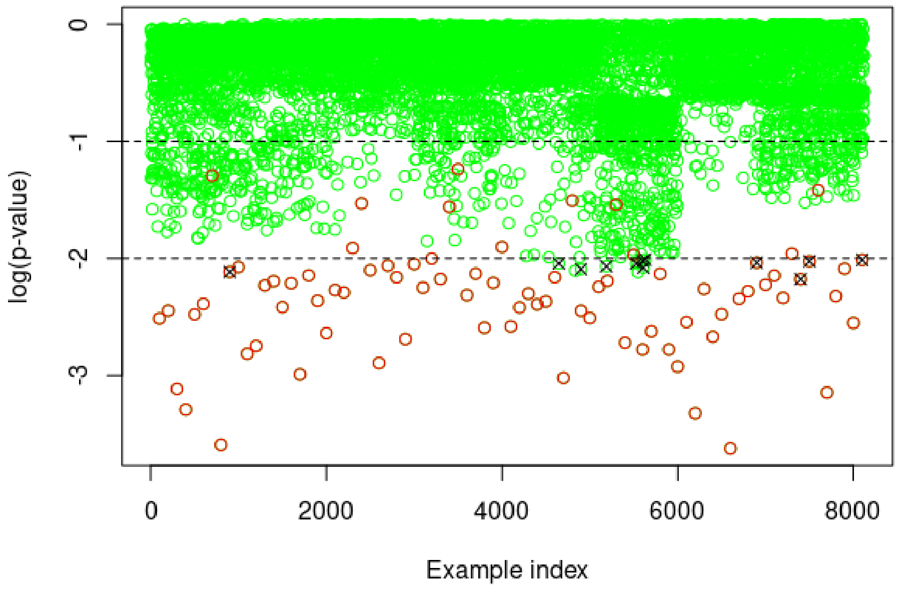
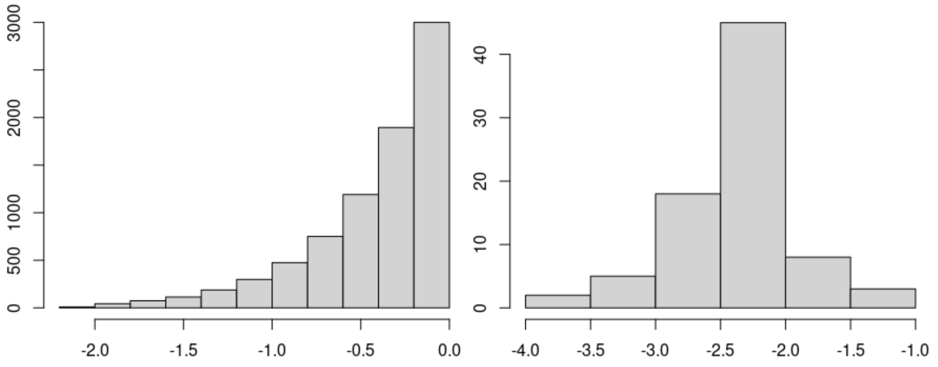


Figure 2: Histograms of p-values (log scale) for non-errors (LHS) and errors (RHS) in Mushrooms



4. This required setting a number of bins as a parameter. For the Wine dataset we used ten bins, while for the Adult Income dataset this was reduced to three as it yielded better performance.

4.2. CARM on Mushrooms data

Figure 1 is a plot of the CARM p-values for all data examples in a base-10 logarithmic scale. Examples with imputed errors are marked in red, and most of them (70 out of 81) are detected as anomalies at a p-value threshold of 1% (which is -2 in base-10 logarithmic scale). All other imputed errors have p-values below 10% (-1 in logarithmic scale).

For this dataset only, as well as running CARM on the original labels, we also ran it on the alternative labels for all suspicious examples (i.e. if an example with label 0 was flagged as suspicious then we computed a p-value for the same example with label 1). For the most part this yielded high p-values close to 1, indicating that the example is no longer suspicious. However, there were ten examples which had low p-values for both labels, and these are indicated with a black cross on the plot. We will analyse these examples further in Section 4.6.

Another way of visualising this result is shown in Figure 2, which shows histograms of the p-values for non-errors and errors, corresponding to the green and red points in Figure 1 respectively. We discuss these distributions in comparison with the other datasets in Section 4.5.

Figure 3: CARM p-values (base-10 log scale) for Wine. Imputed errors in red.

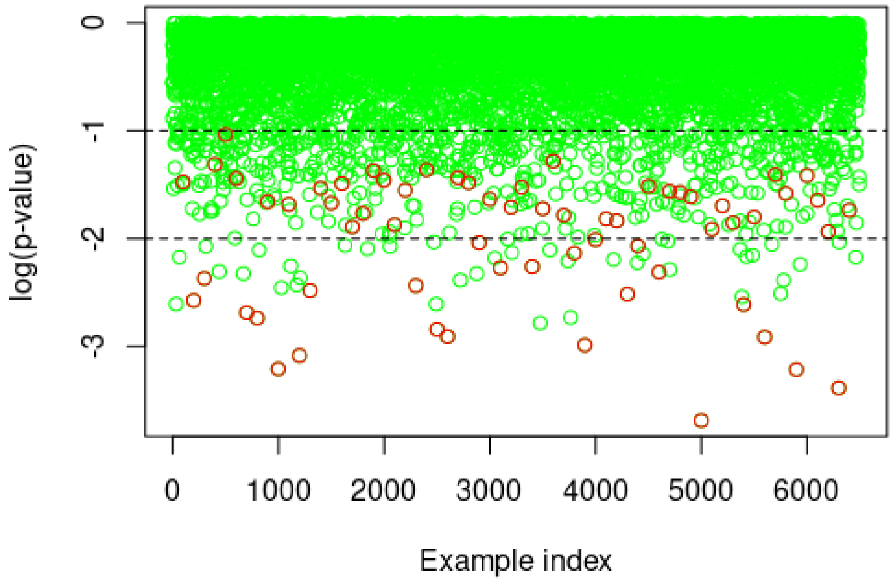
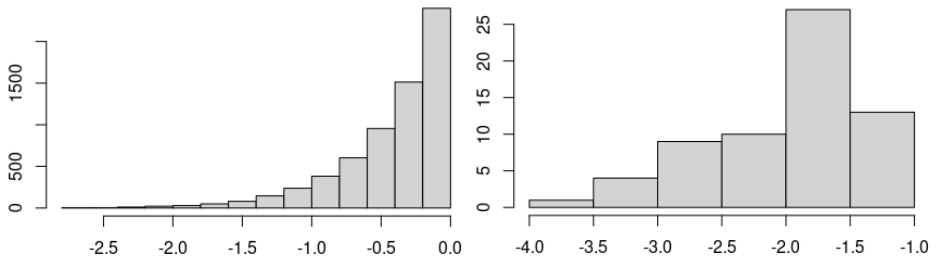


Figure 4: Histograms of p-values (log scale) for non-errors (LHS) and errors (RHS) in Wine



4.3. CARM on Wine data

Figure 3 displays a plot of the CARM p-values for the wine data examples in base-10 logarithmic scale, analagous to Figure 1. Since this dataset is more noisy than mushrooms, the distinction between errors and non-errors is not as clear cut. However, the distributions still differ considerably, as shown in the histograms in Figure 4.

4.4. CARM on Adult Income data

Figure 5 is analagous to Figs 1 and 3. This dataset is the noisiest of the three, and hence the distinction between errors and non-errors is the worst. However, the distributions for the two classes remain quite different, as shown in the histograms in Figure 6, and the majority of errors are still captured below the $p = 10\%$ level.

Figure 5: CARM p-values (base-10 log scale) for Adult. Imputed errors in red.

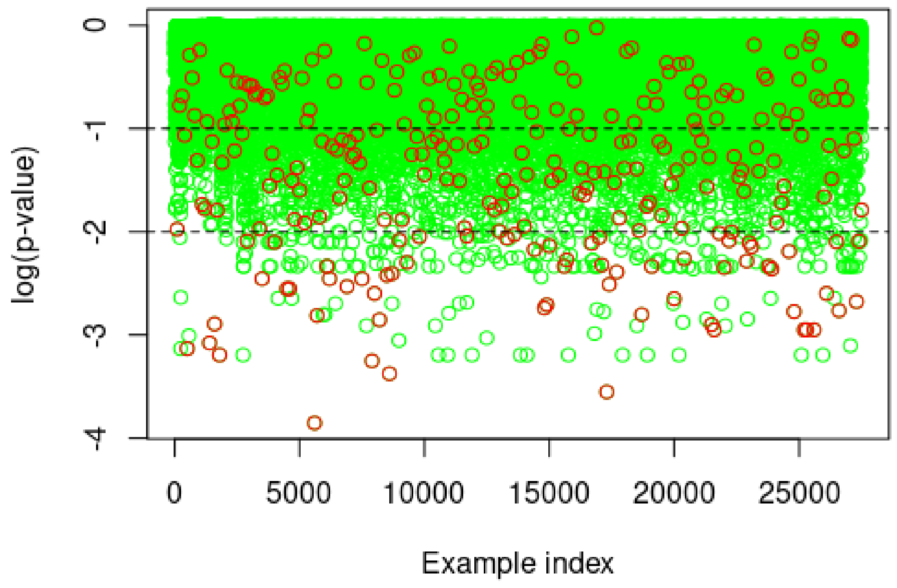
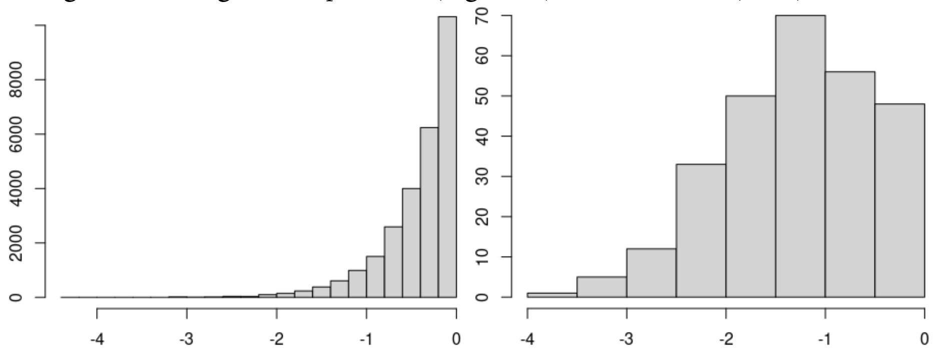


Figure 6: Histograms of p-values (log scale) for non-errors (LHS) and errors (RHS) in Adult



4.5. Discussion of CARM results on the datasets

The p-value histograms for non-errors in Figs 2, 4 and 6 (RHS) look very similar to each other. This observation is consistent with the validity property that comes automatically with CP, which imposes constraints on the number of false alarms as discussed in Section 2.3. This forces the p-values to follow an approximately uniform distribution[5] with an average value around 0.5.

The p-value histograms for the errors (LHS) differ somewhat as they reflect the algorithm's performance on the datasets. Generally speaking, the lower these p-values are, the better the CARM method performs in identifying errors. It is clear to see that the algorithm performs best on Mushrooms, followed by Wine, followed by Adult Income.

We can quantify the performance of CARM using two evaluation metrics usually employed in classification tasks: *precision* and *recall*. Precision measures how many of the predicted positive examples are actually positive. Recall (also called *sensitivity*) measures how many of the actual positive examples were predicted positive.

More formally:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

Hence optimising for precision minimises the number of false positives, while optimising for recall minimises the number of false negatives.

In our case, we can state that any example with a p-value below a given threshold has been identified as an error and is therefore a 'positive'. A 'true positive' is a correctly identified error (red dots in the p-value plots); a 'false positive' is a mistakenly identified error (green dots); and a 'false negative' is a true error that wasn't captured below the thresholds (red dots found above the threshold).

Tables 1 and 2 present these metrics for the datasets at three thresholds.

Table 1: Precision at three p-value thresholds

Dataset	$p < 0.1\%$	$p < 1\%$	$p < 10\%$
Mushrooms	100.0%	87.5%	10.0%
Wine	100.0%	37.5%	9.9%
Adult Income	29.0%	23.3%	6.2%

Table 2: Recall at three p-value thresholds

Dataset	$p < 0.1\%$	$p < 1\%$	$p < 10\%$
Mushrooms	8.6%	86.4%	100.0%
Wine	7.8%	37.5%	100.0%
Adult Income	2.5%	23.2%	62.1%

The metrics confirm the trend in performance on the three datasets. CARM performs very well on Mushrooms, with excellent precision at the lower p-value thresholds and recall at the highest

5. Recall that the p-values have been log-transformed.

threshold. It performs less well on Wine, but still manages 100% precision at the lowest threshold and 100% recall at the highest threshold. It performs worst on Adult Income, although its recall of 62% at $p < 10\%$ shows that it was still able to capture the majority of true errors, albeit with many false positives as shown by the low precision of 6%.

The relatively poor performance on Adult Income can probably be attributed to the intrinsic noise associated with this dataset as opposed to the other two. In the case of Mushrooms, the features denote physical characteristics of individual mushrooms, while for Wine, they comprise physico-chemical metrics. In contrast, the Adult Income dataset pertains to humans rather than natural entities, and the relationship between its features (e.g. education level, occupation, marital status) and the label is inherently less deterministic. Additionally, the label (income greater or less than $50K) fundamentally originates from a regression problem, which has been transformed into binary classification by applying an arbitrary threshold. Furthermore, in this study we have only considered a relatively small pool of simple association rules (with a maximum of two features per rule), leaving its extension to more complex rules for future work.

4.6. Mushrooms: Further analysis and probabilistic prediction

Having evaluated CARM, we now provide further analysis of its results and demonstrate the effectiveness of probabilistic prediction for correcting erroneous labels. For these purposes we use the Mushrooms dataset which yielded the best performance from CARM.

Table 3 shows an investigation report into the mushroom with the smallest p-value, i.e. the most suspicious one. Unsurprisingly, this is one of the examples for which we imputed an error in the label. The table shows its index in the data, its label, p-value, and the rules that it breaks, either involving only one feature or two. There is a confidence score next to each rule, which refers to the conditional probability of the rule's consequent given its antecedent as explained in Section 2.2. We include this to show why the example was flagged by the algorithm as very suspicious: it clearly breaks many strong rules. For example, there is an intuitively true rule highlighted in blue which shows that 99% of the times when a mushroom has an odor ($odor = y$) it is poisonous. Edible mushrooms tend not to smell!

The remainder of the table deals with investigating what the correct label should be. First, CARM is run again on the example, but this time having flipped the label to its alternative value (0 to 1 or vice versa). The output of this is denoted the *alternative p-value*. This yields 0.96, indicating that the alternative label makes the example non-suspicious. We then show the result of running Venn-ABERS probabilistic prediction, which outputs a probability of ~ 0 that the correct label is 1. This matches the true label (prior to error imputation). We can conclude from this report that we have successfully a) detected a label error and b) corrected it to a value that is probabilistically most likely.

Table 4 extends this report to more data examples, showing the top ten most suspicious examples detected by CARM, along with the observed label, p-values for the observed and alternative labels, predicted outputs from Venn-ABERS and the true label. In all cases the alternative p-values are close to 1, indicating that flipping the label makes the example no longer suspicious. Venn-ABERS predicts the true label with high confidence (here used in the CP sense), with predicted probabilities of $\sim 0.01\%$ for true labels of 0 and of $\sim 0.99\%$ for true labels of 1.

Table 3: Investigation report for data example flagged as most suspicious

Index in data	6600	
Observed label	1 (= edible)	
p-value	0.000237	
Broken rules	(listed if significant at the threshold 10^{-9})	**Confidence**
One-feature rules	IF cap-shape = k THEN Y = 0	0.72
	IF cap-surface = y THEN Y = 0	0.54
	IF bruises = f THEN Y = 0	0.69
	IF odor = y THEN Y = 0	0.99
	IF gill-spacing = c THEN Y = 0	0.56
	IF gill-size = n THEN Y = 0	0.88
	IF gill-color = b THEN Y = 0	0.99
	IF stalk-root = ? THEN Y = 0	0.71
	IF stalk-surface-above-ring = k THEN Y = 0	0.93
	IF stalk-surface-below-ring = k THEN Y = 0	0.93
	IF stalk-color-above-ring = p THEN Y = 0	0.69
	IF ring-type = e THEN Y = 0	0.63
	IF spore-print-color = w THEN Y = 0	0.75
	IF population = v THEN Y = 0	0.70
	IF habitat = p THEN Y = 0	0.88
Two-feature rules	IF gill-attachment = f AND ring-number = o THEN Y = 0	0.47
	IF veil-color = w AND ring-number = o THEN Y = 0	0.47
Alternative p-value	$p^{alt} = p(Y = 0) = 0.96$	
VA probabilistic pred.	$0.0074 < Prob(Y = 1) < 0.0078$	
True label	0 (= poisonous)	
Comment	Imputed error detected by CARM and corrected by VA.	

For brevity we do not show the full table of all 81 suspicious examples and their recommended corrections. For the most part, CARM correctly identifies erroneous labels and Venn-ABERS suggests the true one. However, it is worth highlighting cases where mistakes are made, of which there are a total of ten. These correspond to the ten examples marked with black crosses in Figure 1, and Table 5 shows an analysis of them. In each case, the observed and true labels are both 1, but CARM suggests that a label of 0 would look less suspicious, and Venn-ABERS mistakenly predicts a label of 0. That said, these predictions are less confident than for those in Table 4: the alternative p-values are much lower ($\leq 20\%$) and, for most of them, the predicted probabilities are considerably higher than 0%. These predictions therefore have *low credibility*, which might prompt an end user to think twice before making the proposed corrections.

Table 4: CARM and Venn-ABERS outputs for ten most suspicious examples, sorted by p-value

Index	**Lab.**	**p-val.**	**alt. p-val.**	**VA probs**	**VA pred.**	**True lab.**	**Comment**
6600	1	0.00024	0.96	0.0074–0.0078	0	0	1 corrected to 0
800	0	0.00026	0.89	0.990–0.991	1	1	0 corrected to 1
6200	1	0.00048	0.94	0.0074 – 0.0078	0	0	1 corrected to 0
400	0	0.00051	0.84	0.990 – 0.991	1	1	0 corrected to 1
7700	1	0.00071	0.83	0.0074 – 0.0078	0	0	1 corrected to 0
300	0	0.00077	0.90	0.990 – 0.991	1	1	0 corrected to 1
4700	1	0.00095	0.95	0.00780 – 0.98	0	0	1 corrected to 0
1700	0	0.001	1	0.990 – 0.991	1	1	0 corrected to 1
6000	1	0.0012	0.89	0.00737 – 0.00780	0	0	1 corrected to 0
2600	0	0.0013	0.99	0.990 – 0.991	1	1	0 corrected to 1

Table 5: CARM and Venn-ABERS outputs for incorrectly identified errors

Index	**Lab.**	**p-val.**	**alt. p-val.**	**VA probs**	**VA pred.**	**True lab.**	**Comment**
5547	1	0.0076	0.20	0.019–0.077	0	1	1 miscorr. to 0 (low cred.)
4834	1	0.0078	0.20	0.042–0.091	0	1	1 miscorr. to 0 (low cred.)
4899	1	0.0081	0.17	0.17–0.17	0	1	1 miscorr. to 0 (low cred.)
5608	1	0.0083	0.20	0.07-0.11	0	1	1 miscorr. to 0 (low cred.)
5188	1	0.0086	0.17	0.17–0.17	0	1	1 miscorr. to 0 (low cred.)
4647	1	0.0090	0.19	0.17–0.18	0	1	1 miscorr. to 0 (low cred.)
5528	1	0.0090	0.19	0.17–0.18	0	1	1 miscorr. to 0 (low cred.)
5546	1	0.0093	0.20	0.15–0.17	0	1	1 miscorr. to 0 (low cred.)
5601	1	0.0095	0.20	0.10–0.17	0	1	1 miscorr. to 0 (low cred.)
5630	1	0.0098	0.20	0.14–0.17	0	1	1 miscorr. to 0 (low cred.)

5. Conclusions and further work

In this study, we have demonstrated that integrating a modified version of association rule mining with the CP framework can provide valid and transparent error detection in data labels. The validity property allows for setting a limit on the proportion of false alarms when detecting such errors, while using association rules renders the detection process effective and easily interpretable. Additionally, association rules can serve as the basis for probabilistic analysis using the Venn-ABERS framework. Nevertheless, the approach proposed is arguably oversimplified in some respects, potentially resulting in reduced performance. The following topics warrant further investigation:

- Exploration of more advanced pools of association rules.
- Extension of the error detection and correction methodology for multi-class labels, which would necessitate a more complex approach.
- Development of error correction strategies for data features, not just labels.
- Addressing the challenge of multiple error correction, wherein more than one error may occur in the same data example, and devising a valid processing scheme.

- Setting with continuous or batch wise ingestion of new data.

References

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. 22(2), 1993. ISSN 0163-5808.

Amr Alkhatib, Henrik Bostrom, and Ulf Johansson. Assessing Explanation Quality by Venn Prediction. 2022.

Miriam Ayer, H. D. Brunk, G. M. Ewing, W. T. Reid, and Edward Silverman. An Empirical Distribution Function for Sampling with Incomplete Information. *The Annals of Mathematical Statistics*, 26(4):641–647, December 1955. ISSN 0003-4851, 2168-8990. doi: 10.1214/aoms/1177728423.

Vineeth Balasubramanian, Shen-Shyang Ho, and Vladimir Vovk. *Conformal prediction for reliable machine learning: theory, adaptations and applications*. Newnes, 2014.

R. Becker, B.Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: `10.24432/C5XW20`.

Antonin Berthon, Bo Han, Gang Niu, Tongliang Liu, and Masashi Sugiyama. Confidence scores make instance-dependent label-noise learning possible. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 825–836. PMLR, 18–24 Jul 2021.

Kristof Bohmer et al. Mining association rules for anomaly detection in dynamic process runtime behavior and explaining the root cause to users. *Information Systems*, 90:101–438, May 2020.

H. D. Brunk. Maximum Likelihood Estimates of Monotone Parameters. *The Annals of Mathematical Statistics*, 26(4):607–616, December 1955. ISSN 0003-4851, 2168-8990.

Li Chen, Ningyuan Huang, Cong Mu, Hayden S. Helm, Kate Lytvynets, Weiwei Yang, and Carey E. Priebe. Deep learning with label noise: A hierarchical approach, 2022.

Giovanni Cherubin. Bots detection by Conformal Clustering, 2014.

Giovanni Cherubin et al. Conformal clustering and its application to botnet traffic. In *Statistical Learning and Data Sciences*, volume 9047, pages 313–322, 2015.

Derek Chong, Jenny Hong, and Christopher D. Manning. Detecting label errors by using pre-trained language models. *Association for Computational Linguistics*, pages 9074–9091, December 2022.

Cerdeira A. Almeida F. Matos T. Reis J. Cortez, Paulo. Wine Quality. UCI Machine Learning Repository, 2009. DOI: `10.24432/C56S3T`.

Z. Cui, Y. Zhang, and Q. Ji. Label error correction and generation through label relationships. In *AAAI Conference on Artificial Intelligence*, volume 34(04), pages 3693–3700, 2020.

Wilhelmiina Hämäläinen and Matti Nykänen. Efficient Discovery of Statistically Significant Association Rules. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 203–212, January 2009. doi: 10.1109/ICDM.2008.144.

Gargi Joshi. Anomaly Extraction Using Association Rule Mining. *Int. Journal of Engineering Research and Applications*, 4(1), 2014.

J. Kremer, F. Sha, and Ch. Igel. Robust active label correction. In *Proceedings of Machine Learning Research*, volume 84, pages 308–316, 2018.

Andrian Marcus, Jonathan I. Maletic, and King-Ip Lin. Ordinal association rules for error identification in data sets. New York, NY, USA, 2001. Association for Computing Machinery.

Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411, 2021.

Ilia Nouretdinov, Sergi G. Costafreda, Alexander Gammerman, Alexey Chervonenkis, Vladimir Vovk, Vladimir Vapnik, and Cynthia H. Y. Fu. Machine learning classification with confidence: Application of transductive conformal predictors to MRI-based diagnostic and prognostic markers in depression. *NeuroImage*, 56(2):809–813, May 2011.

G. Patetsky-Shapiro. Discovery, analysis and presentation of strong rules. *Knowledge Discovery and Databases*, 1991.

Matthias Rottmann et al. Automated detection of label errors in semantic segmentation datasets via deep learning and uncertainty quantification, 2022.

Zahra Salekshahrezaee, Joffrey L. Leevy, and Taghi M. Khoshgoftaar. A reconstruction error-based framework for label noise detection. *Journal of Big Data*, 8(1):57, December 2021. ISSN 2196-1115. doi: 10.1186/s40537-021-00447-5.

UCI. Mushroom. UCI Machine Learning Repository, 1987. DOI: 10.24432/C5959T.

Vladimir Vovk and Ivan Petej. Venn-Abers predictors, June 2014.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer Science and Business Media, 2005.

Wei-Chen Wang et al. Detecting label errors in token classification data, 2022.

Yan Yan et al. Partial Label Learning with Batch Label Correction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6575–6582, April 2020. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v34i04.6132.

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*, KDD '02, pages 694–699, July 2002.

Bixiao Zeng, Xiaodong Yang, Yiqiang Chen, Hanchao Yu, and Yingwei Zhang. Clc: A consensus-based label correction approach in federated learning. 13(5), 2022. ISSN 2157-6904.

Guoqing Zheng, Ahmed Hassan Awadallah, and Susan Dumais. Meta label correction for noisy label learning, 2021.

Y Zhou and M Kantarcioglu. On transparency of machine learning models: A position paper. In *AI for Social Good Workshop*, 2020.