Null-Labelling: A Generic Approach for Learning in the Presence of Class Noise

Benjamin Denham*, Russel Pears† and M. Asif Naeem§‡
School of Engineering, Computer and Mathematical Sciences
Auckland University of Technology
Auckland, New Zealand
*ben.denham@aut.ac.nz, †russel.pears@aut.ac.nz, ‡mnaeem@aut.ac.nz
§*School of Computing*
National University of Computer & Emerging Sciences
Islamabad, Pakistan

***Abstract*—Datasets containing class noise present significant challenges to accurate classification, thus requiring classifiers that can refuse to classify noisy instances. We demonstrate the inability of the popular confidence-thresholding rejection method to learn from relationships between input features and not-at-random class noise. To take advantage of these relationships, we propose a novel null-labelling scheme based on iterative re-training with relabelled datasets that uses a classifier to learn to reject instances that are likely to be misclassified. We demonstrate the ability of null-labelling to achieve a significantly better tradeoff between classification error and coverage than the confidence-thresholding method. Models generated by the null-labelling scheme have the added advantage of interpretability, in that they are able to identify features correlated with class noise. We also unify prior theories for combining and evaluating sets of rejecting classifiers.**

***Index Terms*—class noise, noisy not-at-random, rejection, abstention, selective classification**

I. Introduction

For as long as machine learning researchers have strived to improve accuracy achieved by classifiers, noisy data has always presented a major obstacle. This has led to the development of many techniques for mitigating both *feature noise*, where feature values are partially dependent on a stochastic process, and *class noise*, where class labels are partially dependent on a stochastic process. While techniques such as feature selection and noise filtering are commonly used to mitigate feature noise, class noise has the potential to be much more disruptive to the learning process. If class noise is more prevalent in certain regions of the input space, under the so-called *noisy not at random* (NNAR) model [1] then learning an accurate classifier for these regions may be impossible without prior knowledge of the noise mechanism. Such *regions of noise* may represent cases that human labellers found difficult to classify, or they may be inherent to the dataset. For example, when attempting to classify topography based on aerial photography, a mix of class values would be expected for the region of the input space where clouds had obscured photographs [2]. Such not-at-random class noise, which has been identified as a relatively unaddressed problem [1], is the primary concern of this paper.

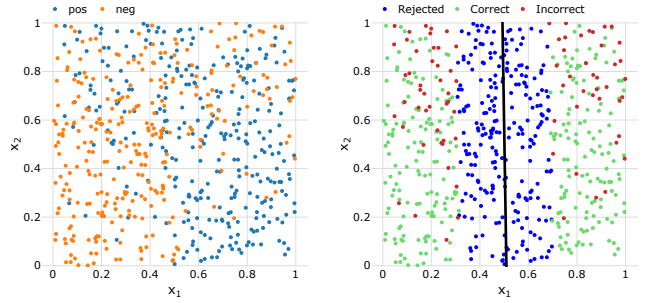
When a dataset contains instances that cannot be accurately classified, one option is to allow the classifier to choose to not classify (or *reject*) those instances. Building such a *rejecting-classifier* makes intuitive sense, as there are many situations where it is preferable for a decision maker to abstain from making a decision in the presence of uncertainty rather than giving their best guess, such as in a medical diagnosis scenario. Depending on the use-case, rejected instances may be forwarded to a more resource-heavy classifier or to a human for manual judgement. The field of *classification with rejection* has also been referred to as *learning with abstention* [3], *selective classification* [4], and *cautious classification* [5].

One of the most prevalent rejection methods is to reject instances where the confidence score or probability estimate provided by the classifier is below a given threshold [6], which we refer to as *confidence-thresholding*. The popularity of this method is largely due to its simplicity of implementation for many common machine learning methods. However, confidence-thresholding may not be the optimal rejection strategy when dealing with class noise. Most classification algorithms are designed to learn optimal decision boundaries under the assumption that every instance must be classified, and therefore may ignore important patterns in the dataset that indicate regions of noise. Because confidence scores are typically related to the learned decision boundaries, they may not be correlated with regions of noise in the input space.

Consider the binary classification dataset in Fig. 1a, which contains a uniform distribution of points that originally expressed the following classification rule: $P(y|x_1 >= 0.5) = 1; P(y|x_1 < 0.5) = 0$, with $P(y)$ denoting the probability of a positive class label. However, introduced class noise replaced each class label with a random class with probability equal to x_2, such that the rule evaluates to: $P(y|x_1 >= 0.5) = 1-\frac{x_2}{2}; P(y|x_1 < 0.5) = \frac{x_2}{2}$. Because the degree of noise is not random but is a function of x_2, this represents not-at-random class noise with a region of noise located in the upper region defined by x_2. We say that x_1 is a *signal dimension/feature* that indicates the class value, while x_2 is a *noise dimension/feature* that indicates the degree of class noise.

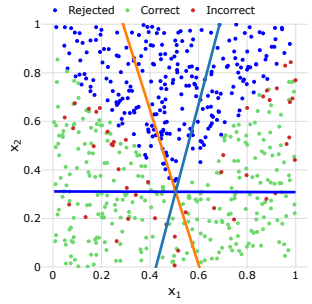
Fig. 1b presents the classification results for a logis-

 DOI of published paper: 10.1109/ICDM50108.2020.00114

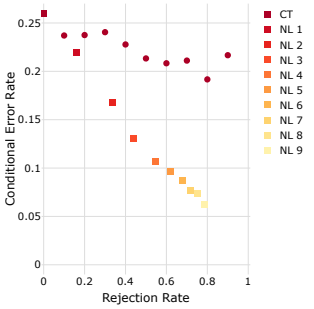


(a) Dataset with not-at-random class noise

(b) LR Classifications with CT (with binary decision boundary)



(c) LR Classifications with NL (with class coefficient boundaries)



(d) Tradeoff plot for classifier without rejection (Base), 9 CT thresholds, and 9 NL iterations

Fig. 1. Comparison of confidence-thresholding (CT) and null-labelling (NL) on a dataset with not-at-random class noise.

tic regression (LR) classifier configured with confidence-thresholding to reject the 40% of instances with the lowest confidence scores. Because there is a uniform distribution of each class along the x_2 dimension, LR produces a vertical decision boundary based solely on x_1. Due to the fact that LR confidence scores are determined by the distance of an instance from the decision boundary, a band of instances are rejected around this vertical decision boundary. However, this leaves a large number of errors in the region of noise (high x_2 values) that have not been rejected, and also results in the unnecessary rejection of instances with low x_2 values and x_1 values close to 0.5 that would be correctly classified without the use of rejection.

This example shows that the classifier is not taking advantage of the noise feature x_2 because it is not discriminative of class boundaries, even though it contains useful information for identifying the region of noise. Such noise features may exist in practice, such as the strength of a radio signal indicating the reliability of other signal measurements.

In this paper, we propose a new rejection method that enables the model to directly learn regions of noise as an explicit `null` class. This is achieved by relabelling instances in the training set with a new `null` class if they are misclassified under a k-fold cross-validation scheme. Fig. 1c demonstrates the classification results after 3 iterations of this method, which we refer to as *null-labelling* (labelling instances with `null`). With null-labelling, LR is able to learn the importance of the x_2 feature for identifying the region of noise, rejecting instances with a high x_2 value while not unnecessarily rejecting correctly classified instances near the "pos"/"neg" decision boundary.

Fig. 1d provides a comparison between confidence-thresholding (CT) and null-labelling (NL) in terms of the tradeoff between the rejection rate and conditional error rate, which is the error rate on non-rejected (*covered*) instances [4]. As the confidence-threshold is raised (and the rejection rate increases), there is little reduction of the error rate - the rejection method is not much better than randomly selecting instances to reject. However, the error rate is drastically reduced by applying an increasing number of null-labelling iterations, demonstrating that it is correctly rejecting instances that were being misclassified in the region of noise.

Our null-labelling method also provides an additional benefit: by treating rejection as an explicit class, we can use model interpretation techniques to gain insights into which regions of the input space are difficult to classify. This is seen in Fig. 1c where the class boundary defined by the coefficients for the `null` class discriminates primarily on the x_2 feature, revealing the relationship between x_2 and the degree of noise. This can aid a user in diagnosing sources of class noise, which they may be able to address in order to provide a cleaner version of the training dataset, thus leading to further improvements in classification accuracy. While other rejection methods have modelled rejection as an explicit class [3], [7], they require alterations to the underlying classification algorithms and loss functions. Null-labelling, on the other hand, is a model-agnostic method that can be applied with any base classifier.

The contributions we make in this paper are:

- A novel model-agnostic null-labelling method for rejection that can achieve a better tradeoff between error and rejection rates than confidence-thresholding in the presence of class noise.
- Interpretation of null-labelled models for identifying features that are correlated with class noise.
- A unification of prior theories to produce a framework for combining rejecting-classifiers and evaluating performance across different rejection rates.

II. Related Work

In this paper we consider class noise to be present when the class label (Y) for an instance depends not only on its input features (X), but also on a stochastic process (E) that distorts the relationship between X and Y. This stochastic process may describe cases where instances with the same input feature values are assigned different classes, or where instances are mislabelled in the training dataset [1]. We therefore consider class noise to be a generalisation of label noise, which only considers the latter case[1]. In particular, we address *noisy not at random* (NNAR) class noise, where E is also dependent on

[1]Though label noise is sometimes also referred to as class noise [8], we make a distinction between the two in this paper.

X - i.e. there are *regions of noise* in the input space where the stochastic process more strongly distorts the relationship between X and Y. This NNAR class noise is a generalisation of the NNAR model for label noise [1], as it does not assume there is a "true" class value that underlies and determines the noisy class label.

There has been extensive research into methods for mitigating label noise. Such methods rely on designing classifiers that model the probability of noisy labels [9] or identifying noisy training instances so that they may be relabelled with the presumably correct class or removed from the training set entirely [8]. These methods assume that the true relationship between X and Y can be learned in spite of label noise. However, any method to identify the instances affected by label noise must rely on prior knowledge of the noise mechanism [10], and errors due to label noise are therefore irreducible if such prior knowledge cannot be obtained or safely assumed. Furthermore, in the more general case of class noise, the stochastic process may affect the true Y values, thereby making it impossible to learn a relationship between X and Y for some regions of the input space. It is these irreducible errors that necessitate rejection, where the classifier is allowed to reject instances expected to have a high probability of misclassification.

Rejection has recently been applied as a means of addressing label noise [7], [11], but it has been extensively researched prior to this. Rejection dates back at least to Chow's rule [12], which rejects instances with classification probabilities below a threshold determined by the relative costs of rejection and misclassification in the application domain. Chow's rule is Bayes-optimal when the classification probabilities are perfect a posteriori probabilities [6], but in practice it is typical for only probability estimates based on confidence scores to be available. This has led to the further refinement of rejection rules based on confidence-thresholding, including the use of class-specific thresholds [5], [6] and metric-based threshold optimisation [13]. However, as we have demonstrated in the previous section, confidence scores are not an optimal means of selecting instances for rejection in general, thus leading to the development of alternative uncertainty measures [14].

The general issue we presented in Section I of confidence scores not being aligned with the optimal regions to reject has been noted previously [3]. This has led to methods that directly learn a rejection function during classifier training that models relationships between input features and noise and can be interpreted to understand those relationships [7]. This research has primarily involved custom classifiers and loss functions [3], [7], [11], [15] that allow the classifier to reject instead of selecting a class (at a user-configured cost). A major drawback to these approaches is that they are model-specific algorithms that require custom implementations, which we address with our model-agnostic null-labelling approach.

In summary, we have highlighted the distinction between label noise and the more general category of class noise, and why the potential for class noise to result in irreducible errors necessitates rejection methods. Much research on rejection has involved confidence-thresholding. While this method is simple to implement and model-agnostic, we have illustrated why it is not an optimal choice for all cases of class noise, and we further discuss the limitations of confidence scores in the next section. Another significant line of research has addressed these limitations with custom classifiers that directly learn a rejection function of the input features. However, progress in this research direction is limited by the need to design and implement model-specific methods. We attempt to bridge the gap between these approaches with our null-labelling method, which is a model-agnostic meta-algorithm that allows the region of rejection to be learned directly from input features while model-interpretation methods of the base classifier can be applied to the rejected region.

III. Foundations for Rejection by Confidence-Thresholding

The goal within the standard classification context is to learn a classifier function $f : \mathcal{X} \rightarrow \vec{y}$ for a given M-dimensional input space $\mathcal{X} = \mathbb{R}^M$ and a set of K classes $\vec{y} = \{c_1, c_2, \ldots, c_K\}$. This is achieved through supervised learning on a training dataset of N instances represented as input/output pairs: $\{X, Y\} = \{(x_i, y_i) | x_i \in \mathcal{X}, y_i \in \vec{y}, i = 1, \ldots, N\}$. This allows a predicted class $\hat{y} \in \vec{y}$ to be produced for any input instance $x \in \mathcal{X}$ as $\hat{y} = f(x)$.

In the context of classification with rejection, the set of possible predictions is extended to include a rejection option $\perp$: $\hat{y} \in \vec{y'}, \vec{y'} = \vec{y} \cup \{\perp\}$. The performance of a rejecting-classifier is typically evaluated in terms of its *conditional error rate* (the error rate on non-rejected/covered instances; $E^c = P(\hat{y} \neq y | \hat{y} \neq \perp)$) and *coverage rate* (the proportion of instances not rejected; $C = P(\hat{y} \neq \perp)$), which are typically evaluated empirically on a test dataset. Coverage can also be expressed as the *rejection rate*, which is the proportion of instances that are rejected: $R = 1 - C = P(\hat{y} = \perp)$. Ideally, we would like a rejecting-classifier that achieves a low conditional error rate while also retaining a high coverage rate.

To achieve rejection through confidence-thresholding, we assume the trained classifier can produce a confidence score for any prediction, represented by the function $g : \mathcal{X} \rightarrow \mathbb{R}$. For a given confidence-threshold t the rejecting-classifier function can be defined as:

$$f_t(x) = \begin{cases} f(x) & \text{if } g(x) \geq t \\ \perp & \text{otherwise} \end{cases} \tag{1}$$

Note that a user can achieve any coverage rate by selecting t such that the desired proportion of confidence scores achieved for their test dataset are greater than t. As the likelihood of an instance being misclassified should decrease with a higher confidence score, we would expect that increasing t would decrease the conditional error rate (while also decreasing the coverage rate). However, this entirely depends on the strength of the relationship between confidence scores and the probability of misclassification. In the case of not-at-random class noise, it is important for low confidence scores to be correlated with noise.

Many common classification algorithms (including decision trees and forests, SVMs, logistic regression, and neural networks) learn decision boundaries based on features that are the most discriminative between the target classes (i.e. signal features). This is an optimal strategy when all instances will be classified, but not necessarily when the classifier is allowed to reject some instances. In particular, a rejecting-classifier should also take into account the noise features that discriminate between regions that can be reliably classified and those that cannot. Because linear models like logistic regression base confidence scores on the distances between instances and the decision boundary, a decision boundary not accounting for noise features will result in confidence scores that do not reflect the true reliability of classification. We have demonstrated this with our motivating example in Fig. 1, where decision boundaries based solely on signal features lead to poor confidence-thresholding performance.

For the remainder of this paper, we will compare confidence-thresholding and our null-labelling method using a logistic regression classifier. Logistic regression is a useful model commonly used in medical applications that can often benefit from judicious rejection (so that difficult cases can be manually reviewed or classified by a more costly test). Logistic regression is also valued for the interpretability of its class coefficients, which can be used to identify noise features through application of our null-labelling method.

IV. Proposed Null-Labelling Method for Rejection

In this section, we present our novel null-labelling method for iteratively training rejecting-classifiers. In each iteration of null-labelling, we train a classifier on a modified version of the training dataset where a `null` class is assigned to instances that are misclassified by the previous iteration's classifier. Specifically, if we define the initial set of training labels as $Y^0 = Y$, then for all null-labelling iterations $j > 0$:

$$y_i^j = \begin{cases} \perp & \text{if } f^{j-1}(x_i) \neq y_i^{j-1} \\ y_i^{j-1} & \text{otherwise} \end{cases} \tag{2}$$

where f^j represents the classifier trained on labels Y^j. Note that, in order for the observed misclassifications to be representative of the classifier's true performance, classification of the training dataset must be performed via k-fold cross-validation. Once a classifier has been trained using a null-labelled training dataset as defined by Equation 2, it will have the ability to directly predict `null` (i.e. reject) just as it can predict any other class value.

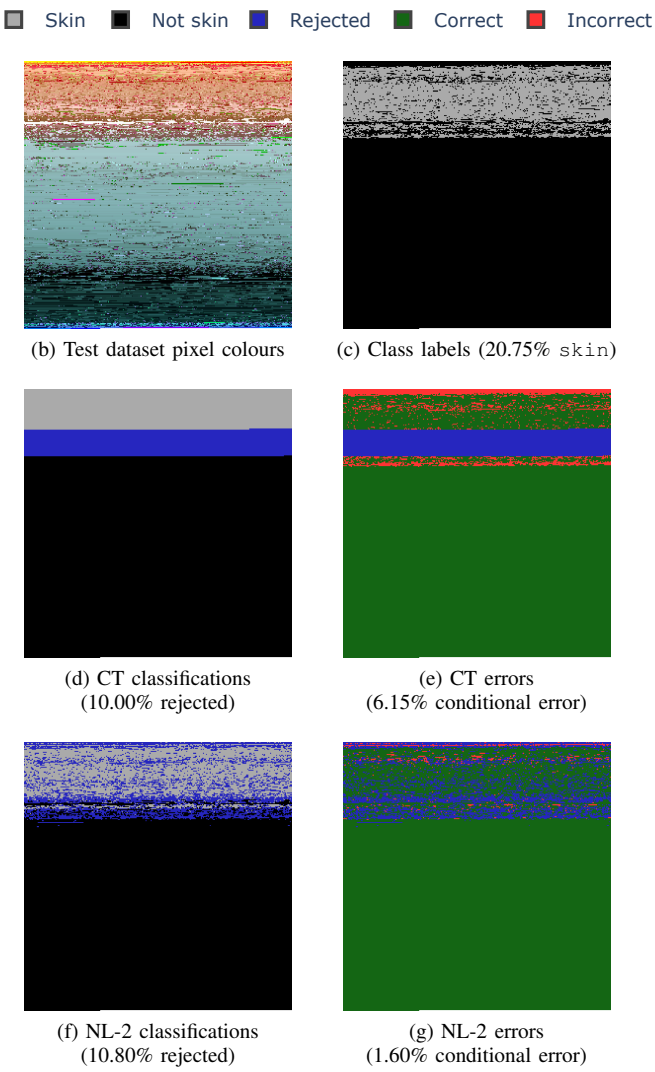
Other rejection methods that enable classifiers to directly reject instances require custom loss functions and implementations [3], [7], while null-labelling can be applied to any base classifier by simply modifying the training dataset. Null-labelling's creation of the `null` class for noisy instances is also distinct from other label noise mitigation methods that alter the training dataset [8], which only seek to remove or correct the class of noisy instances.

By defining a `null` class, we allow the classifier to directly learn a rejection function based empirically on its own misclassifications. Regions of noise that cannot be accurately classified will contain many misclassifications, and will therefore be relabelled to contain many `null` labels. Regions with few misclassifications will only be sparsely null-labelled, allowing the classifier to still learn the majority classes for those regions. Modelling the regions of noise with an explicit class enables the classifier to take advantage of the *noise features* that define these regions. Furthermore, native interpretation methods for the base classifier can be used to identify the noise features that define the `null` class, which we demonstrate in Section VI. Additionally, once null-labelling of the regions of noise has been performed, the remaining regions will have higher class uniformity and be simpler to learn, leading to more reliable decision boundaries for the actual classes. These benefits of null-labelling directly address the issues with confidence-thresholding identified in Section III.

Fig. 2 compares the performance of confidence-thresholding (CT) and null-labelling after two iterations (NL-2) on the UCI Skin Segmentation Dataset [16]. As the goal of this dataset is to predict whether or not a given pixel is from an area of skin based only on RGB colour values, it is reasonable to expect some colours could commonly appear both on skin and elsewhere in an image. A judicious classifier should reject pixels of such colours. We can see that the classifier using null-labelling is able to better identify which colours to reject, achieving a 70% reduction in conditional error over confidence-thresholding when both classifiers only reject approximately 10% of instances. Not only does null-labelling reject the regions that are still misclassified under confidence-thresholding, but regions that were rejected by confidence-thresholding (because they were nearer the decision boundary) are no longer rejected under null-labelling.

Algorithm 1 provides a comprehensive algorithm for implementing null-labelling, including mechanisms for controlling the number of iterations, the achieved coverage rate, and model stability, which we present in the following paragraphs.

Selecting how many null-labelling iterations to perform depends on the properties that are desired of the final classifier. Because null-labelling is a greedy process (instances assigned the `null` class in a previous iteration cannot be reverted to their original class), the coverage rate will tend to decrease with subsequent iterations, typically with a corresponding decrease in conditional error. Therefore, if the goal is to achieve the smallest possible conditional error for a minimum allowable coverage rate (i.e. bounded rejection), iterations can be stopped once the classifier falls below the minimum coverage rate (evaluated on a validation dataset separate to the training dataset), as in lines 4-6 of Algorithm 1. Conversely, if the goal is to achieve as much coverage as possible for a maximum conditional error rate (i.e. bounded error), then iterations can be stopped once the classifier achieves the desired conditional error rate, as in lines 7-9 of Algorithm 1. The maximum number of iterations J can be set as a hard limit on the computation that handles cases where such



(b) Test dataset pixel colours

(c) Class labels (20.75% `skin`)

(d) CT classifications (10.00% rejected)

(e) CT errors (6.15% conditional error)

(f) NL-2 classifications (10.80% rejected)

(g) NL-2 errors (1.60% conditional error)

Fig. 2. Comparison of CT and NL-2 on the skin segmentation dataset. Pixels ordered by pre-rejection logistic regression activations.

constraints are never exceeded. If the above constraints are not given for the application, then J should be set high enough to achieve a broad range of coverage rates. A user can then inspect a tradeoff plot such as Fig. 1d to select their preferred classifier. Alternatively, if the cost of instance rejection (λ_r) can be specified in the range $(0, 1)$ (with 0 denoting the cost of a correct classification and 1 representing the cost of a misclassification), then the cost-optimal classifier can be found by selecting the classifier with minimal expected cost $\mathcal{C}$ as defined in Equation 3 (from [4]) and applied in lines 15-17 of Algorithm 1.

$$\mathbb{E}[\mathcal{C}] = CE^c + (1 - C)\lambda_r \tag{3}$$

If the base classifier is unstable in relation to training set changes, then repeated k-fold cross-validation (CV) can be used to decide which instances to null-label, similar to the label-noise filtering strategy of [17]. The k-fold cross-validation used to identify misclassified instances can be

Input : Training dataset $\{X, Y\}$, Validation dataset $\{X^v, Y^v\}$, Max iterations J, NL rate θ, CV folds k, CV repetitions H, Min CV repetition consensus η, OPTIONAL Min coverage $C_{\min}$ OR Max conditional error $E^c_{\max}$ OR Rejection cost λ_r

Output: A set of potential rejecting-classifiers

1 $Y^0 \leftarrow Y$ **for** $j \leftarrow 0$ **to** J **do**
2 Train classifier f^j on $\{X, Y^j\}$;
3 Evaluate f^j on $\{X^v, Y^v\}$ to determine C_j and E^c_j;
4 **if** $C_{\min}$ *provided and* $C_j < C_{\min}$ **then**
5 **return** $\{f^{j-1}\}$;
6 **end**
7 **if** $E^c_{\max}$ *provided and* $E^c_j \leq E^c_{\max}$ **then**
8 **return** $\{f^j\}$;
9 **end**
10 **for** *k-fold CV runs* $h \leftarrow 0$ **to** H *over* $\{X, Y^j\}$ **do**
11 Train classifier $f^{j,h}$ and confidence-scorer $g^{j,h}$ to cover the domain of X;
12 **end**
13 Construct Y^{j+1} according to Equation 4;
14 **end**
15 **if** λ_r *provided* **then**
 // Minimise cost as defined in Equation 3;
16 **return** $\{f^j | j \in \arg\min_{j \in 0,\ldots,J} \mathbb{E}[\mathcal{C}_j]\}$
17 **end**
18 **return** $\{f^0, \ldots, f^J\}$;

Algorithm 1: Application of null-labelling with stopping criteria and coverage controls.

repeated H times (lines 10-12 of Algorithm 1) with different random splits. This will produce a set of classification results for each training instance, and the set of null-labelled instances can be limited to those that would be null-labelled by a minimum consensus (η) of repetitions. η should be set to represent at least a majority consensus ($\frac{\eta}{H} > 0.5$). If classifier stability is not of concern, then default values of $H = \eta = 1$ can be used for computational efficiency.

Given the above parameters, we can express the complexity of null-labelling as requiring $O(JHk)$ invocations of the underlying base classifier for both training and testing. This complexity is derived from the iterations over J and H on lines 1 and 10 in Algorithm 1, as well as the k-folds in each application of cross-validation. The typical ranges for each of these parameters are relatively small. We have found $J = 10$ is often more than sufficient to achieve an adequate range of coverage rates. H can be set to 1 except when the classifier is unstable to training set changes, in which case $H \leq 10$ is reasonable. Finally, our experiments show that $k = 5$ is sufficient for identifying noisy records via cross-validation.

One challenge to effectively applying null-labelling lies in achieving a desired coverage rate. If the error rate of the original classifier is very low, then a large number of instances

may be null-labelled in even the first iteration, resulting in an initially low level of coverage that only continues to decrease with subsequent iterations. In order to provide more control over the level of coverage achieved by null-labelling, the set of null-labelled instances can be reduced to only include instances misclassified with low confidence scores. This technique essentially uses confidence-thresholding to contribute to the decision of which instances to null-label, and its influence can be controlled by a null-labelling rate θ that determines the proportion of misclassified instances that will be null-labelled. θ should be set to a default value of 1, and only reduced in order to increase coverage when necessary.

We can incorporate the θ, H, and η parameters in a reformulation of Equation 2 (applied on line 13 of Algorithm 1) to construct the training set for each null-labelling iteration:

$$\begin{aligned} \Xi^{j,h} &= L\big(g^{j,h}, \theta, \{x_i | f^{j,h}(x_i) \neq y_i^j\}\big) \\ y_i^j &= \begin{cases} \perp & \text{if } \big|\big|\{h | x_i \in \Xi^{j-1,h}\}\big|\big| \geq \eta, h \in 1 \text{ to } H \\ y_i^{j-1} & \text{otherwise} \end{cases} \end{aligned} \quad (4)$$

where $f^{j,h}$ and $g^{j,h}$ are the classifier and confidence-scoring functions for CV repetition h of iteration j, and $L(r, q, S)$ selects the lowest proportion q of elements in set S as ranked by function r. $\Xi^{j,h}$ represents the proportion θ of instances classified with the lowest confidence scores that were misclassified by CV repetition h of iteration j, which are then aggregated over H iterations so that only instances that appear at least η times are null-labelled.

V. Combining and Evaluating Sets of Rejecting-Classifiers

While the threshold used in confidence-thresholding can be varied to achieve a given coverage rate, the classifier produced by each null-labelling iteration has a fixed coverage rate. We present a framework for combining a given set of rejecting-classifiers into a composite classifier in order to achieve a desired coverage rate. Such composite classifiers are inspired by the concept of "proportionally mixed classifiers" proposed by Ferri & Hernández-Orallo [5].

Two given rejecting-classifiers A and B can be combined to produce a composite classifier D by randomly selecting which classifier to apply with probability $P(A) = 1 - P(B) = p$. To evaluate this composite classifier, we express its performance metrics in terms of metrics for the original classifiers A and B. The conditional error rate E^c we have used so far is intuitive to understand in tradeoff plots (such as Fig. 1d) as the error rate achieved on covered instances. It turns out that the conditional error rate for a composite classifier is a function of the unconditional error and coverage rates for the original classifiers, as shown in Theorem 1. The unconditional error rate represents the error rate achieved over all classified instances, including covered and rejected instances: $E^u = P(\hat{y} \notin \{y, \perp\})$. The unconditional error rate is therefore the product of the conditional error rate and coverage rate, as shown in Equation 5. Consequently, when Algorithm 1 minimises the product of conditional error and coverage contained in the classifier cost (Equation 3), in effect it minimises the unconditional error.

$$\begin{aligned} E^c \times C &= P(\hat{y} \neq y | \hat{y} \neq \perp) \times P(\hat{y} \neq \perp) \\ &= P(\hat{y} \notin \{y, \perp\}) = E^u \end{aligned} \quad (5)$$

We show in the first part of Theorem 1 that the composite classifier's expected unconditional error rate ($\mathbb{E}[E_D^u]$) is a linear combination of the unconditional error rates for classifiers A and B, and that the same property also holds for the expected coverage rate $\mathbb{E}[C_D]$. This is consistent with Ferri & Hernández-Orallo's claim that "proportionally mixed classifiers" can be represented as a linear interpolation between the original two classifiers on a plot of unconditional error against coverage[2].

Theorem 1. *The expected conditional error rate ($\mathbb{E}[E_D^c]$) of composite classifier D is given by $\frac{pE_A^u + (1-p)E_B^u}{pC_A + (1-p)C_B}$, where E_A^u, E_B^u, C_A, and C_B are the unconditional error rates and coverage rates for original classifiers A and B, and p is the probability of applying classifier A.*

Proof:

We first prove intermediate results for the expected unconditional error $\mathbb{E}[E_D^u]$ and coverage $\mathbb{E}[C_D]$ rates:

$$\begin{aligned} \mathbb{E}[E_D^u] &= P(\hat{y} \notin \{y, \perp\}) \\ &= P(A)P(\hat{y} \notin \{y, \perp\}|A) + P(B)P(\hat{y} \notin \{y, \perp\}|B) \\ &= pE_A^u + (1-p)E_B^u \\ \mathbb{E}[C_D] &= P(\hat{y} \neq \perp) \\ &= P(A)P(\hat{y} \neq \perp|A) + P(B)P(\hat{y} \neq \perp|B) \\ &= pC_A + (1-p)C_B \\ \mathbb{E}[E_D^c] &= P(\hat{y} \neq y | \hat{y} \neq \perp) \\ &= P(A|\hat{y} \neq \perp)P(\hat{y} \neq y|\hat{y} \neq \perp, A) + \\ &\quad P(B|\hat{y} \neq \perp)P(\hat{y} \neq y|\hat{y} \neq \perp, B) \\ &= \frac{P(A) \times P(\hat{y} \neq \perp|A)}{P(\hat{y} \neq \perp)} \times E_A^c + \\ &\quad \frac{P(B) \times P(\hat{y} \neq \perp|B)}{P(\hat{y} \neq \perp)} \times E_B^c \\ &= \frac{pC_AE_A^c}{\mathbb{E}[C_D]} + \frac{(1-p)C_BE_B^c}{\mathbb{E}[C_D]} = \frac{pE_A^u + (1-p)E_B^u}{pC_A + (1-p)C_B} \end{aligned}$$

■

Furthermore, the second part of Theorem 1 proves that the composite classifier's expected conditional error rate ($\mathbb{E}[E_D^c]$) can be expressed in terms of the unconditional error and coverage rates of classifiers A and B, mirroring the relationship in Equation 5 ($E^c = \frac{E^u}{C}$). Theorem 2 also proves that this simple formulation for $\mathbb{E}[E_D^c]$ is equivalent to the non-linear interpolation scheme for conditional error proposed by Hanczar [4]. Note that while their notation interpolates between classifiers $\mathbf{X}$ and $\mathbf{0}$ to produce a classifier $\mathbf{0} + \mathbf{x}$, we continue to refer

[2]Though they refer to"unconditional error" simply as error, and plot the inverse of coverage (rejection), which they refer to as "abstention".

to these classifiers as A, B, and D respectively. Based on their notation, classifier A rejects X more instances than B, and D rejects $x = pX$ more instances than B. We also use the notation of G and M as the counts of instances rejected by classifier A that are correctly and incorrectly classified by classifier B ($G = X - M$), N as the total number of instances, $r = NR = N(1-C)$ as the number of instances rejected by a classifier, and $R_\Delta = R_A - R_B = C_B - C_A = \frac{X}{N}$. Finally, we note that the corrected definition of $M = N(E^u_B - E^u_A)$ must be used in place of $M = N(E^c_B - E^c_A)$ as stated by Hanczar, as the E^c values are relative to different coverage rates.

Theorem 2. *Hanczar's method [4] of non-linear interpolation for two classifiers A and B also results in an expected conditional error given by $\mathbb{E}[E^c_D] = \frac{pE^u_A + (1-p)E^u_B}{pC_A + (1-p)C_B}$.*

Proof:

Hanczar sums conditional errors for each possible g correct classifications out of the x extra instances classified by D ($E^{c\{g\}}_D$), weighted by $P(g)$ for binomial distribution $\mathcal{B}(x, \frac{G}{X})$:

$$\mathbb{E}[E^c_D] = \sum_{g=0}^{x} P(g) \times E^{c\{g\}}_D$$

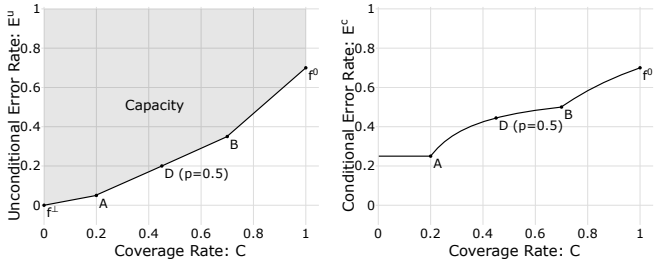
We expand the above summation and express the conditional error in terms of the expected value of g, which is $x\frac{G}{X}$:

$$\begin{aligned}
&= E^{c\{x\frac{G}{X}\}}_D = \frac{E^c_B(N - r_B) - (x - x\frac{G}{X})}{N - r_B - x\frac{G}{X} - (x - x\frac{G}{X})} \\
&= \frac{E^c_B(N - NR_B) - pX + pX\frac{X-M}{X}}{N - NR_B - pX} \\
&= \frac{E^c_B(1 - R_B) - pR_\Delta + pR_\Delta\frac{R_\Delta - (E^u_B - E^u_A)}{R_\Delta}}{1 - R_B - pR_\Delta} \\
&= \frac{E^c_B C_B - pR_\Delta + p(R_\Delta - E^u_B + E^u_A)}{C_B - p(C_B - C_A)} \\
&= \frac{E^u_B - pE^u_B + pE^u_A}{C_B - pC_B + pC_A} = \frac{pE^u_A + (1-p)E^u_B}{pC_A + (1-p)C_B}
\end{aligned}$$

■

This framework for combining rejecting-classifiers also supports use of the capacity curve and metric proposed by Ferri & Hernández-Orallo [5] to compare the performance of sets of rejecting-classifiers. To construct a capacity curve for a set of classifiers, we begin by plotting each classifier's coverage rate against its unconditional error rate. To cover the full range of coverage rates, we also plot points for the non-rejecting base classifier (f^0) and the fully rejecting classifier ($f^\perp(x) = \perp$)[3]. As linear interpolation between any two classifiers represents a composite classifier, the convex hull of classifier points represents a Pareto front where the coverage/error tradeoff of any point above the curve will be dominated by a point on the curve. An example of such a capacity curve is plotted in Fig. 3a. Furthermore, the plot of coverage against conditional error in Fig. 3b demonstrates the results of the non-linear interpolation required for conditional error.

[3]We do not consider artificial variations of rejecting-classifiers with full coverage as Ferri & Hernández-Orallo do, because estimating their performance relies on assumptions of class distributions.



(a) Plotting C vs. E^u (b) Plotting C vs. E^c

Fig. 3. Plotting and interpolation between classifiers.

From an end-user perspective, the conditional error E^c of a classifier should be as small as possible for a given coverage level C. Given that unconditional error decreases monotonically with coverage, we can conclude from Equation 5 that classifiers with a rapid decrease of unconditional error E^u with decrease in coverage C are preferred over ones whose decline is less rapid. In turn, this implies that the area above the plot of unconditional error versus coverage (i.e. the capacity metric) should be as large as possible. Furthermore, the capacity metric evaluates the performance that can be achieved over the full range of coverage rates $[0, 1]$, similar to how the AUC for ROC and PRC curves evaluates performance over all classification thresholds. Therefore, we use the capacity metric to compare the sets of rejecting-classifiers produced by confidence-thresholding and null-labelling in Section VI.

VI. Experimental Study

In the following experimental study, we demonstrate null-labelling's superior handling of class noise as compared to confidence-thresholding. We also explore the impact of the correlation between signal and noise on null-labelling and confidence-thresholding. Furthermore, we interpret the coefficients of the *null* class to identify features correlated with class noise (so-called "noise features") and we demonstrate the use of the proposed θ parameter to increase the coverage rate achieved by null-labelling.

All source-code for the experimentation (including dataset pre-processing) has been made available online[4]. The scikit-learn implementation of logistic regression is applied using the `lbfgs` solver, a maximum of 10,000 iterations (never reached), and a multinomial loss for multi-class datasets. Except where noted otherwise, null-labelling is performed without repetitions ($H = 1, \eta = 1$) on all misclassifications ($\theta = 1$) from 5-fold cross-validation.

To compare confidence-thresholding and null-labelling in the presence of class noise, we inject noise into a set of benchmark datasets from the UCI Machine Learning Repository [16], listed in Table I. These datasets represent a variety

[4]https://github.com/ben-denham/pyrejection

TABLE I
BENCHMARK DATASETS

Dataset	N	K	Num. M	Cat. M
ARM (AReM - top 5 activities)	35,999	5	6	0
BNK (Banknote)	1372	2	4	0
DIA (Diabetic)	1151	2	16	3
ELE (Electrical)	10,000	2	13	0
EYE	14,980	2	14	0
DIG (Handwritten Digits)	1797	10	64	0
GAS	13,910	6	128	0
LED	10,000	10	0	7
LTR (Letter Recognition)	20,000	26	16	0
MUS (Mushroom)	8124	2	0	22
PHI (Phishing)	11,055	2	0	30
SEG (Statlog - Segment)	2310	7	19	0
VEH (Vehicle)	846	4	18	0

of instance (N), class (K), and feature (M) counts (including numerical and categorical features), and many have been used in past studies on rejection. While the Gaussian distribution is commonly used to model feature noise, we use a distribution that simulates not-at-random class noise. To achieve this, we add a `noise_feature` to each dataset with values uniformly distributed in range $[0, 1]$. We then re-assign instance class values with probability increasing exponentially with the `noise_feature`: $P(\text{noise}|x) = \lambda e^{-\lambda s(1-x_{\texttt{noise\_feature}})}$, where $\lambda = 0.8$ to produce an 80% re-assignment rate when $x_{\texttt{noise\_feature}} = 1$, and s is used to scale the $[0, 1]$ `noise_feature` values such that they range over 95% of the probability mass of the exponential distribution, with s computed as the 95% quantile of the distribution: $s = -\frac{\ln(1-0.95)}{\lambda}$. If a class value is re-assigned, it will be replaced with a random selection from all possible class values (which may result in the selection of the instance's original class value). This exponential noise distribution simulates a scenario in which the correlation between input features and class values holds for typical feature values, but not when the noise feature(s) reaches extreme/outlier values. Note that because we inject our own noise, the typical feature noise used with the LED dataset was not included in our generated dataset.

In addition to the benchmark datasets, we perform experiments with synthetic "radial" datasets that vary the correlation between the signal features that determine the class value and noise features that determine the rate of class noise. We expect null-labelling to outperform confidence-thresholding to a greater extent when the signal and noise features are orthogonal to each other. We generate 2-dimensional (x_1, x_2) datasets with instances randomly distributed within the unit circle and an initial binary class value of 1 when $x_1 \geq 0$ and 0 otherwise (i.e. X_1 is the signal feature). Exponentially distributed class noise is introduced by randomly re-assigning class values with probability: $P(\text{noise}|x) = \lambda e^{-\lambda s(1-\omega_x)}$, using the same values of λ and s as for the benchmark datasets, and where ω_x represents a combination of x_1 and x_2 values that is mapped from range $[-1, 1]$ to $[0, 1]$:

TABLE II
RESULTS FOR CT AND NL ON NOISY DATASETS

	Capacity			E^u% @ 80% C			C% @ 50% of E^u		
	CT	NL	σ	CT	NL	σ	CT	NL	σ
ARM	.800	**.805**	.002	**33.3**	34.2	0.3	58.2	58.4	0.5
BNK	.941	**.951**	.007	9.6	**8.2**	1.1	59.5	**73.1**	3.7
DIA	**.859**	.844	.011	**24.9**	26.4	1.8	**61.8**	56.7	2.8
ELE	.937	**.953**	.002	10.1	**8.4**	0.4	59.4	**75.0**	1.7
EYE	.796	**.797**	.004	34.1	34.2	0.5	54.3	54.5	0.6
DIG	.872	**.889**	.009	21.0	20.7	1.4	64.2	**70.4**	2.3
GAS	.891	**.927**	.003	17.5	**13.5**	0.5	54.7	**75.0**	1.2
LED	.893	**.933**	.003	17.7	**12.6**	0.5	54.0	**77.4**	1.5
LTR	.821	**.829**	.002	**31.1**	32.5	0.4	**65.1**	63.9	0.4
MUS	.937	**.951**	.004	10.2	**7.7**	0.6	50.9	**70.6**	4.6
PHI	.933	**.942**	.003	10.7	10.5	0.5	68.0	**72.9**	1.8
SEG	.887	**.908**	.007	19.4	**17.8**	1.2	66.5	**74.4**	2.2
R90	.946	**.957**	.006	8.7	**7.3**	1.0	57.8	**72.2**	4.0
R65	.947	**.956**	.006	8.4	**7.4**	1.0	62.7	**73.8**	4.2
R45	.944	**.954**	.005	9.0	**8.1**	0.9	66.5	**75.5**	4.6
R25	.943	**.950**	.006	9.0	8.6	1.0	72.7	**77.0**	4.7
R00	.942	**.951**	.006	9.0	8.9	1.0	78.1	79.9	3.8

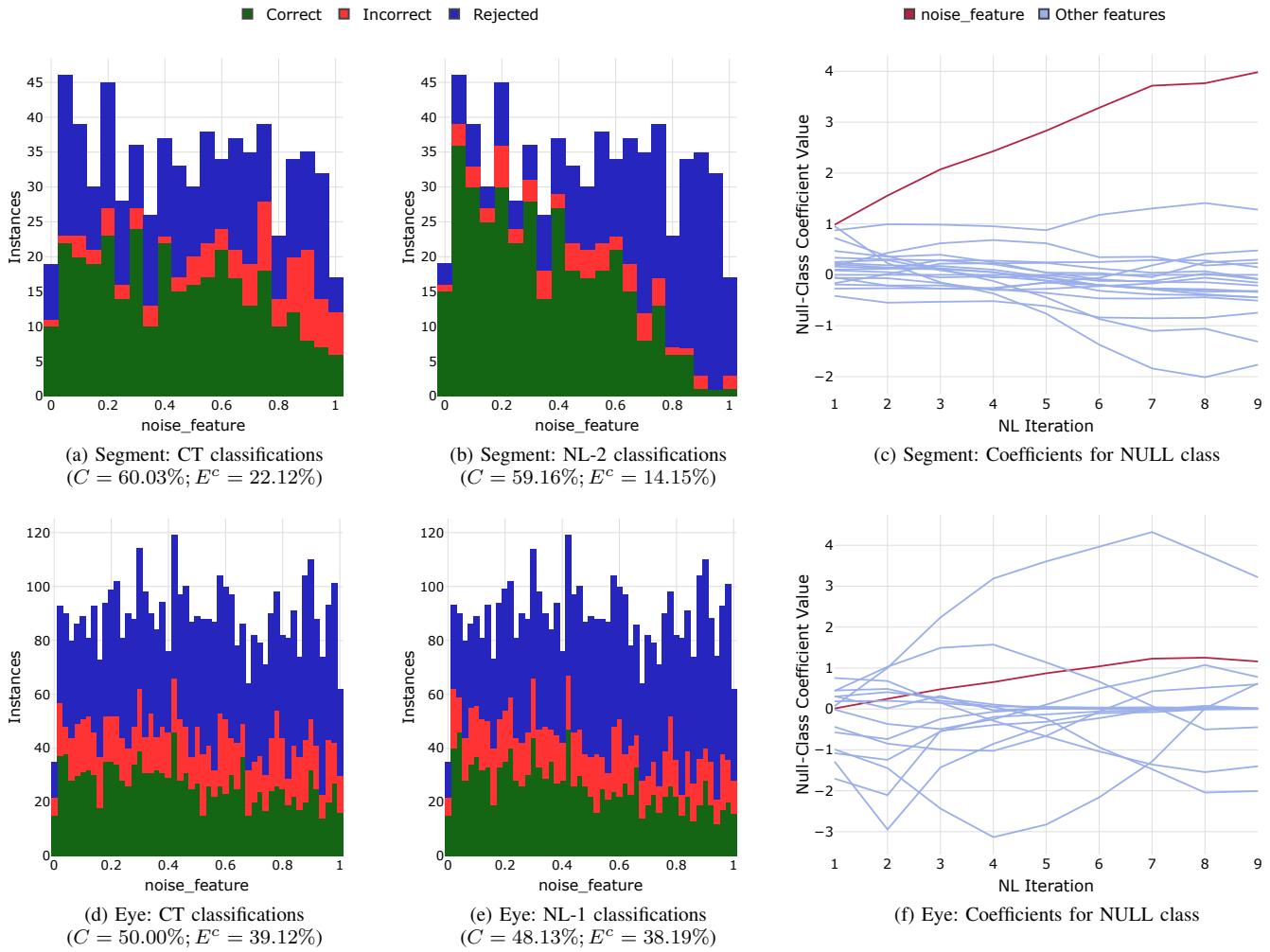
$$\omega_x = \frac{\Big([x_1, x_2] \cdot [\cos(\alpha), \sin(\alpha)]\Big) + 1}{2} \tag{6}$$

where α is an angle that determines the correlation between the signal feature x_1 and the probability of noise. We perform experiments with a range of α values from $0°$ (signal and noise are fully correlated / both determined by x_1) to $90°$ (signal and noise are orthogonal / x_1 is a signal feature, x_2 is a noise feature). These datasets are referred to as `R00` through `R90`.

All datasets are randomly divided into stratified 70/30 train/test splits for performance evaluation, with z-score normalisation applied to numeric features and drop-first one-hot encoding applied to categorical features. Dataset preprocessing is performed before train/test splitting, as the influence of differences in train/test distribution is not part of the evaluation.

We performed tests with confidence-thresholding (CT) and null-labelling (NL) on each dataset. Nine iterations of null-labelling were performed on each dataset, allowing a capacity curve to be plotted for classifiers: $f_0, \ldots, f_9$. For confidence-thresholding, a capacity curve was created for each dataset using ten thresholds that achieved coverage rates $[0.1, 0.2, \ldots, 1]$, matching the number of rejecting-classifiers produced with null-labelling. These capacity curves were used to determine the capacity metric, as well as the unconditional error rate achieved for a coverage rate of 80% (E^u% @ 80% C), and the coverage rate achieved for a 50% reduction in unconditional error rate as compared to the classifier with 100% coverage (C% @ 50% of E^u).

One hundred experiments with different randomly selected train/test splits were performed on each dataset, resulting in one hundred capacity curves and associated metrics for each rejection method on each dataset. The metric means are presented in Table II, along with the maximum standard deviation over both methods (for all metrics and all datasets other than



(a) Segment: CT classifications ($C = 60.03\%; E^c = 22.12\%$)

(b) Segment: NL-2 classifications ($C = 59.16\%; E^c = 14.15\%$)

(c) Segment: Coefficients for NULL class

(d) Eye: CT classifications ($C = 50.00\%; E^c = 39.12\%$)

(e) Eye: NL-1 classifications ($C = 48.13\%; E^c = 38.19\%$)

(f) Eye: Coefficients for NULL class

Fig. 4. Comparison of CT and NL on benchmark datasets.

MUS, NL achieved a lower variance). We performed corrected re-sampled t-tests to test the null hypothesis that there was no significant difference between the metric values achieved with confidence-thresholding and null-labelling. The superior method is bold-faced in Table II where the null hypothesis was rejected with $p < 0.05$ for that dataset and metric. These results were cross-checked with a two-sided Wilcoxon signed-rank test, which rejected the null hypothesis with $p < 0.05$ for all datasets and metrics.

Table II shows that null-labelling significantly outperformed confidence-thresholding on all 3 metrics that we tracked. Null-labelling achieved greater capacity than confidence-thresholding for all datasets except DIA, and was also able to achieve lower error and higher coverage at the fixed points more often than confidence-thresholding. The results with the radial datasets also confirm our hypothesis that improvement of null-labelling over confidence-thresholding tends to increase with the cosine of the angle between the signal and noise features, reaching a maximum at $90°$.

In Fig. 4, we carry out an in-depth examination of two datasets to gain greater insights into the performance of null-labelling vis-a-vis confidence-thresholding. On the Segment dataset, we can see from Fig. 4b that NL captures the relationship between the `noise_feature` and the rate of misclassification: rejecting more instances with high `noise_feature` values, and fewer instances with low `noise_feature` values. However, in Fig. 4a, CT fails to capture this relationship, as the rate of rejection is approximately uniform across the range of `noise_feature` values, thus exposing the limitation of confidence as a measure for estimating noise. The superior noise detection capability of NL explains why it is able to achieve a lower conditional error rate at approximately the same coverage rate as CT. Conversely, the plots in Figs. 4d and 4e show that there is a much higher error rate with little correlation to the `noise_feature` for the Eye dataset, indicating that there are other factors that have a greater impact on classification accuracy than the injected class noise. These inherent misclassifications do not appear to be strongly correlated with any particular "noise" features, resulting in rejection performance from NL that is not much better than that of CT. We observed similar results with the Diabetic dataset, for which NL does not perform as well as

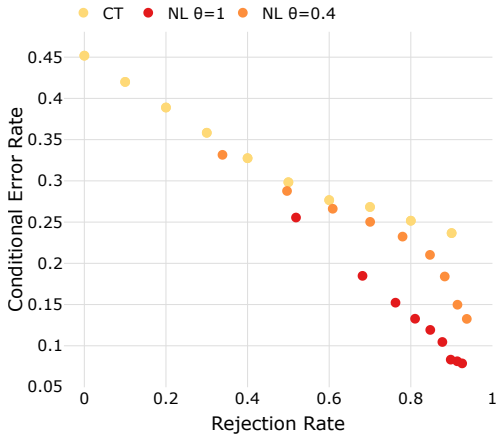


Fig. 5. Increasing NL coverage on letter-recognition.

CT.

Furthermore, Figs. 4c and 4f demonstrate how we can interpret the learned coefficients for null-labelling's `null` class to identify which features are correlated with misclassifications/class noise. For the Segment dataset, the coefficient for the `noise_feature` tends to increase with subsequent null-labelling iterations, indicating that instances with high `noise_feature` values will be assigned the `null` class. Therefore, we can interpret the `noise_feature` as being strongly correlated with the class noise. In the case of the Eye dataset, the `noise_feature` coefficient also increases with subsequent null-labelling iterations, but there are other features with coefficients of greater magnitude, indicating that the `noise_feature` is not correlated with the majority of inherent misclassifications in the dataset. Note that the coefficients for `noise_feature` may not always increase monotonically: when each coefficient is re-learned by logistic regression at each iteration, regularisation will result in coefficient values that are only sufficient for class discrimination relative to all other coefficients.

Finally, Fig. 5 demonstrates use of θ to more precisely control the coverage rate achieved by null-labelling on the letter-recognition dataset. Decreasing the value of θ to 0.4 is able to achieve a lower rejection/higher coverage rate. However, this comes at the cost of conditional error rates closer to those achieved with confidence-thresholding, because we are now using the same confidence scores to influence the selection of instances to null-label.

VII. Conclusions and Future Work

In this paper, we have demonstrated the short-comings of rejection by confidence-thresholding in the presence of not-at-random class noise, and proposed an alternative rejection method: null-labelling. Through iterative learning from the empirical misclassifications of a model, null-labelling enables a classifier to learn the relationships between input features and class noise. We have experimentally demonstrated null-labelling's capability to provide a better tradeoff between coverage and error using an evaluation based on a framework for combining sets of rejecting-classifiers that unifies prior theories. We have also demonstrated how a classification model produced by null-labelling can be interpreted to identify features that are correlated with class noise.

This research has also opened several avenues for future research. Further work is required to explore the impacts of class noise on imbalanced datasets, where class noise may alter the class balance and require more specialised learning and evaluation methods. The knowledge gained by modelling class noise with an explicit `null` class could also be leveraged to further improve models, in a similar fashion to the probabilistic and model-based methods for addressing label noise [1]. We also see potential in applying null-labelling in the context of active learning, as a means of identifying instances to query.

Acknowledgment

This research was funded by a Callaghan Innovation R&D Fellowship Grant (FPAP1902), for a Fisher & Paykel Appliances Ltd project. The first author is also funded by a Doctoral Fees Scholarship from Auckland University of Technology.

References

[1] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 845–869, 2013.

[2] B. A. Johnson and K. Iizuka, "Integrating openstreetmap crowdsourced data and landsat time-series imagery for rapid land use/land cover (lulc) mapping: Case study of the laguna de bay area of the philippines," *Applied Geography*, vol. 67, pp. 140–149, 2016.

[3] C. Cortes, G. DeSalvo, and M. Mohri, "Boosting with abstention," in *Advances in Neural Information Processing Systems*, 2016, pp. 1660–1668.

[4] B. Hanczar, "Performance visualization spaces for classification with rejection option," *Pattern Recognition*, vol. 96, p. 106984, 2019.

[5] C. Ferri and J. Hernández-Orallo, "Cautious classifiers." *ROCAI*, vol. 4, pp. 27–36, 2004.

[6] G. Fumera, F. Roli, and G. Giacinto, "Reject option with multiple thresholds," *Pattern recognition*, vol. 33, no. 12, pp. 2099–2101, 2000.

[7] S. Thulasidasan, T. Bhattacharya, J. Bilmes, G. Chennupati, and J. Mohd-Yusof, "Combating label noise in deep learning using abstention," *arXiv preprint arXiv:1905.10964*, 2019.

[8] J. Luengo, S.-O. Shim, S. Alshomrani, A. Altalhi, and F. Herrera, "Cnc-nos: Class noise cleaning by ensemble filtering and noise scoring," *Knowledge-Based Systems*, vol. 140, pp. 27–49, 2018.

[9] J. Bootkrajang and J. Chaijaruwanich, "Towards instance-dependent label noise-tolerant classification: a probabilistic approach," *Pattern Analysis and Applications*, pp. 1–17, 2018.

[10] M. J. García-Zattera, T. Mutsvari, A. Jara, D. Declerck, and E. Lesaffre, "Correcting for misclassification for a monotone disease process with an application in dental research," *Statistics in medicine*, vol. 29, no. 30, pp. 3103–3117, 2010.

[11] K. Shah and N. Manwani, "Sparse reject option classifier using successive linear programming," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4870–4877.

[12] C. Chow, "On optimum recognition error and reject tradeoff," *IEEE Transactions on information theory*, vol. 16, no. 1, pp. 41–46, 1970.

[13] A. Shrikumar, A. Alexandari, and A. Kundaje, "A flexible and adaptive framework for abstention under class imbalance," *arXiv preprint arXiv:1802.07024*, 2018.

[14] H. Jiang, B. Kim, M. Guan, and M. Gupta, "To trust or not to trust a classifier," in *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 5541–5552.

[15] C. Ni, N. Charoenphakdee, J. Honda, and M. Sugiyama, "On the calibration of multiclass classification with rejection," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 2586–2596.

[16] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[17] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *Journal of artificial intelligence research*, vol. 11, pp. 131–167, 1999.