POLITECNICO DI TORINO

Master's Degree in
DATA SCIENCE AND ENGINEERING

Master's Degree Thesis

To ask or to abstain, what is the best strategy? Finding the best trade-off between: Active Learning and Learning to Reject

Supervisors

Prof. PAOLO GARZA

Candidate

DANIELE GIANNUZZI

December 2022

Abstract

The problem of abstaining from making uncertain predictions has received rising interest in the last few years. However, even if introducing a reject option for a machine learning model in a supervised scenario has already been addressed in many works in literature, it seems to be a completely unexplored field for anomaly detection, where few or no labels are available and making a misclassification can be very expensive for a company. In this work, we introduced a novel technique for anomaly detectors to abstain from making uncertain predictions, introducing a reject option for both unsupervised and semi-supervised scenarios. The novel framework, being based on a dependent rejector making use of the model confidence, is exploitable without regard to the anomaly detector chosen. In unsupervised setting a natural threshold is used to reject samples. On the other hand, in semi-supervised scenario the threshold is tuned using labels, minimizing the overall cost. The cosine distance is used to measure the model reward in using labels for Active Learning or Learning to Reject. Then, a trade-off is found in the usage of labels for one or the other strategy. We evaluated our approach on a benchmark of 9 datasets for anomaly detection. The results show significant performance in rejecting samples for which the misclassification cost could be high. The framework comprised of rejection outperforms the simple Active Learning without rejection both in unsupervised and semi-supervised setting.

Table of Contents

List of Tables iv

List of Figures v

Acronyms vii

1 Introduction 1

2 Related work 5

3 Background 9
3.1 Anomaly detection . . . 9
3.1.1 Unsupervised anomaly detection . . . 9
3.1.2 Semi-supervised anomaly detection . . . 10
3.1.3 Uncertainty in anomaly detection . . . 10
3.2 Active Learning . . . 11
3.3 Rejection . . . 12
3.3.1 Types of rejection . . . 12
3.3.2 Types of rejector architectures . . . 13
3.3.3 Need for a confidence metric . . . 15

4 Methodology 17
4.1 Unsupervised Learning to Reject . . . 18
4.1.1 Confidence metric . . . 19
4.1.2 Introducing the reject option into an unsupervised model . . . 23
4.2 Semi-supervised Learning to Reject . . . 23
4.2.1 Cost metric . . . 23
4.2.2 Tuning threshold . . . 24
4.2.3 Observations on the cost constraints in threshold tuning . . . 25
4.3 Trade-off AL and LR . . . 25
4.3.1 Metric to measure model reward . . . 26

4.3.2 General framework 27

5 Experiments 30
5.1 Data and experimental setup 30
5.2 Results 32

6 Conclusion 43

Bibliography 45

List of Tables

4.1 Three possibilities of using labels (L) or not, being in unlabeled setting (U), for predictor and rejector in semi-supervised setting . . 26

5.1 The 9 benchmark anomaly detection datasets 31

List of Figures

1.1 An example of anomalies and normal data in two-dimensional dataset 1

3.1 Example of binary classification scenario as illustrated in [18]. The dashed black lines represent the rejector, the red solid line the fitted predictor, the orange/green solid line the ground truth function. a) illustrates ambiguity rejection with a non-deterministic relation between the two classes; b) shows the ambiguity rejection performed by a biased model c) is an example of novelty rejection 13

3.2 Architecture of a dependent rejector as in [18] 14

4.1 Anomaly detector: SSDO with 20% dataset labels and IForest as prior. Posterior probabilities computed with 'linear' method. Dataset: Cardiotocography 20

4.2 Anomaly detector: SSDO with 20% dataset labels and IForest as prior. Posterior probabilities computed with 'unify' method. Dataset: Cardiotocography 20

4.3 Count of model confidence values for intervals: [0,0.99); [0.99, 1); {1}. Anomaly detector: SSDO with 20% dataset labels and IForest as prior. Posterior probabilities computed with 'unify' method. Dataset: Cardiotocography 21

4.4 Anomaly detector: SSDO with 20% dataset labels and IForest as prior. Model confidences computed with ExCeeD method. Dataset: Cardiotocography 22

4.5 Count of model confidence values for intervals: [0,0.99); [0.99, 1); {1}. Anomaly detector: SSDO with 20% dataset labels and IForest as prior. Dataset: Cardiotocography 22

5.1 Evaluating the performance of several anomaly detectors: IForest, KNN, LOF, OCSVM with and without reject option. Cost setting: $c_{fp} = 10$, $c_{fn} = 1$, $c_{rej} = 0.5$. 33

5.2 Difference in absolute value between tuned and natural threshold with increasing c_{fp} on logarithmic scale. Anomaly detector: IForest. Cost setting: $c_{fn} = 1$, $c_{rej} = 0.5$. 34
5.3 Difference between cost on test obtained with natural and tuned threshold, with increasing c_{fp} on logarithmic scale. Anomaly detector: IForest. Cost setting: $c_{fn} = 1$, $c_{rej} = 0.5$. 35
5.4 Evaluating the performance of several strategies of using labels for AL or LR or both, compared to the baseline of using only AL without reject option. Cost setting: $c_{fp} = 10$, $c_{fn} = 1$, $c_{rej} = 0.5$. 36
5.5 Evaluating the performance of *Our Framework* using model confidence or confidence based on posterior probability. Cost setting: $c_{fp} = 10$, $c_{fn} = 1$, $c_{rej} = 0.5$. 38
5.6 Distribution of the difference in AL and LR reward for all labels percentages and possible AL/LR strategy permutations. 39
5.7 Evaluating the performance of several strategies of using labels for AL or LR or both, compared to the baseline of using only AL without reject option. Cost setting: $c_{fp} = 1$, $c_{fn} = 10$, $c_{rej} = 0.5$. 40
5.8 Evaluating the performance of several strategies of using labels for AL or LR or both, compared to the baseline of using only AL without reject option. Cost setting: $c_{fp} = 5$, $c_{fn} = 5$, $c_{rej} = 0.5$. 41

Acronyms

AL
Active Learning

AUC
Area Under the Curve

A-R
Accuracy - Rejection Rate

KNN
K-Nearest Neighbors

LOF
Local Outlier Factor

LR
Learning to Reject

OCSVM
One Class Support Vector Machine

ROC
Receiver operating characteristic

SSDO
Semi-Supevised Detection of Anomalies

SVM
Support Vector Machine

Chapter 1

Introduction

The field of interest of this work is anomaly detection [1, 2, 3, 4, 5, 6]. In anomaly detection, the aim is to distinguish the anomalies, sometimes named outliers, from the normal data [1]. In a two-dimensional dataset as the one in Figure 1.1 anomalies can be observed as those points far enough from the dense central cluster. We remember that several types of anomalies exist: *local* having information similar to the normal data; *global* sharing little or no information with normal data [7]; anomalies which form a group.

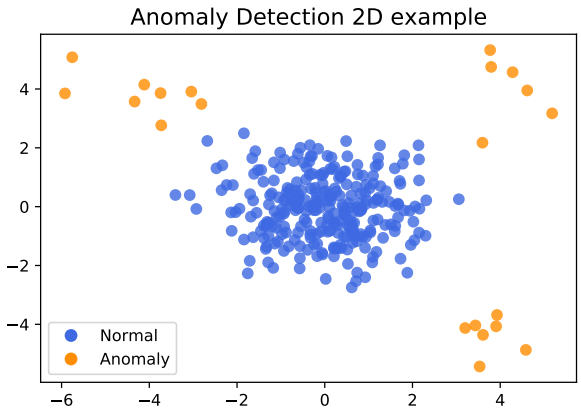

Figure 1.1: An example of anomalies and normal data in two-dimensional dataset

Detecting anomalies is crucial in many scenarios since anomalies are generally related to critical situations for people or machines. We can just think about:

- *Intrusion detection*: it is about malign actions in computer systems [8]. The aim of an intrusion made by a hacker could be to gain access to a network to steal information or to generate problems for the network.

- *Fraud detection*: it usually happens when users which are customers of a company use the resources provided by the company in a suspicious way. It regards credit cards [9], mobile phones, insurance claim [10, 11].
- *Medical detection*: it is generally about patient records. An anomaly in this field could be generated by errors in the machines used or by abnormal conditions of the patient [12].
- *Industrial damage detection*: it is about damages to industrial machines, which can be subject to breakages due to continuous use that must be detected before they can lead to the damage of other components [13, 14].

Since anomalies are rare events [15], in case one could label all the instances, he would probably end up collecting almost always normal labels. In contrast, anomaly labels are the ones that allow the model to learn the most. For this reason, collecting informative labels in anomaly detection is quite expensive and this is why anomaly detection techniques work generally in unsupervised or semi-supervised manner.

Once labels are collected, one can use those to better train an anomaly detector or introduce a reject option so as to abstain from making predictions for those samples which can be easily misclassified. In this way, we can reduce the costs, losses, and damages deriving from making a misprediction. Learning to Reject and Active Learning both focus on the most uncertain regions of the sample space. However, while with Active Learning we focus on uncertain samples at training time [16, 17], the objective of Learning to Reject is to capture the samples for which the predictor is more uncertain at test time [18]. Even if a combination of these two strategies could be beneficial for the final model [19], both these strategies make use of labels.

Therefore, in this work we try to evaluate and optimize the tradeoff in using labels to feed one or the other strategy. First of all, since with rejection a third label is comprised, we defined a new metric to evaluate the performance of the models with rejection, making use of costs for false positives, negatives, and rejected samples. Then we tried to focus on both unsupervised and semi-supervised scenarios, making use of a dependent rejector. In unsupervised scenario we make use of model confidence, rejecting all the samples for which the anomaly detector shows a confidence that is less than the natural threshold. In semi-supervised scenario, we focused on finding a way to tune the threshold, based on minimizing the cost on the validation set. We tried to improve the naive strategies based on using the labels only for the predictor or the rejector, using a model reward metric based on cosine distance to evaluate each time which part of the architecture can produce a greater overall benefit using other labeled samples.

To summarize this work makes the following contributions:

- A novel approach for rejection for anomaly detection in unsupervised scenario
- A novel framework for rejection for anomaly detection in semi-supervised scenario

Chapter 2

Related work

Many works in literature treat ambiguity and novelty rejection in typical machine learning tasks. To the best of our knowledge, there is no literature about ambiguity rejection in anomaly detection, being anomaly detection already a special case of novelty detection.

However, in literature many types of rejection architecture exist for supervised setting: separated, dependent or integrated. Since we used the dependent architecture, we analyze now the main related works, but in a supervised setting. A dependent rejector needs a metric of confidence. This can be based on hard or soft predictions. Works using a metric of confidence on hard predictions are usually founded on ensemble methods. If there is a great disagreement about a prediction, it is supposed that the sample is located into an ambiguous region of the space and for this reason the related confidence will be low so as that sample is rejected [20, 21, 22]. That confidence can be measured for instance using the entropy of the predictions. Another way of measuring the confidence on hard predictions using a single predictor is based on softly perturbing the test sample: in this way if the sample stays in an ambiguous region of the space the associated predictions will differ many times. An example of a perturbation in the field of computer vision can be the rotation or mirroring of an image [23, 24]. However, also methods with confidence score based on soft predictions exist and are widely used in this type of rejection architecture. They are generally based on using the posterior probability in order to estimate the confidence about a prediction. While some models such as the Logistic Regression [25, 26, 27, 28, 29] or Gaussian Mixture Models [30] or Neural Networks with a softmax layer [31] naturally produce a probability as output , some other models need some additional technique to produce the posterior. For some other model one can use ad hoc techniques. For example, in case of a tree-based model, one can estimate the posterior using the entropy or gini, measuring the impurity of each leaf [32, 33]. Another trivial strategy for a model that produces scores as output can be normalizing them in order to estimate the

posterior [34]. After the confidence metric is defined one wants to set a threshold in order to reject or accept the predictions. In literature two main types of thresholds exist: global and local. The underlying assumption to use the global threshold is that the measure of confidence is equally calibrated and accurate over all the sample space. While using the global threshold is simpler since we just need to set a single value, exploiting the local thresholds is surely more difficult as we should set many values. However, the benefit of having more thresholds [35] is that for some region of the space or for some classes the predictor can be more inaccurate, or it can also happen that the misclassification of one class can be more costly than the others [36]. The drawback of this approach is that in order to tune this variety of thresholds one needs many labeled data [37], that is not a trivial requirement for anomaly detection tasks.

Given the possibilities one has to build a dependent rejector architecture, we now describe the use-case-specific works for this type of rejection. In [34] the authors focused the analysis on the medical field concerning the detection of vocal fold paralysis and edema. They use different costs for misclassification in training the linear classifier. However, they use two different rejection costs for one or the other class. Moreover, the thresholds for rejection are asymmetric with respect to the decision threshold learned for the scores. To assess the performance of the model, they use the ROC curve. In [38] again different costs for the rejection and misclassification of false positives and negatives are used. In this case the A-R curve (Accuracy - Rejection Rate) is used to measure the performance of the SVM and Neural Networks comprised of an integrated rejector. Indeed, the predictor and rejector are trained jointly, with a loss function that considers the costs of misclassification and rejection. In [39] they treat the breast cancer diagnosis by means of SVM and KNN and a dependent confidence-based rejector. They try to minimize the average cost due to rejection and misclassification by maximizing the AUC score subject to constraints on the quantity of rejected samples. Then, they evaluate the performance using the Accuracy and AUC with an increasing Rejection Rate. In [33] Random Forest with a dependent rejector working with confidence based on posterior probabilities is used to deal with handwritten character recognition for those applications where the cost of misclassification is high. However, the results show that if Random Forest is used as a classifier producing only one output, abstaining from making predictions is beneficial only for a high rejection percentage. In all the previous works ambiguity rejection is treated.

While in [40] the classification and rejection quality are separately evaluated, that is also an option as described in [41], in many works such as [42] the focus is only on the accuracy in classification, evaluated for different rejection percentages allowed. In this work, the focus is on the classification of the state of wake or sleep based on scores deriving from electrocardiogram (ECG) or polysomnogram

(PSG). In this case, the reliability of classification is based on the distance of the sample from the decision boundaries, as developed in the reliability toolbox for neural networks by [43]. Also in [44] the rejection is based on the distance from the decision boundaries even if in this case SVM is used. In these cases, due to the architecture, both ambiguity and novelty rejection can be performed. In [45] the task is multi-class: handwritten character recognition. The rejector architecture is dependent, then a posterior probability estimation based on the distance of the sample from the prototypes of each class is calculated to produce two metrics of confidence both for ambiguity and novelty rejection. The authors here used Error Rate as a metric of performance, with respect to the Rejection Rate. Another possibility one has to reject is to use an ensemble of classifiers, as it happens in [22] where the authors focus on time-series classification. In particular two classifiers are exploited for the prediction: if they agree the sample output is predicted, otherwise the prediction is rejected. In this case, there is no need for a threshold to be tuned and the architecture is very simple. The authors to evaluate the performance of the model only consider the quality of classification, hence evaluating the accuracy of the model on the accepted samples, disregarding the quality and quantity of rejection.

Chapter 3

Background

In this chapter, we describe the background useful to understand all the techniques used in the next chapter about methodology. We first introduce anomaly detection in unsupervised and semi-supervised setting, with used techniques. Then, a description of rejection follows, with its types and architectures.

3.1 Anomaly detection

3.1.1 Unsupervised anomaly detection

Unsupervised anomaly techniques do not need labels to recognize anomalies, therefore they can be applied to a large number of anomaly detection tasks. They are based on the assumption that anomalies are far rarer than normal data. Every anomaly detection technique uses some underlying heuristic intuition to assign the anomaly scores to the data, where the higher the more anomalous. Here we describe the algorithms used in this work.

IForest: *IForest* [46] is a tree-based approach based on the assumption that anomalies are easier to isolate than normals taking random splits on the features. IForest generates an ensemble of iTrees. Each iTree randomly selects an attribute and then a split value between the maximum and minimum values of the selected attribute. In a tree the splitting is recursively, therefore the number of splittings necessary to isolate a sample is equal to the path length from the root node to the terminating node. Then, anomalies are those samples that have short average path lengths on the ensemble of iTrees.

KNN: K-Nearest Neighbor [47, 48] is based on the assumption that anomalies stay in non-dense regions. They are those samples that are the furthest away from

their K nearest neighbors. Given a distance metric, the basic algorithm measures the distance of each sample to its k^{th} nearest neighbor in a given dataset and determines a threshold for the anomaly score measured by means of this distance. Other versions use the average distance to the k-nearest neighbors.

LOF: Local Outlier Factor [49, 50] is based on the assumption that normal data stay in dense region, similarly to the KNN. It is based on the definition of *K-distance* that is the distance between a sample and its k^{th} nearest neighbor and the *reachability distance* that is the maximum between the K-distance and the distance to the sample of interest, that is used to estimate its density. Than, it compares the density of the sample of interest with the average density of the samples in its neighborhood. If the ratio that is the anomaly score named as Local Outlier Factor is greater than a threshold, the sample is considered an outlier.

OCSVM: One-Class Support Vector Machine [51] is a variation of the SVM used in unsupervised setting. It learns complex regions in the space trying to describe the probability of a sample to belong to each class. To do this, it maps the data into a feature space making use of a kernel function, separating data from the origin maximizing the margin. In practice SVM based techniques profile the normal data into a region of the space and all the data outside this region are considered anomalies.

3.1.2 Semi-supervised anomaly detection

Semisupervised anomaly techniques make use of labels to improve the classification performance. In this work the semisupervised technique named *SSDO (Semi-Supervised Detection of Outliers)* [3] is used.

SSDO: it is a semi-supervised anomaly detector that makes use of labels. SSDO starts from an unsupervised model to have prior knowledge. Then it takes the labels and propagates them to improve the unsupervised score. The closer a sample is to the labeled samples, the greater the influence of label propagation. SSDO can also produce anomaly scores in unsupervised setting using them as prior knowledge. Nevertheless, it was noted that using IForest anomaly scores as a prior, as well as those generated by the best performing unsupervised anomaly detectors, improves the performance.

3.1.3 Uncertainty in anomaly detection

In this work, we deal with rejection, hence also with confidence about predictions. As confidence is based on the probability of a sample being labeled as normal or

anomaly, we are interested in the probabilities produced by an anomaly detector.

Probabilities : Generally, three approaches calculating posterior probabilities from anomaly scores are possible:

- The first one is the *linear* one. It is based on using a simple Min-Max conversion in a way that the anomaly scores are transformed linearly into the range of [0,1]. This transformation is ranking-stable, this means the probabilities rank reflects the ranking of the anomaly scores.

- The second approach is named *unify* [52]. It is based on the concept of unifying outlier scores, performed in two steps. Firstly, *regularization* is applied. It is based on transforming the scores into the interval $[0, \infty]$ in such a way that inliers have a score close to 0 and outliers' scores are much larger than 0. The main reason for re-scaling the anomaly scores is to have enough contrast between outlier and inlier scores since often there is no clear distinction between these two. Secondly, *normalization* is performed to transform the regularized scores into the range [0,1]. In Equation 3.1 gaussian scaling is described, where μ_S and σ_S are the mean and standard deviation of the anomaly scores S, x the sample of interest, while erf is the gaussian error function useful to transform the scores into probabilities.

$$\text{Norm}_S^{\text{gauss}}(x) := \max\left\{0, \text{erf}\left(\frac{S(x) - \mu_S}{\sigma_S \cdot \sqrt{2}}\right)\right\} \tag{3.1}$$

- A third approach named *squashing* is possible [3]. It is based on a squashing function from the exponential group. It squashes the anomaly scores in the range of probabilities [0,1]. The formula is reported in Equation 3.2. $S(x)$ refers to the anomaly score of the sample x, Γ to the scores threshold to distinguish between anomaly and normal samples. Γ is set in a manner so that the percentage of samples with a score greater than 0.5 after the squashing function is equal to the contamination factor. For its nature, also this transformation is ranking-stable.

$$Norm_S^{\text{Gamma}}(x; \Gamma) := 2^{-\frac{S(x)^2}{\Gamma^2}} \tag{3.2}$$

3.2 Active Learning

In anomaly detection, there is abundance of unlabeled data and manual labeling is expensive. In such scenario, it could be beneficial having an Active Learning

strategy. Using this strategy, the anomaly detector can interactively query an expert to label random samples to improve its performance. A clever strategy to query the most informative samples is named *Uncertainty Sampling* [17]. It means the algorithm query to the expert to label those samples on which it is more unconfident. The most popular strategy for *Uncertainty Sampling* is based on entropy used as uncertainty measure as in Equation 3.3, where x is the sample of interest, y the relative output, p refers to the posterior probability of having that output. For binary classification, as it can happen for anomaly detection, using the entropy-based approach for uncertainty sampling is equivalent to query the samples for which the absolute difference in posterior probabilities between normal and anomaly class is closest to 0.

$$x_H^* = \arg\max_x - \sum_i p_\theta \left(y_i \mid x\right) \log p_\theta \left(y_i \mid x\right) \tag{3.3}$$

Another potential approach is the *query-by-committee* [16]. Using this strategy, there is a committee of models trained on the same labeled data. Then, each model votes a label for each sample. The samples with the most disagrees are considered the most informative queries.

3.3 Rejection

3.3.1 Types of rejection

Two main types of rejection exist. Here we briefly describe them with an illustration of the concepts addressed.

1. *Ambiguity rejection*: it is performed in two main cases. The first one is when the model can not distinguish one class from the others due to the way they are distributed in the feature space. Indeed, they can overlap having a non-deterministic relation leading to high probable misclassification at test time. An example of this is shown in Figure 3.1 image a). The second case for which ambiguity rejection is useful is when the classes have a deterministic feature relation but the prediction model can make mispredictions due to its biased nature. In Figure 3.1 image b) we can observe a simple example: in this case the predictor is linear while the classes are non-linear separated.

2. *Novelty rejection*: it is also named outlier rejection. It is exploited to reject those samples which are very different from the ones the machine learning model is trained on. It is well known that machine learning models have trouble in predicting samples that are not in the training data space. We can observe in Figure 3.1 image c) an illustration of novelty rejection.

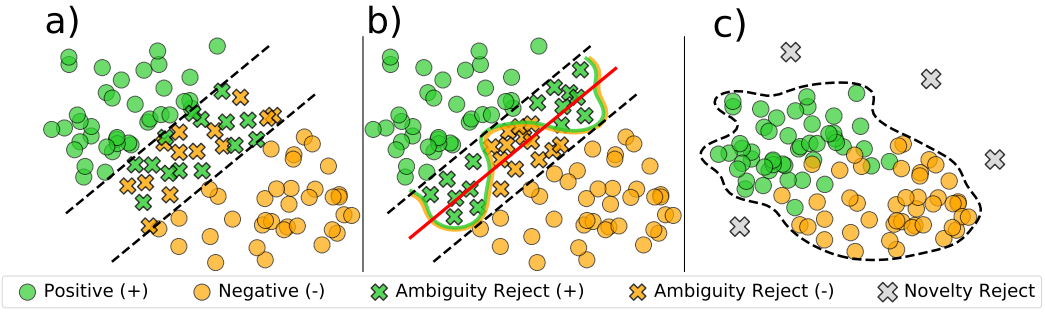

Figure 3.1: Example of binary classification scenario as illustrated in [18]. The dashed black lines represent the rejector, the red solid line the fitted predictor, the orange/green solid line the ground truth function. a) illustrates ambiguity rejection with a non-deterministic relation between the two classes; b) shows the ambiguity rejection performed by a biased model c) is an example of novelty rejection

In this work, ambiguity rejection is performed: normal samples and anomalies are considered the two classes of interest for which the prediction can be ambiguous. In fact, anomaly detection is already a special case of novelty rejection, hence if this type of rejection is performed, one would end up rejecting all the anomalies, while the goal is to detect them.

3.3.2 Types of rejector architectures

In literature, several types of rejector architectures exist.

- *Separated*: the rejector is separated from the predictor. They operate independently. The main benefits are: any predictor can be chosen without changing the rejection part; having a dedicated rejector; the decision about rejecting a sample or not is transparent for the user.
- *Dependent*: the rejector is dependent on the outputs of the predictor, which are used to get confidence values in the predictions made. All the samples below a set global confidence threshold are rejected. In Equation 3.4 it is formally reported a dependent rejector using a threshold. τ refers to the global threshold that needs to be tuned, $c(x, f)$ is the confidence score about the prediction of the sample x made by the predictor f. If $r(x, f) = 1$ the sample x is rejected, otherwise it is accepted.

$$r(x, f) = -sign[c(x, f) - \tau] \tag{3.4}$$

Therefore, a confidence metric is needed for this architecture. The crucial part is that this architecture is highly dependent in performance on this metric. The

main benefits of a dependent architecture are: it allows ambiguity rejection; it is an extension of the predictor, hence adaptation to different predictors can be easily done.

- *Integrated*: rejector and predictor are fused in a single model. The benefits are: both rejection and prediction tasks are optimized when training; there is less bias introduced by rejection.

In this work we focused on the dependent architecture. The separated architecture is based on a dedicated rejector, then it is not suited for the anomaly detection field where few or no labels are available for the training of a separated model. Moreover, it can not take benefit from the misclassification information of the predictor, being only suited for novelty rejection. The dependent architecture is instead well suited for ambiguity rejection. Moreover, it allows us to use existing prediction models for anomaly detection. The integrated architecture is usually the best performing because it optimizes both prediction and rejection at training time. However, it needs a redesign of the predictor, making its use not trivial.

Therefore, formally, the basic objective of this work is to build an anomaly detector with reject option as in Figure 3.2, such that:

1. it does ambiguity rejection using a confidence metric
2. it learns a threshold on the top of the model confidence dependent rejector
3. it learns detector and threshold sequentially
4. it is a discriminative model

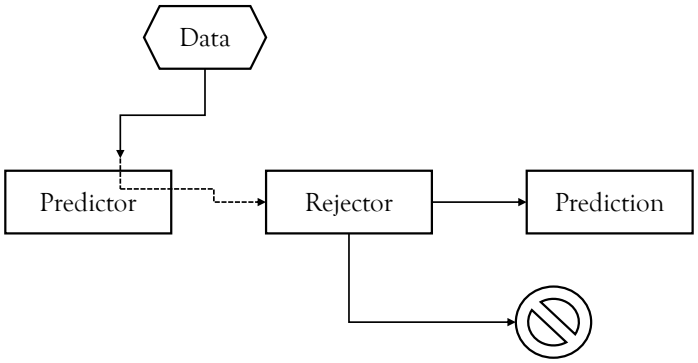

Figure 3.2: Architecture of a dependent rejector as in [18]

3.3.3 Need for a confidence metric

As in this work we want to build a dependent rejector based on a confidence threshold, we need a confidence metric. Two main types of confidence are explored in this work: aleatoric and model-based.

- *Aleatoric*: we simply need to convert the output scores of the model into probabilities with some of the techniques explained in Section 3.1.3, using the entropy, as for Active Learning in Section 3.2, to retrieve a metric of confidence.

- *Model-based*: it is based on slightly perturbing the training data, firstly converting the anomaly scores into probabilities with the Bayesian approach, then deriving the confidence values by observing that the perturbed datasets lead to find different anomaly score thresholds to find hard predictions [2].

  In practice, the method is based on Bernoulli trials. One sample is picked each time. Its anomaly probability is computed as in Equation 3.5, where x is the sample of interest, ls is the number of samples in the training data with an anomaly score lower than x and n is the number of training samples. A prior uniform beta distribution is considered in this computation. Then, Bernoulli trials to get the final example-wise confidence of the model are computed as described in Equation 3.6, where γ parameter is the contamination factor of the dataset.

$$\hat{p}_x := \frac{1 + ls}{2 + n} \tag{3.5}$$

$$c(x) = \sum_{i=n(1-\gamma)+1}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) \hat{p}_x^i \, (1 - \hat{p}_x)^{n-i} \tag{3.6}$$

Chapter 4

Methodology

As described in Section 3.1.1 and 3.1.2, anomaly detection contemplates two main scenarios: unsupervised and semi-supervised. In this work, we introduce a method to implement a reject option for an anomaly detector in both cases. Therefore, the tasks addressed in this work can be formally described as follows.

Task 1:

Given: a dataset X, an anomaly detector, misclassification and rejection costs

Do: introduce a reject option to an unsupervised anomaly detector

Task 2:

Given: a dataset X, an anomaly detector, K labels, misclassification and rejection costs

Do: decide whether to use the labels to learn the model or set rejection threshold

Although achieving these goals is interesting, several problems arise when one would like to accomplish these tasks. First of all, basically a rejector requires labels to tune its parameters, such as the confidence threshold for a dependent rejector. Therefore, in unsupervised scenario where you can not use labels introducing a reject option is not trivial. Moreover, one should be careful at having a method that does not reject too many samples if not necessary, since rejecting a sample is costly due to the fact that an expert has to label it. Secondly, if we consider a semi-supervised scenario, we should find a method to improve the performance of the rejector using labels. The problem is that while in machine learning tasks generally one has at disposal all the samples and can decide how to use them, in anomaly detection we have a small number of labels available. Thirdly, in semi-supervised scenario one generally uses labels to improve the predictor performance. In this case, the problem is that we have also a rejector to train with labels, therefore we

should find a method to reach a trade-off in assigning the few labels we have to train one or the other part of the model. Generally, one would like to assign labels to the strategy that increments the overall performance on the test set more. In order to decide which one between predictor and rejector is the most suitable to do this, one should have a metric so as to evaluate each time how much the model rewards using more labels for the predictor or the rejector.

We found the dependent rejector as the preferable choice for these tasks as described in Section 3.3.2. It generally performs better than the separated one and it easily implementable than the integrated one, for which you need to redesign the model. The unsupervised challenge is addressed by using the model confidence with a natural confidence threshold which approximates the performance of a tuned threshold. For the semi-supervised scenario, a method for tuning the confidence threshold is proposed based on using the labels of the validation set and evaluating with a cost metric the saving of rejecting those samples one by one, choosing the threshold that produces the highest saving. Moreover, to address the problem of having a trade-off between Active Learning and Learning to Reject in using labels, a cosine similarity adapted to the case of 3 output labels (normal, anomaly, rejected) is chosen as a metric of reward. It is needed to select which strategy to use if we have a new budget of labels to spend on improving the performance of our model. In the next sections, all these problems are addressed and a novel framework for anomaly detection with a reject option both in unsupervised and semi-supervised settings is described.

In Section 4.1 given a dependent rejector architecture, we consider the most relevant confidence metrics, evaluating the best one for the unsupervised setting and proposing a method for this scenario, based on a natural confidence threshold. In Section 4.2 we describe an approach for semi-supervised rejection, in particular how to tune a confidence threshold by having a small number of labels making use of a cost metric. The last one is useful also to evaluate the performance of the model comprised of rejection. In Section 4.3 we explore the difficulties behind a trade-off between Active Learning and Learning to Reject in semi-supervised setting, proposing a method to evaluate the reward of assigning labels to the predictor or rejector, based on the cosine similarity, so as to optimize both the prediction and rejection parts of the model.

4.1 Unsupervised Learning to Reject

In unsupervised scenario, we can use labels neither for learning the predictor nor for training the rejector. Even if strategies for learning the predictor already exist as mentioned in Section 3.1.1, learning the rejector is a completely new task. Since we want to learn a rejector that is dependent on the anomaly detector, we should use

the anomaly scores or hard predictions generated by the anomaly detector as input to generate a confidence value for each sample prediction. This confidence value is useful for the rejector both at training and test time. At training time one should use the predictions of the anomaly detector and the associated confidences to set a confidence threshold. At test time one can use the chosen threshold to reject all the samples with confidence below that value. The main problem here is that in unsupervised setting we should set a rejection threshold a priori, without having labels to tune it. Next, the main confidence metrics are discussed, highlighting the most suited for the unsupervised rejection.

4.1.1 Confidence metric

To build a dependent rejector, we must first have a confidence measure of the predictions. Two main methods to estimate confidence in predictions are analyzed in this work.

Posterior confidence

One option we have is using the posterior probabilities. Generally, two approaches to calculate posterior probabilities from outlier scores are possible. These are named *linear* and *unify* and are properly described in Section 3.1.3. Given the posterior probabilities, we need to generate the confidence values. To do that, we can use the same strategy used for Uncertainty Sampling for Active Learning, as described in Section 3.2. Since this strategy is widely used in literature, we chose to adopt it to measure the aleatoric confidence of the instances. In Figure 4.1 one can observe how using the *linear* method, the distribution of confidence value is spread over the entire range of values: [0,1]; while in Figure 4.2 using *unify* approach the values are more skewed to the 1 values.

As we can observe in Figure 4.3 if we use the posterior confidence with 'unify' method, many values have confidence lower than 0.99, while the amount of confidence values between 0.99 and 1 is ignorable. This means that if we set a natural threshold to 0.99 we tend to reject a high percentage of samples.

We do not consider the confidence metric based on *linear* posterior, as well as the one based on *squashing function* to set a natural threshold because we have values of confidence spread between 0 and 1, with a distribution variable from dataset to dataset. Therefore there is no clear strategy for these approaches to find a natural confidence threshold.

Model confidence

The model confidence [2] focuses on the decision boundaries of the predictor. As further explained in Section 3.3.3, for each sample the model confidence indicates

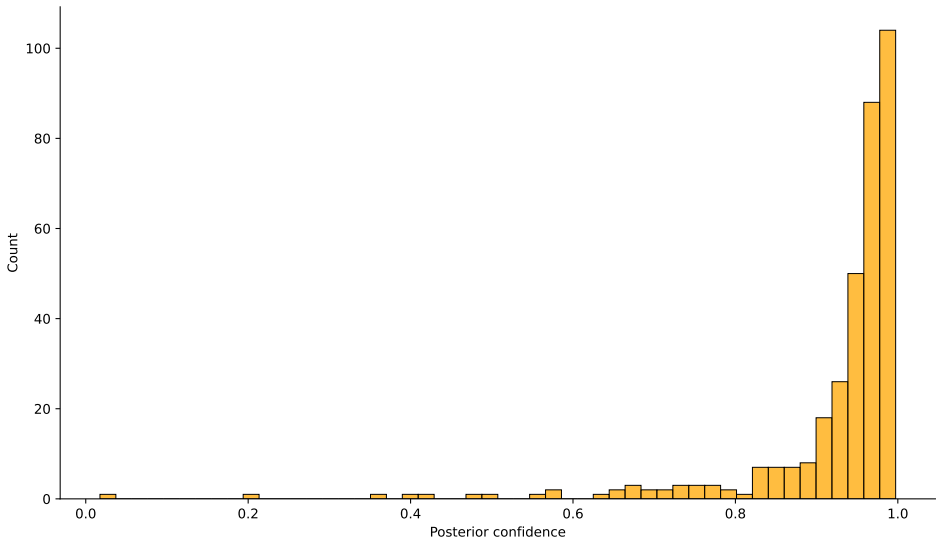

Figure 4.1: Anomaly detector: SSDO with 20% dataset labels and IForest as prior. Posterior probabilities computed with 'linear' method. Dataset: Cardiotocography

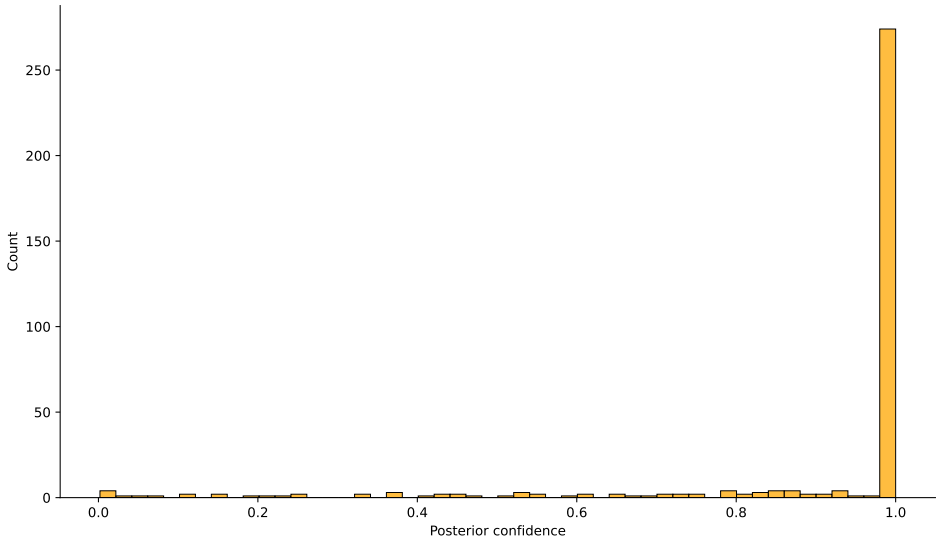

Figure 4.2: Anomaly detector: SSDO with 20% dataset labels and IForest as prior. Posterior probabilities computed with 'unify' method. Dataset: Cardiotocography

the consistency with which the model would make the same prediction in case of perturbation of the training set. It returns a value ranging in [0,1]. In Figure 4.4 one can observe the typical distribution of model confidence values between 0 and 1.

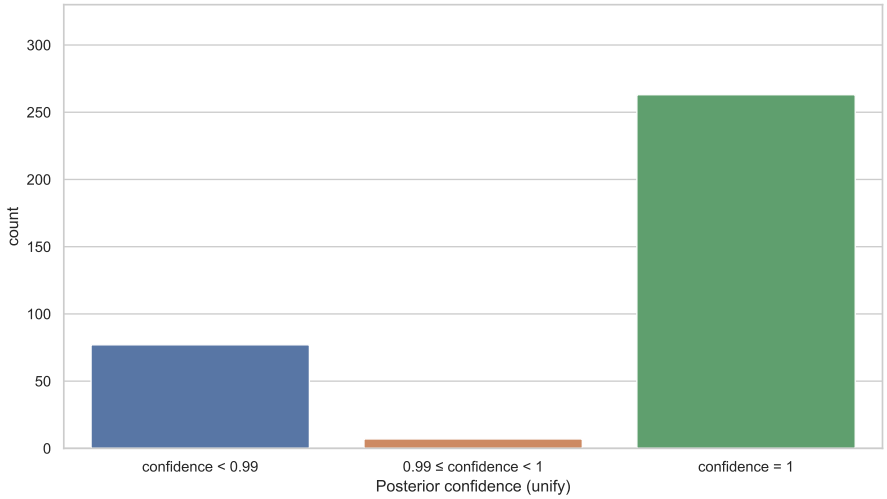

Figure 4.3: Count of model confidence values for intervals: [0,0.99); [0.99, 1); {1}. Anomaly detector: SSDO with 20% dataset labels and IForest as prior. Posterior probabilities computed with 'unify' method. Dataset: Cardiotocography

Using this confidence evaluation method, it is evident that the majority of samples have values equal or very close to 1. This means that it is actually discriminative for a few amount of samples. This translates into the possibility of using a natural threshold (e.g. 0.99) to capture uncertain predictions and reject samples when we are in unsupervised setting.

In Figure 4.5 we show an example reporting how many values of model confidence are in the intervals: [0,0.99); $[0.99, 1)$ and how many values are equal to 1. As we can observe, the majority of values are equal to 1, while the number of values in [0,0.99); $[0.99, 1)$ intervals is similar and very small.

Since the values in the interval $[0.99, 1)$ are not very discriminative as they are mainly due to machine errors and we need an unsupervised strategy that is solid, trying to not reject samples on which the model is confident, we can opt to reject all the samples for which the model confidence is really discriminative. From this analysis, we concluded that in unsupervised setting having a natural confidence threshold of 0.99, being conservative with the amount of rejected samples as rejecting a sample is costly, can be only possible using the model confidence.

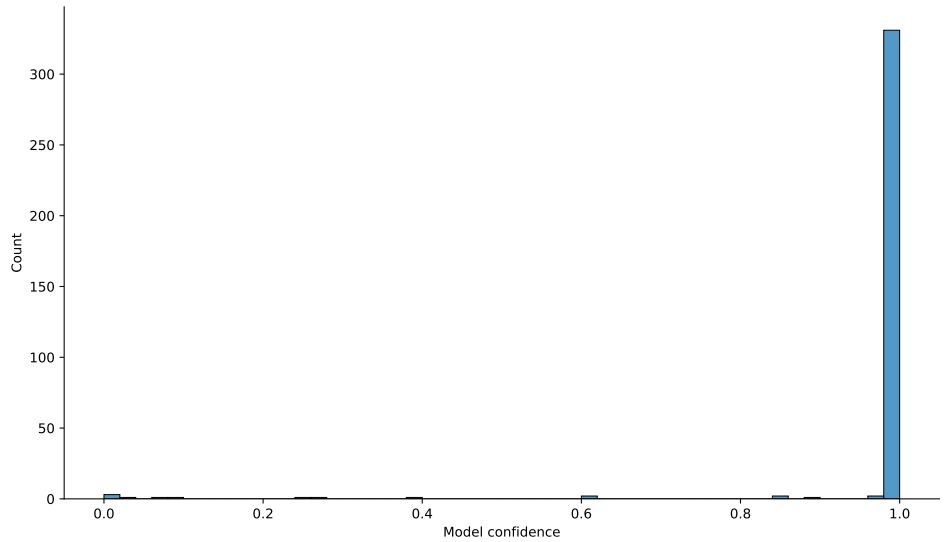

Figure 4.4: Anomaly detector: SSDO with 20% dataset labels and IForest as prior. Model confidences computed with ExCeeD method. Dataset: Cardiotocography

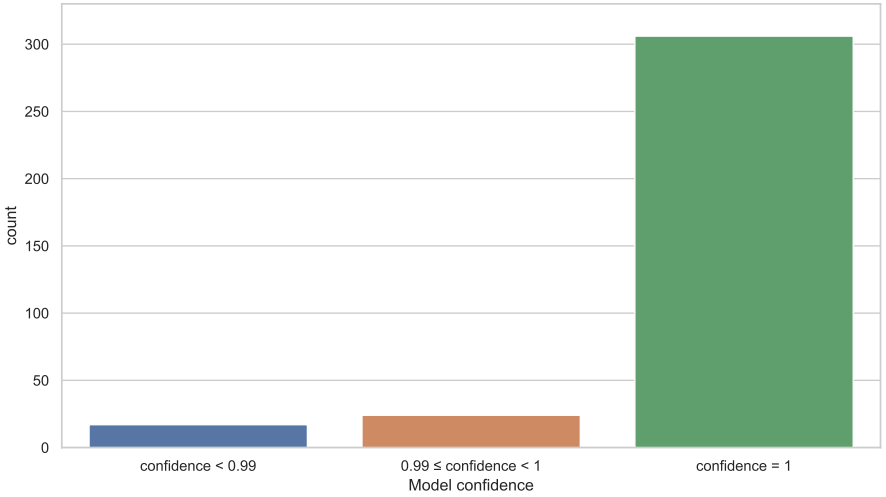

Figure 4.5: Count of model confidence values for intervals: [0,0.99); [0.99, 1); {1}. Anomaly detector: SSDO with 20% dataset labels and IForest as prior. Dataset: Cardiotocography

4.1.2 Introducing the reject option into an unsupervised model

Since the model confidence by nature tends to capture the uncertainty of the model, focusing on the decision boundaries of the learned predictor and since using this confidence metric the majority of confidence values for an unsupervised predictor are equal or very close to 1, we can set a natural threshold to 0.99 a priori.

In Equation 4.1 it is formally reported a dependent rejector using this natural threshold. $c(x, f)$ refers to the confidence score about the prediction of the sample x made by the predictor f. If $r(x, f) = 1$ the sample x is rejected, otherwise it is accepted.

$$r(x, f) = -sign[c(x, f) - 0.99] \tag{4.1}$$

4.2 Semi-supervised Learning to Reject

In semi-supervised setting we can use labels to tune the confidence threshold of the rejector. In this section, we describe the method used to set the threshold as well as the performance metric exploited to do this, useful also to evaluate the performance of the model on the test set.

4.2.1 Cost metric

Having a model made both of a predictor and a rejector, one should evaluate the prediction quality and the rejection quality. The prediction quality is evaluated using the standard metrics: accuracy or F1-score on the non-rejected samples. The rejecting quality is evaluated by the percentage of misclassified samples among the rejected samples. However, finding a metric that combines both these qualities is not trivial [41]. Two main points make this task very tricky:

1. Since we have 3 classes: normal (0), anomaly (1), and rejected (2) we can not use the usual performance metrics such as F1-score, ROC, Precision-Recall. Using these curves we should disregard all the rejected samples, but they still affect the costs. For this reason, we need to consider also rejected samples in our evaluation metric.

2. In anomaly detection misclassifying a sample can be very expensive. Therefore one should evaluate the trade-off between rejection and misclassification costs. Based on the scenario, one can prefer to reject more samples even if there is high probability they are correctly classified, in order to avoid even just one costly misclassification.

Therefore, in our metric we should consider the misclassification cost as well as the rejection cost. The misclassification cost, as in Equation 4.2, is the sum of the product of the number of false positives or false negatives multiplied by their cost of misclassification. The rejection cost, as reported in Equation 4.3, is just the number of rejected samples multiplied by the cost of rejecting a single sample. Hence the total cost used as a metric in this work is just the sum of these costs, as written in Equation 4.4.

$$Misclassification\ cost = (c_{FP} \cdot FP) + (c_{FN} \cdot FN) \tag{4.2}$$

$$Rejection\ cost = c_{REJ} \cdot REJ \tag{4.3}$$

$$Cost = Misclassification\ cost + Rejection\ cost \tag{4.4}$$

If we want to evaluate the performance of the model on the test set as the cost per sample, as in Equation 4.5, we can just normalize the computed *Cost* by the number of samples $|D_{test}|$ present in the test set.

$$Cost\ per\ sample = \frac{Cost}{|D_{test}|} \tag{4.5}$$

It derives that: $Cost\ per\ sample \in [0,\ \max(c_{FP},\ c_{FN},\ c_{REJ})]$.

4.2.2 Tuning threshold

In semi-supervised setting, we can use labels to find the rejection threshold. However, we first need to define what is the rejector training set made of. In practice, since the training set is generally populated by samples with which the predictor is fitted, using the same labels to set the rejection threshold could introduce a bias. Therefore, we use the validation set as the rejector training set. As for Active Learning, samples from the validation set are queried to an expert to get the relative labels. Then, the best threshold is found to minimize the cost on the validation set.

Rejected percentage upper bound. Generally, one wants to avoid rejecting huge percentages of samples. To do that, we set the maximum percentage we allow our system to reject. Then, we take the training set of the rejector (i.e. the validation set), we generate the confidence values for those samples, sorting them and taking the percentile we are interested in. In this way, we have the upper bound for the confidence threshold. It is important to notice that we use both labeled and unlabeled samples from the validation set to exploit more data. It is also beneficial if one uses a sampling strategy to get labels that is different from random sampling, so as to have a distribution that is not biased.

4.2.3 Observations on the cost constraints in threshold tuning

Since we prefer to reject a sample rather than misclassify it, the cost of rejection c_{REJ} should be smaller than both false positive and false negative costs c_{FP} and c_{FN}, as in Equation 4.6. At the same time it can not be too small, otherwise training the rejector in finding the optimal rejection threshold could lead it to set the threshold too high in a way that it tends to reject all the samples.

$$c_{REJ} < c_{FP},\ c_{FN} \tag{4.6}$$

Moreover, given the nature of the cost metric, it is possible that one could have more than one local minimum. To solve this problem we take the threshold associated with the lowest percentage of rejected samples. In this way, we tend to reject the least we can. We tend to prefer to have a model that generates the greatest possible income rejecting as least as possible. What one would like to avoid is to reject all the samples only because only some of them are misclassified.

4.3 Trade-off AL and LR

A semi-supervised anomaly detector benefits from labels in two ways: when using labels to learn its parameters and when using the labels to tune the reject option. In the first case, we are trying to improve the classification performance while in the second the rejection one. Because one can allocate the label budget either ways, our task is to develop a strategy that maximizes the model performance by finding a trade-off in assigning labels to one or the other strategy.

In this case, we need a method that is capable of understanding how much the model is changing with each strategy so as to assign the labels to the one that favors the overall greatest performances, hence the one that minimizes the cost at test time. When we query an expert to label a certain amount of samples we should already know if these labels will be used to further train the predictor (AL) or the rejector (LR), since these labels should be picked from the training or validation set with ad hoc strategies such as Uncertainty Sampling or Random Sampling.

The Table 4.1 shows the three possibilities we have in using labels: we can use them to only train the rejector, using an unsupervised anomaly detector; we can choose to train only the anomaly detector using the natural threshold on the model confidence so as to not use labels for the rejector, or we can find a trade-off in allocating labels to one or the other.

Predictor	Rejector
U	L
L	U
L	L

Table 4.1: Three possibilities of using labels (L) or not, being in unlabeled setting (U), for predictor and rejector in semi-supervised setting

4.3.1 Metric to measure model reward

In order to choose which strategy to use between AL and LR for the next labeling step, we should evaluate how the model reward after training with the new labels. With the term *reward*, we mean how much the model is improving its performance if new labels are given for training. It is important to measure how much is the reward given by each strategy because if we keep adding labels for just one strategy, for instance AL, the reward of the model can decrease. For example, having 99 or 100 percent of labels may not make any difference. So it may be meaningful to combine AL with LR as the model can gain from it as soon as it does not gain from AL. Strategies to measure the reward are heuristic since we can not measure how much the performance is improving at test time. We can not evaluate the model reward using the cost metric on training or validation set because we can have problems with overfitting. Therefore, a novel metric to measure model reward is needed. We have a few metrics to measure how much a predictor is changing with respect to the past. The problem with these metrics is they are suited for the predictor, but they should be adapted in the case of a model consisting also of a rejector. Indeed, with a reject option we can have a third prediction label that corresponds to the rejection choice and makes this metrics not usable except with an adaptation. In our work two reward metrics are considered, with their adaptation to the 3 classes problem:

- **Cosine distance of the predictions**: measures the distance between two vectors [53] of hard predictions $\hat{y}$, as in Equation 4.7. It measures how many predictions changed after model retraining. If the distance of the hard predictions before and after retraining is large, we can conclude that the model changed a lot. This results in a probable improvement of the model. We assume this is an improvement because we imagine that the predictions changed so as to have fewer misclassifications.

$$model\ reward = 1 - \frac{\hat{y}_{next} \cdot \hat{y}_{prev}}{\|\hat{y}_{next}\|_2 \|\hat{y}_{prev}\|_2} \tag{4.7}$$

  To adapt the cosine distance to a third label, when a new sample is rejected,

we simply reverse the prediction that the anomaly detector made in the more recent past when the sample was not rejected. For instance, if a sample is predicted as normal (0) and then it is rejected (2), then a 1 is assigned to consider the decision change for that sample. We use this modified metric both for the AL and LR strategies.

- **Entropy of the predictions**: measures the decreasing uncertainty in predictions. It is formalized in Equation 4.8, where x refers to a sample in D^f_{train} which is the training set of the predictor. H_{next} and H_{prev} refer to the Shannon entropy of the predicted label probability for the retrained and old predictor respectively. This metric is considerable a reward since high values mean that the model changed a lot the probabilities assigned to each class and so also the assigned scores, which we assume is done to improve the predictions at test time.

$$model\ reward = \sum_{x \in D^f_{train}} [H_{next}(x) - H_{prev}(x)] \tag{4.8}$$

  For the AL reward, we use probabilities of a sample to be classified as a normal or anomaly. To adapt the entropy distance to measure how much is the reward using LR, we need to consider rejector probabilities, so the probability that a sample from the validation set is likely to be rejected or not. From confidence measures, we can retrieve the probability of a sample being rejected using the same squashing function as described in Equation 3.2 but using the tuned threshold as Γ parameter.

In this work, we prefer to use the strategy based on cosine distance, as the framework proposed is based on the model confidence, hence not suitable for using the reward based on entropy. Indeed, for the model confidence, the confidence values do not change as much as the labels are given for the rejector training. In this way, the probability of a sample being rejected does not change either. This happens because the majority of confidences using model confidence are close or equal to 1. This results in a model reward for LR strategy that is often too low to be compared to reward resulting from using AL for the predictor.

4.3.2 General framework

Given the strategies to measure AL and LR reward we now explain how to choose the strategy to use when iteratively labeling the samples. When training the first time a model with labels, one should prefer the AL strategy. In fact, as it happens for SSDO, having labels can greatly change the distribution of the confidence values. Then, one should use the next budget of labels to train the rejector, so as to

tune the threshold, trying to improve the performance given by using the natural threshold. Once the AL and LR have both labels, one should use the reward metric to choose the next strategy. We can simply measure the reward of AL considering the outputs on the training set, and the LR reward considering the outputs on the validation set. We can compare the reward of using the first time AL and LR by comparing the new outputs to the ones had in unsupervised setting before having labels to assign. The strategy that produces the greater reward is chosen as the strategy to pursue for the next samples labeling. If the reward is null for both the strategies, it means that the last labeling for each strategy did not produce changes in model outputs. Therefore, when there is a new budget for labeling, the strategy adopted less recently is chosen.

Chapter 5

Experiments

We report the results of the evaluation of our unsupervised and semi-supervised rejection-based algorithm, answering several questions.

Unsupervised setting

Q1) Is adding unsupervised rejection useful to improve the performance of an unsupervised model?

Q2) Is the model confidence natural threshold a good approximation of the threshold set with a fully labeled validation?

Semi-supervised setting

Q3) How does our trade-off strategy perform when compared to using labels only either for active learning or for learning to reject?

Q4) How does the model confidence compare to the posterior confidence as a confidence metric?

Q5) Is the cosine distance-based reward metric balanced in choosing AL and LR strategies?

Q6) Is the framework effective also for other costs schemes?

5.1 Data and experimental setup

Data: We perform our evaluations on a benchmark consisting of 9 anomaly detection datasets. The datasets as reported in Table 5.1 vary in size, number of features, and proportion of anomalies γ.

Dataset	# Examples	# Vars	γ
Cardiotocography	1734	21	0.049
PageBlocks	5393	10	0.094
Arrhythmia	305	259	0.200
Waveform	3443	21	0.029
Shuttle	1013	9	0.013
Wilt	4819	5	0.053
KDDCup99	2500	40	0.080
PenDigits	540	16	0.037
WBC	223	9	0.044

Table 5.1: The 9 benchmark anomaly detection datasets

Experimental setup in unsupervised scenario: We split each dataset into: D_{train}, D_{val}, D_{test}, using a stratified 60-20-20 split. We use 5-Fold validation to split the complete dataset into train and test sets, then we further split the train in D_{train}, D_{val}.

Experimental setup in semi-supervised scenario: We split each dataset into: D_{train}, D_{val}, D_{test}, using a stratified 40-40-20 split. We use a split that is different than the unsupervised case because having the training and validation set of the same size we can give the same possibilities to AL and LR to use labels and to have a reward metric that is comparable. We start in the unsupervised scenario. We incrementally collect 2% of labels, starting from an unlabeled scenario and up to 40% of the dataset size. This results in 20 steps. For each step, we allocate such a 2% budget either to collect labels for the training set (AL) or for the validation set (LR). We use 5-Fold validation to split the complete dataset in train and test sets, then we further split the train in D_{train}, D_{val} in a stratified way. We repeat each experiment 5 times with different random seeds, for a total of $5 \times 5 = 25$ runs. The results are evaluated on the test set D_{test}, as the average of the 25 runs.

Evaluation metric: As evaluation metric we use the cost per sample as explained in the Section 4.2.1. Following the principle of lower is better, it focuses on the costs that misclassification and rejection produce on average on each test sample. The formula used has been described in Equations 4.2, 4.3, 4.4, 4.5. The default cost values used are: $c_{fp} = 10,\ c_{fn} = 1,\ c_{rej} = 0.5$.

Hyperparameters: Regarding unsupervised anomaly detectors, the hyperparameters are set by default [4] as in PyOD[1] for the 4 classifiers *(IForest* [46], *KNN* [47, 48], *LOF* [49, 50], *OCSVM* [51] *)*. For the semi-supervised scenario, as IForest is used as a prior, SSDO predictor has only 2 hyperparameters to be set. Taken by default they are: $k = 30$, $\alpha = 2.3$.

5.2 Results

Q1) Rejection in unsupervised scenario. In this experiment, we investigate whether an anomaly detector would benefit from the reject option based on the natural threshold on the confidence metric. For this task, we exploit 4 widely used unsupervised anomaly detectors and measure their performance with and without the reject option. The boxplots in Figure 5.1 shows the results. As clearly illustrated, for all of the 9 datasets and all of the 4 detectors adding a reject option results in lower costs. Moreover, most of the times it results in a lower variance as well. This means that rejecting samples based on the natural threshold leads to a more accurate and robust anomaly detector. Note that for the dataset Pendigits (center, bottom), OCSVM gets the costs that are 8 times as large as the other models. Thus, we report the value divided by 8.

Q2) Natural threshold compared to the tuned threshold. If we tuned the threshold using a labeled validation set, the model would likely be more accurate and achieve a lower cost. However, if the natural threshold was a good approximation of the tuned threshold, we could allocate the budget on a different task or save it up. Thus, we compare the threshold of a rejector that has labels to tune it with the natural threshold. We increase the cost of the false positive incrementally from 0.5 to 100 and measure the difference between setting the natural threshold and tuning it using a fully labeled validation set. Figure 5.2 shows the results in a logarithmic scale. While we have an increasing cost of false positives on the x-axis, fixed the cost of false negatives and rejection to: $c_{fn} = 1$, $c_{rej} = 0.5$; on the y-axis there is the difference in absolute value between the tuned and natural threshold. As we can observe, the natural threshold approximates in a good way the tuned threshold, as the cost increases. However, we can observe exceptions for Shuttle and WBC datasets. The reason is that both these datasets contain too few misclassifications of false positives (the ones with the greatest cost in this experiment). This is due to: the low number of samples in the dataset, the low contamination factor, and the good performance of the anomaly detector on the dataset. With few false positives, the rejector does not have enough information to tune properly the threshold.

[1] https://pyod.readthedocs.io/en/latest/index.html

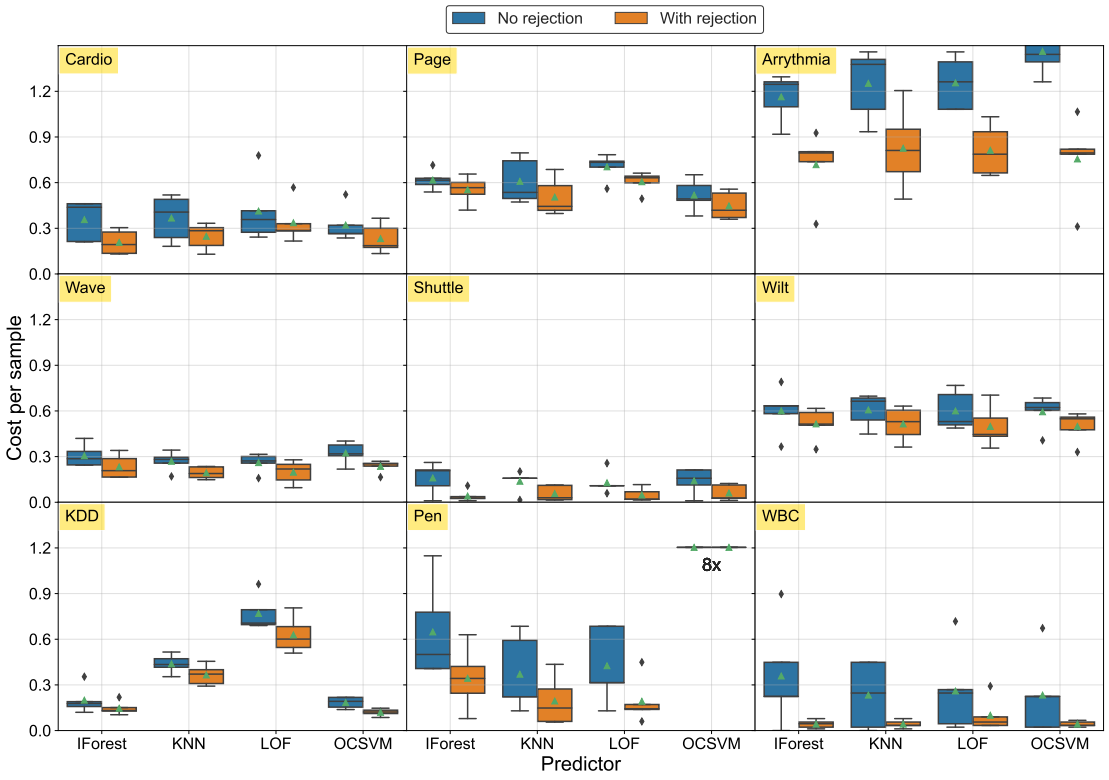

Figure 5.1: Evaluating the performance of several anomaly detectors: IForest, KNN, LOF, OCSVM with and without reject option. Cost setting: $c_{fp} = 10$, $c_{fn} = 1$, $c_{rej} = 0.5$.

In Figure 5.3 the difference of the cost on the test set had using the natural and tuned threshold is reported on the y-axis. We measure the cost on the test set because we want to investigate whether using the approximated threshold results in similar costs. When the cost of false positives is very low, the natural threshold tends to reject a bit more than the tuned one. However, the performance of the cost on the test set is still comparable. This is due to the cost of rejection c_{rej}: if it increases, the performances are no more comparable for a low cost of false positives. Moreover, the more the cost of false positives increases, the more the difference in cost between the two strategies. The reason is that the higher the false positives cost, the higher the probability of tuning a threshold that leads to reject more than the natural one. As we discussed before, we can also observe the results of a threshold tuned without having enough information available on the decreasing trend for Shuttle and WBC dataset, showing that the natural threshold works better than the tuned one.

We can conclude that the natural threshold is a good approximation of the one

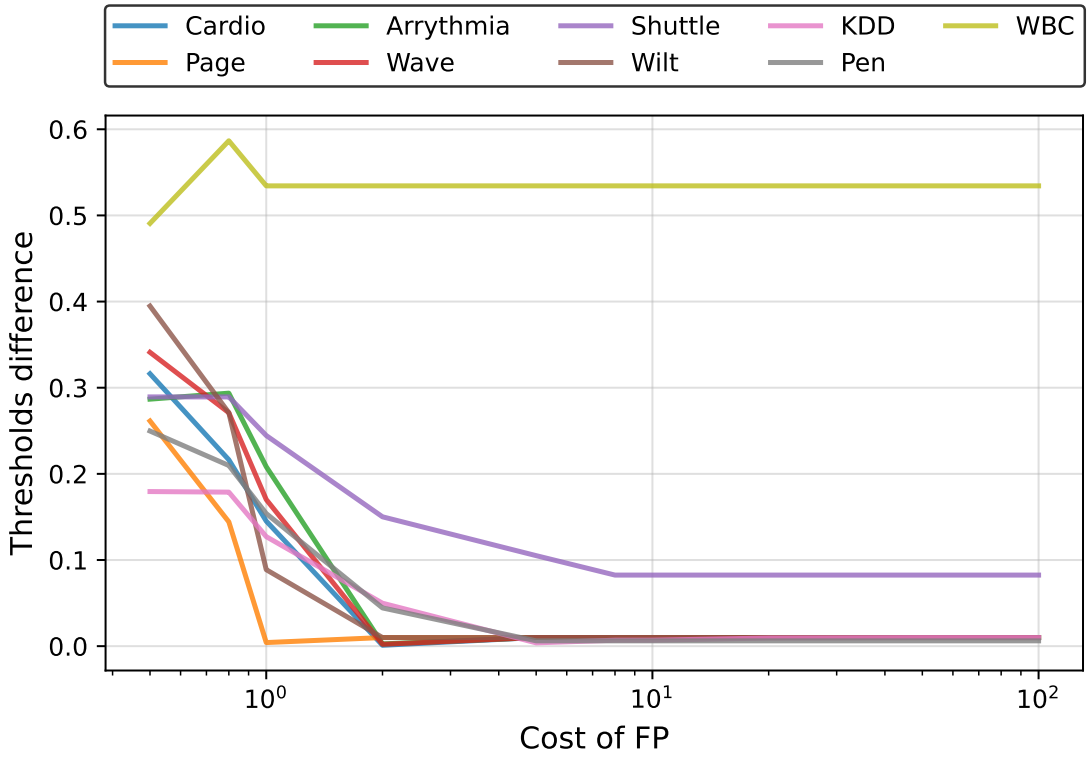

Figure 5.2: Difference in absolute value between tuned and natural threshold with increasing c_{fp} on logarithmic scale. Anomaly detector: IForest. Cost setting: $c_{fn} = 1$, $c_{rej} = 0.5$.

found making use of labels in the majority of cases. However, when the cost of false positives or rejected samples is high, the unlabeled strategy could suffer the approximation.

Q3) Semi-supervised rejection framework. Figure 5.4 shows the performance of *Our Framework* based on SSDO anomaly detector and reject option with labeled model confidence, compared to:

- *All-in AL*: it uses SSDO as anomaly detector, it exploits the whole budget only for AL; it exploits the natural threshold on the model confidence as reject option.
- *All-in LR*: it uses IForest as anomaly detector; it exploits the whole budget only for tuning the threshold on the model confidence.
- *AL without reject option*: it uses SSDO as anomaly detector; no reject option.

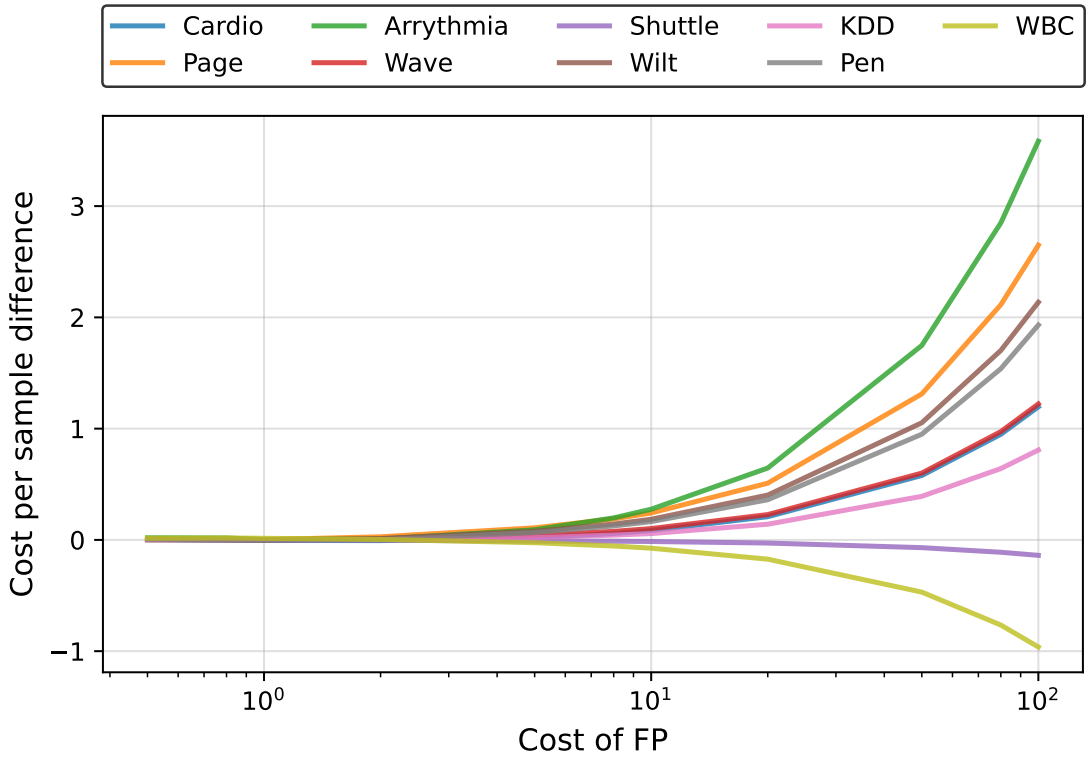

Figure 5.3: Difference between cost on test obtained with natural and tuned threshold, with increasing c_{fp} on logarithmic scale. Anomaly detector: IForest. Cost setting: $c_{fn} = 1$, $c_{rej} = 0.5$.

We need it as a baseline to beat when reject option is included in semi-supervised setting.

As we can observe, models including reject option perform better than the ones without it for 8 datasets out of 9. Solely for the Pendigits dataset, the model without reject option performs better than IForest with reject option from 14% of dataset labels. This happens because the model based on IForest saturates after 8% of labels since the discriminative model confidence values as observed in Figure 4.3 are few. *Our framework* outperforms the others in 4 datasets out of 9 (Cardiotocography, PageBlocks, Waveform, Wilt). The *All-in LR* model outperforms the others only on 1 dataset (KDD). The reason behind this is that SSDO on this dataset shows performance that is poorer as more labels are given. Even if *Our framework* and *All-in LR* thanks to the reject option can limit this drawback, it seems to not be enough to perform better than IForest with LR. As we can notice, rejection is evidently useful in those cases when the model makes a lot of misclassifications. The *All-in AL* model outperforms the others on 3 datasets

(Shuttle, PenDigits, WBC).

We can conclude that our framework seems to perform better than the others when we have a high contamination factor or a high number of samples in the dataset or generally when the anomaly detector makes many misclassifications. Indeed, the more misclassifications are made, the more the rejector can learn from them to better set the threshold. It is clear that if the number of misclassifications on the validation set is small, the framework using the natural threshold (*All-in AL*) can perform better, as happens for Shuttle, PenDigits and WBC datasets. In these cases, the rejection threshold is not tuned properly by *Our Framework* and it results to be always much lower than the natural one set to 0.99. This leads accepting prediction of samples with a low confidence value, for which the probability of making a misclassification is high.

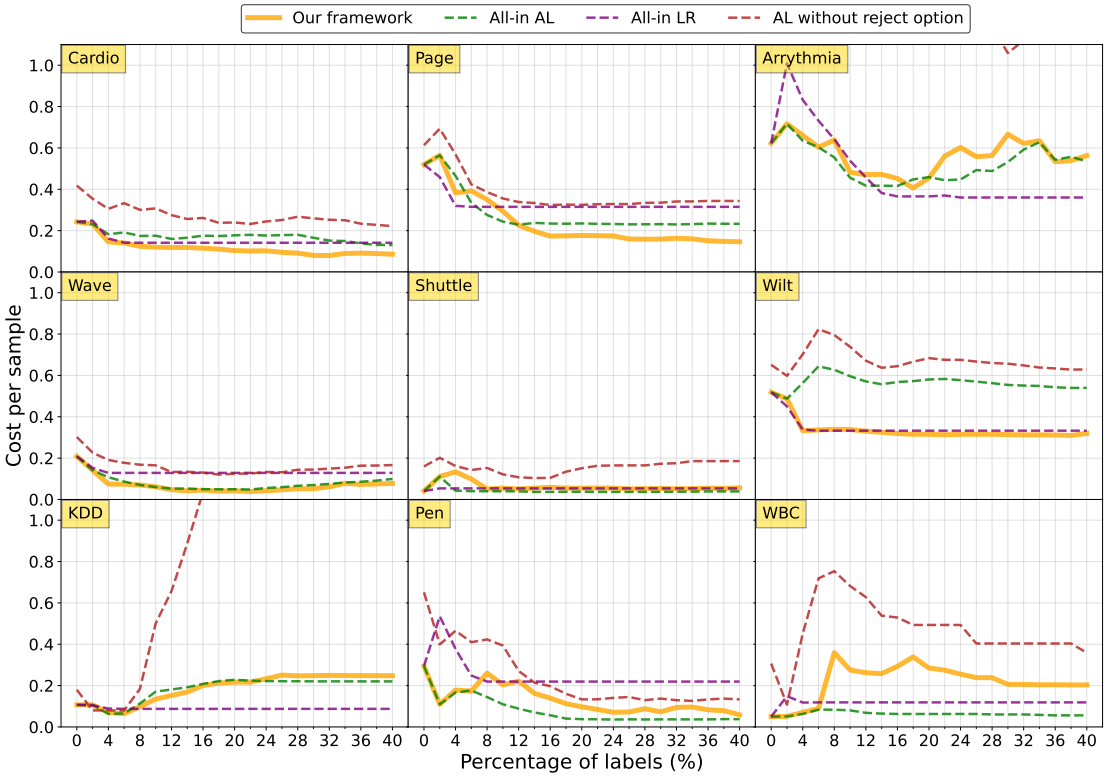

Figure 5.4: Evaluating the performance of several strategies of using labels for AL or LR or both, compared to the baseline of using only AL without reject option. Cost setting: $c_{fp} = 10$, $c_{fn} = 1$, $c_{rej} = 0.5$.

Q4) Model confidence compared to posterior confidence. In this section, we compare *Our Framework* making use of model confidence with its variant using

confidence based on posterior probability. As shown in Figure 5.5, the model confidence outperforms the posterior-based one for 8 datasets out of 9. Only for PageBlock dataset when the percentage of labels goes from 4% to 10% the posterior works dramatically better than the model confidence. With the posterior in that interval, we are rejecting an average of 4.06% samples, while with the model confidence 5.46%. This means that its better performance is not due to a greater percentage of rejected samples. It is simply due to better behavior of the posterior confidence in assigning the confidence values and setting the threshold, given that specific fitting of the anomaly detector. The same happens for KDD dataset from 10% to 18% of labels. In that case, the posterior rejects a mean of 3.45% samples while the model confidence rejects 6.09%. After 20% of labels, the posterior rejects a mean of 10.47% samples while the model confidence rejects 4.82%. Here the anomaly detector worsens its performance a lot due to AL (as observable in Figure 5.4), but while the model confidence is still able to mitigate its worsening, the posterior is not.

We can conclude that the model confidence is the metric of confidence to prefer. In general, it tends to reject less, while the performance is still better than the posterior one, being also able to alleviate the misclassification costs that can rise with AL.

Q5) Distribution in choosing AL and LR strategies. In this experiment, we want to evaluate if the reward metric based on cosine distance is balanced in choosing AL and LR. To do this, we perform a simulation, using labels from 0% to 20% with steps of 2%, considering every possible strategy for each step. Since the first two steps are forced to be respectively AL and LR, we have $2^8 = 256$ possible strategy permutations for each dataset. The cosine distance for AL and LR strategy is evaluated between each step and the corresponding previous one. We briefly explain the method. For 2% labels AL is performed: then the reward of AL with respect to the 0% labels situation is considered; for 4% labels LR is perfomed: then the reward of LR with respect to the 0% labels situation is considered. Then, for 6% labels and so on, if AL strategy is considered, then the reward with respect to the last time AL was performed is considered, while the reward of LR for this step remains the same as before. For every step from 6% to 20% of labels, the difference between AL reward and LR reward is considered. We observe that the difference can only stay in the interval $[-1, 1]$ since the cosine distance varies in $[0, 1]$ interval. The distribution of this difference is shown in Figure 5.6. As we can notice, the distribution is for each dataset skewed to the right with respect to the value 0. This means that AL strategy is preferred most of the times. This behavior was expected since with LR, once a good threshold value is found, small refinements keep most of the predictions unchanged. However, what we are interested in is that one strategy does not dominate the other, so as to exploit all the advantages

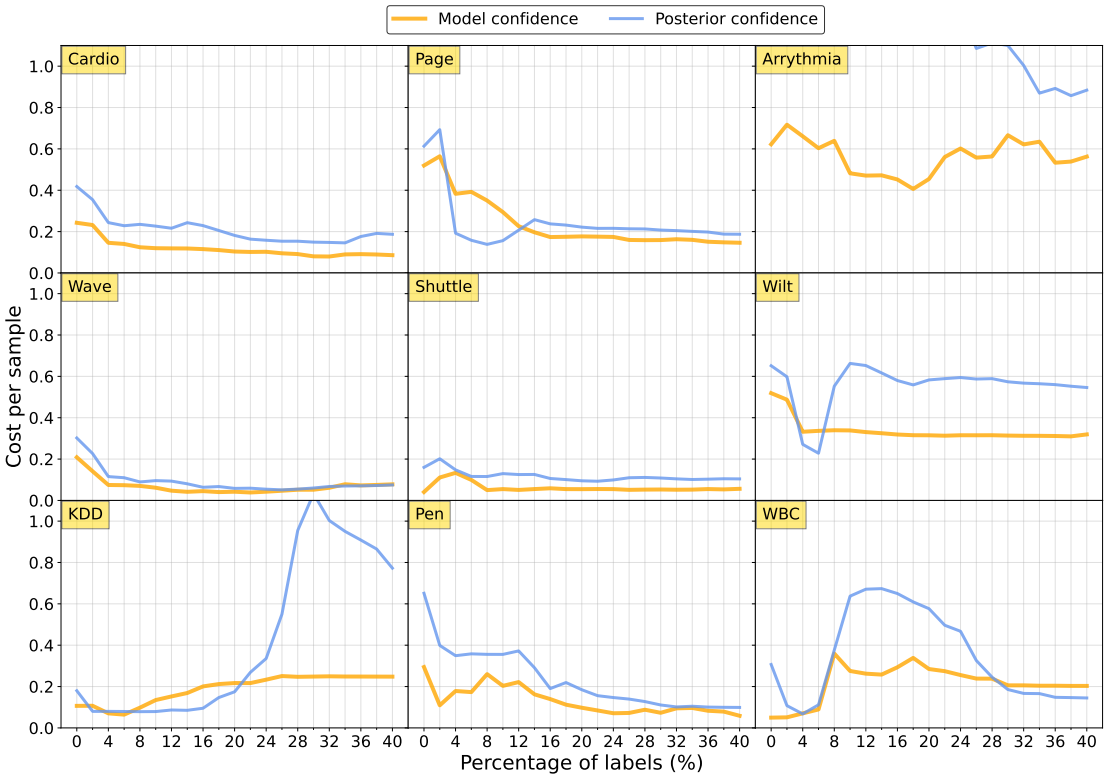

Figure 5.5: Evaluating the performance of *Our Framework* using model confidence or confidence based on posterior probability. Cost setting: $c_{fp} = 10$, $c_{fn} = 1$, $c_{rej} = 0.5$.

of AL and LR. From the graph, we can derive that this happens given the fact that the distributions are almost centered in 0, with the exception of Waveform and Arrythmia datasets, for which AL is preferred by far.

Q6) Framework performance on other cost schemes. We have seen that *Our Framework* as well as *All-in AL* and *All-in LR* strategies outperforms the simple *AL without reject option* when the cost of false positives is higher than the others. Here we want to evaluate the opposite case: when the cost of false negative is the main problem of misclassification. In Figure 5.7 we can observe the performance of the 4 models as in Section 5.2.

In this case, our framework outperforms the others in 3 datasets (Cardiotocography, Waveform, Wilt). In Pageblock it reaches performance better than the others from 14% of labels, when it has enough labels from the validation set to find a good confidence threshold. This happens also for Shuttle dataset from 20% of labels.

The *All-in LR* model outperforms the others on Arrythmia dataset from 16%

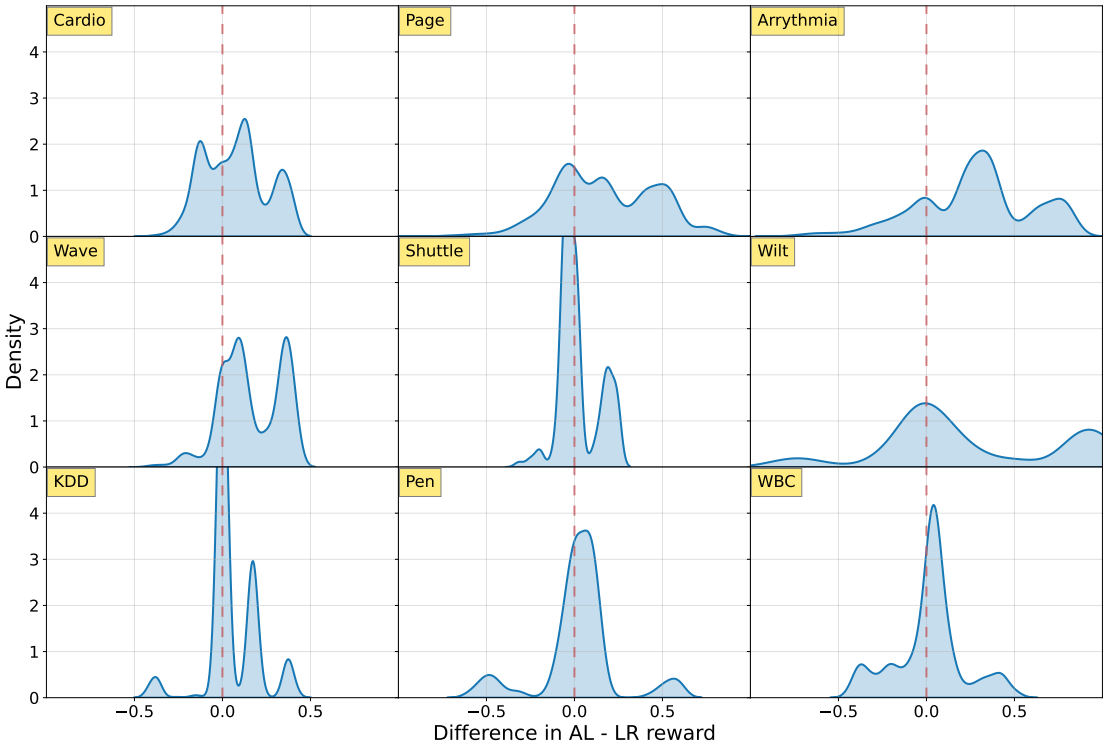

Figure 5.6: Distribution of the difference in AL and LR reward for all labels percentages and possible AL/LR strategy permutations.

of labels, since having more labels for the rejector translates into finding a better confidence threshold, while SSDO based on *AL without a reject option* seems to worsen its performance as more labels are provided, thus resulting in low performances for the rejection strategies based on this anomaly detector. *All-in LR* model outperforms the other strategies also for WBC dataset until 20% of labels. This is because the performance is already great for IForest. Indeed, for the *All-in LR* model the rejection threshold is set to 0 from 6% of labels. The other strategies on WBC dataset suffer from the poor performance of SSDO, until their rejector has enough labels to properly tune the threshold. Indeed, from (20%) of labels *Our Framework* becomes the one preferable.

The *All-in AL* model outperforms the others only on PenDigits dataset, for which the natural confidence threshold seems to improve just a bit the performance of SSDO with *AL without reject option*. Evidently, for this dataset the false negatives are not captured by low model confidence values, while only false positives are. For this dataset, *Our Framework* rejects more than what is really helpful. For this reason, its performance is also worse than the model without rejection.

We can conclude that the lower the contamination factor, the most probable

is that the number of false negatives is lower. For this reason in those datasets such as Shuttle, PenDigits, and WBC for which we also have a small number of samples and a well-performing anomaly detector, having enough false negatives in the validation set to properly set the confidence threshold is even more difficult compared to what was happening with false positive in Section 5.2. Being the false negatives rarer than false positives, it is clear that more labels are needed for the dependent rejector to find a proper confidence threshold.

We must highlight that for the Arrythmia dataset the cost per sample is never lower than the cost of rejection $c_{rej} = 0.5$. This is due to the contamination factor $\gamma = 0.20$ which is much larger than the ones of the other datasets. This clearly influences the number of anomalies that the anomaly detector can misclassify as false negatives. In this case, given $c_{rej} = 0.5$, rejecting all the samples would be the best option, a possibility that is not covered by *Our Framework* that has a rejection percentage upper bound of 40%.

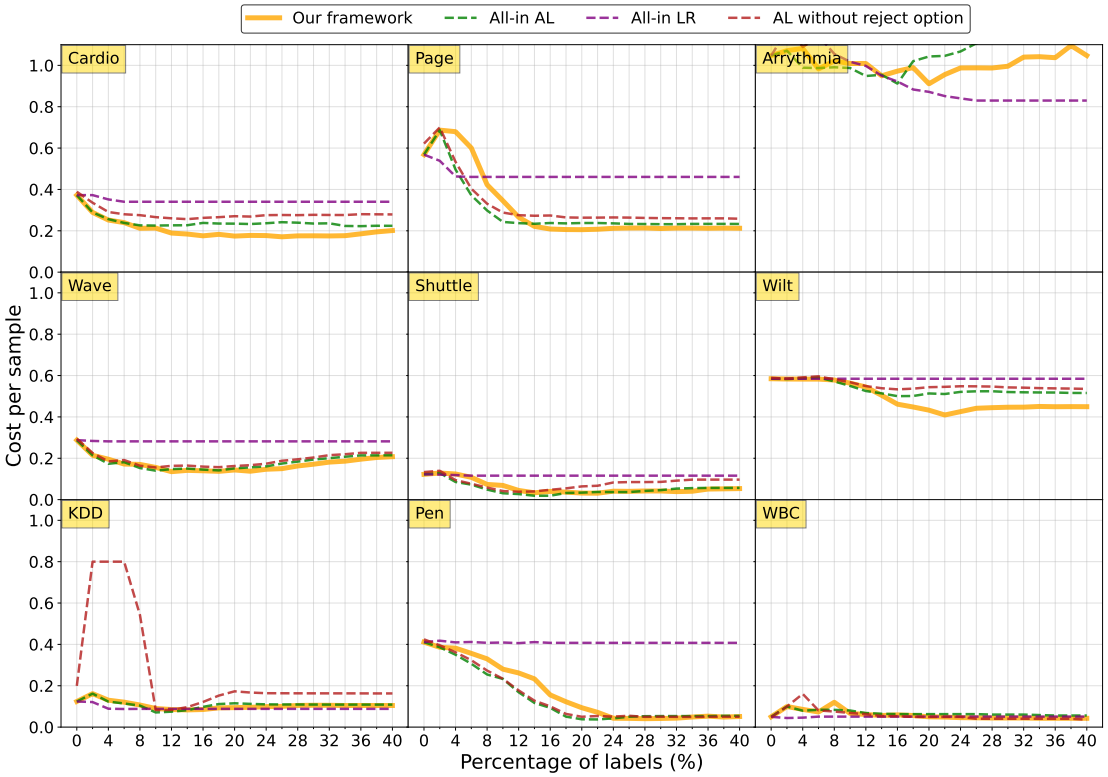

Figure 5.7: Evaluating the performance of several strategies of using labels for AL or LR or both, compared to the baseline of using only AL without reject option. Cost setting: $c_{fp} = 1$, $c_{fn} = 10$, $c_{rej} = 0.5$.

In Figure 5.8 we analyze the situation with equal costs for false positives and

negatives, setting: $c_{fp} = 5$, $c_{fn} = 5$, $c_{rej} = 0.5$. The same considerations made for Figure 5.7 apply also here with the exception of KDD and WBC datasets. In KDD we have again that SSDO after 10% of labels starts to worsen its performance, predicting many false positives, so as to make the model based on unsupervised anomaly detector (*All-in LR*) the one that performs better. For WBC, we have that *Our Framework* performs worse than the other two strategies with the reject option because it sets a threshold that is too low. For example, for 6% of labels, the tuned threshold is 0.43, which leads to reject fewer samples than the other two rejection strategies: for this reason, the curve tends to be closer to the one without rejection. For the *All-in LR* model, the same considerations made for Figure 5.4 apply to this dataset.

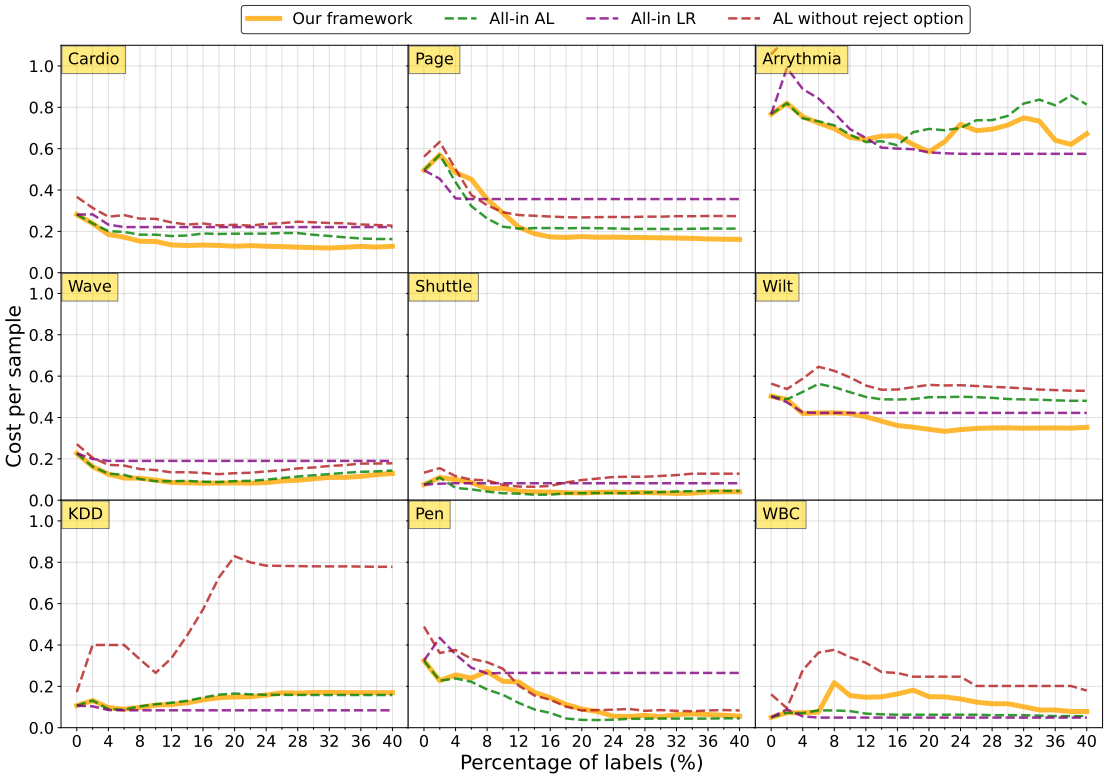

Figure 5.8: Evaluating the performance of several strategies of using labels for AL or LR or both, compared to the baseline of using only AL without reject option. Cost setting: $c_{fp} = 5$, $c_{fn} = 5$, $c_{rej} = 0.5$.

Chapter 6

Conclusion

This work introduced a novel rejection-based framework for anomaly detection. When the scenario is unsupervised and we have no labels to tune the threshold for the dependent rejector, the model confidence with a natural threshold statically set to 0.99 showed good performance in lowering the costs due to the misclassification of an unsupervised predictor, showing in the majority of cases to be a good approximation of the threshold tuned using all the labels of the rejector training set. When we are in a semi-supervised scenario, having labels is useful to better train the predictor or the rejector. A model reward metric based on the cosine distance is used to give labels to Active Learning or Learning to Reject. In case the LR strategy is chosen, samples from the validation set are queried to an expert to be labeled, then the threshold is tuned minimizing the cost on the validation set so as to avoid problems of overfitting. Future works can focus on finding a way to unbalance the rejection in favor of the more costly misclassification since rejection as applied in our framework is symmetric for its nature around the decision boundary. Moreover, one should find a performance metric that is less dependent on hyperparameters so as to be generalizable for many scenarios. Furthermore, as a model reward metric, would be optimal to find a way to give for each of the 3 output classes a probability, such that those sum to 1, then using the entropy to measure how much the model changed after re-training.

Bibliography

[1] Varun Chandola, Arindam Banerjee, and Vipin Kumar. «Anomaly Detection: A Survey». In: *ACM Comput. Surv.* 41 (July 2009). DOI: 10.1145/1541880.1541882 (cit. on p. 1).

[2] Lorenzo Perini, Vincent Vercruyssen, and Jesse Davis. «Quantifying the Confidence of Anomaly Detectors in Their Example-Wise Predictions». In: Feb. 2021, pp. 227–243. ISBN: 978-3-030-67663-6. DOI: 10.1007/978-3-030-67664-3_14 (cit. on pp. 1, 15, 19).

[3] Vincent Vercruyssen, Wannes Meert, Gust Verbruggen, Koen Maes, Ruben Baumer, and Jesse Davis. «Semi-Supervised Anomaly Detection with an Application to Water Analytics». In: *2018 IEEE International Conference on Data Mining (ICDM)*. 2018, pp. 527–536. DOI: 10.1109/ICDM.2018.00068 (cit. on pp. 1, 10, 11).

[4] Jonas Soenen, Elia Van Wolputte, Lorenzo Perini, Vincent Vercruyssen, Wannes Meert, Jesse Davis, and Hendrik Blockeel. «The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods». In: *Proceedings of the KDD*. Vol. 21. 2021, pp. 1–9 (cit. on pp. 1, 32).

[5] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. «A survey of network anomaly detection techniques». In: *Journal of Network and Computer Applications* 60 (2016), pp. 19–31 (cit. on p. 1).

[6] Thomas T. Lu Nadipuram R. Prasad Salvador Almanza-Garcia. «Anomaly Detection». In: *Computers, Materials & Continua* 14.1 (2009), pp. 1–22. ISSN: 1546-2226 (cit. on p. 1).

[7] Ayad Mohammed Jabbar. «Local and Global Outlier Detection Algorithms in Unsupervised Approach: A Review.» In: *Iraqi Journal for Electrical & Electronic Engineering* 17.1 (2021) (cit. on p. 1).

[8] Robert Gwadera, Mikhail J Atallah, and Wojciech Szpankowski. «Reliable detection of episodes in event sequences». In: *Knowledge and Information Systems* 7.4 (2005), pp. 415–437 (cit. on p. 1).

[9] Sushmito Ghosh and Douglas L Reilly. «Credit card fraud detection with a neural-network». In: *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*. Vol. 3. IEEE. 1994, pp. 621–630 (cit. on p. 2).

[10] Tom Fawcett and Foster Provost. «Activity monitoring: Noticing interesting changes in behavior». In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. 1999, pp. 53–62 (cit. on p. 2).

[11] Zengyou He, Xiaofei Xu, and Shengchun Deng. «Discovering cluster-based local outliers». In: *Pattern recognition letters* 24.9-10 (2003), pp. 1641–1650 (cit. on p. 2).

[12] Jessica Lin, Eamonn Keogh, Ada Fu, and Helga Van Herle. «Approximations to magic: Finding unusual medical time series». In: *18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)*. IEEE. 2005, pp. 329–334 (cit. on p. 2).

[13] Eamonn Keogh, Jessica Lin, Sang-Hee Lee, and Helga Van Herle. «Finding the most unusual time series subsequence: algorithms and applications». In: *Knowledge and Information Systems* 11.1 (2007), pp. 1–27 (cit. on p. 2).

[14] Eamonn Keogh, Stefano Lonardi, and Bill'Yuan-chi' Chiu. «Finding surprising patterns in a time series database in linear time and space». In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002, pp. 550–556 (cit. on p. 2).

[15] Dennis Ulmer and Giovanni Cinà. «Know your limits: Uncertainty estimation with relu classifiers fails at reliable ood detection». In: *Uncertainty in Artificial Intelligence*. PMLR. 2021, pp. 1766–1776 (cit. on p. 2).

[16] Burr Settles. «Active Learning Literature Survey». In: 2009 (cit. on pp. 2, 12).

[17] David D. Lewis and William A. Gale. «A Sequential Algorithm for Training Text Classifiers». In: *SIGIR '94*. Ed. by Bruce W. Croft and C. J. van Rijsbergen. London: Springer London, 1994, pp. 3–12. ISBN: 978-1-4471-2099-5 (cit. on pp. 2, 12).

[18] Kilian Hendrickx, Lorenzo Perini, Dries Van der Plas, Wannes Meert, and Jesse Davis. «Machine Learning with a Reject Option: A survey». In: *arXiv preprint arXiv:2107.11277* (2021) (cit. on pp. 2, 13, 14).

[19] Lukasz Korycki, Alberto Cano, and Bartosz Krawczyk. «Active Learning with Abstaining Classifiers for Imbalanced Drifting Data Streams». In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 2334–2343. DOI: 10.1109/BigData47090.2019.9006453 (cit. on p. 2).

[20] Ricardo Gamelas Sousa, Ajalmar R Rocha Neto, Jaime S Cardoso, and Guilherme A Barreto. «Robust classification with reject option using the self-organizing map». In: *Neural Computing and Applications* 26.7 (2015), pp. 1603–1619 (cit. on p. 5).

[21] Lars Kai Hansen, Christian Liisberg, and Peter Salamon. «The error-reject tradeoff». In: *Open Systems & Information Dynamics* 4.2 (1997), pp. 159–184 (cit. on p. 5).

[22] Nima Hatami and Camelia Chira. «Classifiers with a reject option for early time-series classification». In: *2013 IEEE symposium on computational intelligence and ensemble learning (CIEL)*. IEEE. 2013, pp. 9–16 (cit. on pp. 5, 7).

[23] Christophe Denis and Mohamed Hebiri. «Consistency of plug-in confidence sets for classification in semi-supervised learning». In: *Journal of Nonparametric Statistics* 32.1 (2020), pp. 42–72 (cit. on p. 5).

[24] Wenming Jiang, Ying Zhao, and Zehan Wang. «Risk-Controlled Selective Prediction for Regression Deep Neural Network Models». In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8 (cit. on p. 5).

[25] Filipe Condessa, José Bioucas-Dias, Carlos A Castro, John A Ozolek, and Jelena Kovačević. «Classification with reject option using contextual information». In: *2013 IEEE 10th International Symposium on Biomedical Imaging*. IEEE. 2013, pp. 1340–1343 (cit. on p. 5).

[26] Filipe Condessa, José Bioucas-Dias, Carlos Castro, John Ozolek, and Jelena Kovačević. «Image classification with rejection using contextual information». In: *arXiv preprint arXiv:1509.01287* (2015) (cit. on p. 5).

[27] Filipe Condessa, José Bioucas-Dias, and Jelena Kovačević. «Robust hyperspectral image classification with rejection fields». In: *2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. IEEE. 2015, pp. 1–4 (cit. on p. 5).

[28] Filipe Condessa, José Bioucas-Dias, and Jelena Kovačević. «Supervised hyperspectral image classification with rejection». In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 9.6 (2016), pp. 2321–2332 (cit. on p. 5).

[29] Jing Lei. «Classification with confidence». In: *Biometrika* 101.4 (2014), pp. 755–769 (cit. on p. 5).

[30] Sarah Laroui, Xavier Descombes, Aurélia Vernay, Florent Villiers, François Villalba, and Eric Debreuve. «How to define a rejection class based on model learning?» In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 569–576 (cit. on p. 5).

[31] Yonatan Geifman and Ran El-Yaniv. «Selective classification for deep neural networks». In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 5).

[32] Jan Philip Göpfert, Barbara Hammer, and Heiko Wersing. «Mitigating concept drift via rejection». In: *International Conference on Artificial Neural Networks*. Springer. 2018, pp. 456–467 (cit. on p. 5).

[33] Luigi P Cordella, Claudio De Stefano, Francesco Fontanella, and A Scotto Di Freca. «Random forest for reliable pre-classification of handwritten characters». In: *2014 22nd International Conference on Pattern Recognition*. IEEE. 2014, pp. 1319–1324 (cit. on pp. 5, 6).

[34] Constantine Kotropoulos and Gonzalo R Arce. «Linear classifier with reject option for the detection of vocal fold paralysis and vocal fold edema». In: *EURASIP Journal on Advances in Signal Processing* 2009 (2009), pp. 1–13 (cit. on p. 6).

[35] Lydia Fischer, Barbara Hammer, and Heiko Wersing. «Efficient rejection strategies for prototype-based classification». In: *Neurocomputing* 169 (2015), pp. 334–342 (cit. on p. 6).

[36] En-hui Zheng, Chao Zou, Jian Sun, and Le Chen. «Cost-sensitive SVM with error cost and class-dependent reject cost». In: *International Journal of Computer Theory and Engineering* 3.1 (2011), p. 130 (cit. on p. 6).

[37] Aditya Vailaya and Anil Jain. «Reject option for vq-based bayesian classification». In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. Vol. 2. IEEE. 2000, pp. 48–51 (cit. on p. 6).

[38] Ricardo Sousa, Beatriz Mora, and Jaime S Cardoso. «An ordinal data method for the classification with reject option». In: *2009 International Conference on Machine Learning and Applications*. IEEE. 2009, pp. 746–750 (cit. on p. 6).

[39] Hongjiao Guan, Yingtao Zhang, Heng-Da Cheng, and Xianglong Tang. «Bounded-abstaining classification for breast tumors in imbalanced ultrasound images». In: *International Journal of Applied Mathematics and Computer Science* 30.2 (2020) (cit. on p. 6).

[40] Fabien Lotte, Harold Mouchere, and Anatole Lécuyer. «Pattern rejection strategies for the design of self-paced eeg-based brain-computer interfaces». In: *2008 19th International Conference on Pattern Recognition*. IEEE. 2008, pp. 1–5 (cit. on p. 6).

[41] Filipe Condessa, José Bioucas-Dias, and Jelena Kovačević. «Performance measures for classification systems with rejection». In: *Pattern Recognition* 63 (2017), pp. 437–450 (cit. on pp. 6, 23).

[42] Aaron Lewicke, Edward Sazonov, Michael J Corwin, Michael Neuman, and Stephanie Schuckers. «Sleep versus wake classification from heart rate variability using computational intelligence: consideration of rejection in classification models». In: *IEEE Transactions on Biomedical Engineering* 55.1 (2007), pp. 108–118 (cit. on p. 6).

[43] Claudio De Stefano, Carlo Sansone, and Mario Vento. «To reject or not to reject: that is the question-an answer in case of neural classifiers». In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.1 (2000), pp. 84–94 (cit. on p. 7).

[44] Chao Zou, En-hui Zheng, Hong-wei Xu, and Le Chen. «Cost-sensitive Multi-class SVM with reject option: a method for steam turbine generator fault diagnosis». In: *International Journal of Computer Theory and Engineering* 3.1 (2011), p. 77 (cit. on p. 7).

[45] Joaquim Arlandis, Juan Carlos Pérez-Cortes, and Javier Cano. «Rejection strategies and confidence measures for a k-nn classifier in an ocr task». In: *2002 International Conference on Pattern Recognition*. Vol. 1. IEEE. 2002, pp. 576–579 (cit. on p. 7).

[46] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. «Isolation Forest». In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17 (cit. on pp. 9, 32).

[47] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. «Efficient algorithms for mining outliers from large data sets». In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 427–438 (cit. on pp. 9, 32).

[48] Fabrizio Angiulli and Clara Pizzuti. «Fast outlier detection in high dimensional spaces». In: *European conference on principles of data mining and knowledge discovery*. Springer. 2002, pp. 15–27 (cit. on pp. 9, 32).

[49] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. «Optics-of: Identifying local outliers». In: *European conference on principles of data mining and knowledge discovery*. Springer. 1999, pp. 262–270 (cit. on pp. 10, 32).

[50] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. «LOF: identifying density-based local outliers». In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 93–104 (cit. on pp. 10, 32).

[51] Bernhard Scholkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. «Estimating the Support of a High-Dimensional Distribution». In: *Neural Computation* 13.7 (July 2001), pp. 1443–1471. ISSN: 0899-7667. DOI: 10.1162/089976601750264965. eprint: https://direct.mit.edu/neco/article-pdf/13/7/1443/814849/089976601750264965.pdf. URL: https://doi.org/10.1162/089976601750264965 (cit. on pp. 10, 32).

[52] Hans-Peter Kriegel, Peer Kroger, Erich Schubert, and Arthur Zimek. «Interpreting and Unifying Outlier Scores». In: Apr. 2011, pp. 13–24. DOI: 10.1137/1.9781611972818.2 (cit. on p. 11).

[53] Gerard Salton and Christopher Buckley. «Term-weighting approaches in automatic text retrieval». In: *Information processing & management* 24.5 (1988), pp. 513–523 (cit. on p. 26).