POLITECNICO DI TORINO

Master's Degree in Computer engineering

Master's Degree Thesis

Semi-supervised Tree-based Anomaly Detection

Supervisor	Candidate
Prof. Paolo GARZA	**Luca STRADIOTTI**

October 2022

Abstract

In many real-world applications, abnormal behaviors must be detected immediately to avoid dangerous situations. Several automated approaches have been proposed that aim to analyze the data collection provided and identify critical and dangerous patterns.

This task was always considered as unsupervised learning since no labeled instances were available: obtaining training labels is extremely expensive and requires a lot of time from experts who have to carefully read the data and provide the labels. However, nowadays there are often few labels available, so many semi-supervised models have been studied, which significantly improve the unsupervised performance.

Semi-supervised models are divided into three categories depending on their approach. One of them is composed of tree-based models that learn how to properly classify anomalies and normal data by building an ensemble of trees. Although these models are very powerful, they are poorly studied in the literature due to the difficulty of using both unlabeled and labeled information during the tree-construction phase.

Therefore, a novel semi-supervised tree-based approach is proposed in this work. The model learns from both the available labeled instances and unlabeled data to intelligently partition the space into regions to distinguish normal samples from outliers.

The model is then evaluated on several benchmark datasets and its performance is compared with available state-of-the-art algorithms. Empirically the obtained results show that the proposed approach outperforms the unsupervised and semi-supervised baselines for most of the datasets used.

Table of Contents

List of Tables

List of Figures

Acronyms

SSIF
Semi-Supervised Isolation Forest

kNNo
k-Nearest Neighbours outlier detection

OCSVM
One-Class Support Vector Machines

LOF
Local Outlier Factor

CBLOF
Cluster-Based Local Outlier Factor

IF
Isolation Forest

EIF
Extended Isolation Forest

INNE
Isolation-based anomaly detection using Nearest-Neighbor Ensembles

DeepSVDD
Deep Support Vector Data Description

SO-GAAL
Single-Objective Generative Adversarial Active Learning

MO-GAAL

Multiple-Objective Generative Adversarial Active Learning

SSDO

Semi-Supervised Detection of Outliers

SSkNNo

Semi-Supervised k-Nearest Neighbours outlier detection

SSAD

Semi-Supervised Anomaly Detection

DeepSAD

Deep Semi-Supervised Anomaly Detection

HIF

Hybrid Isolation Forest

HEX

Hybrid EXtended isolation forest

Chapter 1

INTRODUCTION

Anomaly or outlier detection is an important data mining technique that attempts to find patterns in data that differ significantly from normal behaviour. These patterns are usually referred to as anomalies or outliers and usually involve only a small portion of the dataset. Anomalous instances must be identified immediately, otherwise they pose major problems for a system that can cost a lot of money.

Anomaly detection applications include computer network intrusion detection[1], where sending sensitive information to an unauthorized destination can be detected thanks to anomalous traffic on the network, and financial transaction fraud detection[2], where anomalous transactions can indicate that a credit card or identity has been stolen. Anomaly detection can also be useful for performing predictive maintenance in manufacturing by detecting potential failures in advance[3], or for detecting potential problems in health data, where malignant tumors can be identified through anomalous MRI images[4]. The application domain is very important in anomaly detection because many contextual anomalies can be detected based on it[5]. How well an algorithm performs on a task depends on how well its assumptions match what is of interest from an application perspective[6].

Initially, the algorithms developed to detect anomalies were unsupervised because a huge amount of data is constantly generated in everyday life and labeling is costly, as analyzing the data and deciding whether the instances are normal or anomalous requires a significant amount of effort from domain experts. These models attempt to distinguish between anomalies and normal instances by exploiting only the inherent structure of unlabeled data, and thus it is often too difficult to deduce how a test instance can be correctly classified using only this limited knowledge.

Nowadays, few labeled instances are available in many applications, so several semi-supervised algorithms have been proposed that significantly outperform unsupervised methods. These algorithms are usually derived from existing unsupervised models, which are extended by adding a suitable bias to derive knowledge from the available labels.

The basic intuition of many anomaly detection algorithms is that anomalous instances are significantly different from normal ones and therefore easier to isolate[7]. These models rely on a tree ensemble to properly classify normal and anomalous instances: this bagging strategy reduces the variance[8] and increases the robustness of the model[9]. Although tree-based approaches perform well in both unsupervised and semi-supervised settings, they are poorly studied in the literature, and no model exploits labeled instances during the tree construction phase: all the proposed models make use of the labeled instances only to improve the scoring stage.

Therefore, in this thesis, a novel tree-based semi-supervised anomaly detection algorithm is proposed which addresses the limited availability of labels. The method attempts to intelligently partition space into regions using both unlabeled data and provided labeled instances. It recursively builds an ensemble of trees by selecting both the feature and the split value for the following split sampling from highly informative probability distributions, using both sources of information.

The proposed algorithm is then evaluated on several benchmark datasets and its performance is compared with state-of-the-art anomaly detection models. Empirically, the model achieves good performance even when only a few labels are available, and it outperforms the competing methods on most of the datasets used. Moreover, the contribution of unlabeled data can be used even when labels are not available, and experiments show that this often improves the unsupervised baseline considered.

Chapter 2

RELATED WORK

This chapter discusses the most closely related work in both an unsupervised and a semi-supervised setting. Originally, anomaly detection was based on unsupervised models, since no labels were available. Then, thanks to a limited amount of labeled instances, several semi-supervised models were proposed, which significantly improved the performance of the unsupervised models by using labeled instances. The basic assumption of this work is that a limited amount of labels is available. Thus, unsupervised anomaly detection models can still be competitive. The main algorithms are briefly analyzed in Section 2.1. Moreover, a few semi-supervised approaches that deal with labeled and unlabeled data have been proposed in the literature. The state-of-the-art methods for each category are described in Section 2.2.

2.1 Unsupervised algorithms

Obtaining labeled training data for anomaly detection is expensive. For this reason, the first anomaly detection algorithms developed were unsupervised. These algorithms can be divided into four categories:

- Distance-based algorithms
- Density-based algorithms
- Isolation-based algorithms
- Network-based techniques

2.1.1 Distance-based algorithms

These models are based on the assumption that normal instances are usually very close to each other, while outliers are very far from the other data points. For this

reason, they calculate an anomaly score for each instance based on the distance to its neighbors.

k-Nearest Neighbors Outlier detection(kNNo)

kNNo[10] was one of the first algorithms developed and is one of the easiest to understand. It stores all available instances and then classifies the new instances based on a distance metric. Usually, Euclidean distance is used for continuous data, while Hamming distance is used in the discrete domain, so it can handle non-standard data types.

The anomaly score of an instance is obtained by calculating the sum of the distances to the nearest k neighbors of the point. Based on the above intuition, normal instances have many points close to each other, so their anomaly score is small, while anomalies are located in a sparsely populated region, so their anomaly score is high.

Since kNNo needs to store all available training data and compute many distances for each test instance, it has large storage requirements and high computational complexity. Another drawback is that this algorithm is very sensitive to the choice of the distance metric and the hyperparameter k.

One-Class Support Vector Machines (OCSVM)

OCSVM[11] is a variation of the SVM that is used for unsupervised anomaly detection. The goal of this model is to learn in an unsupervised setting the boundaries of the normal-data region and all the points that lie outside this region are classified as anomalies.

Also for this variation, kernel extensions can be used.

2.1.2 Density-based algorithms

These algorithms calculate the anomaly score for each instance based on the density of its neighborhood. The basic assumption of these models is that outliers are usually far from their neighbors and therefore their neighborhood is sparse, while normal instances are concentrated in a dense region.

Local Outlier Factor (LOF)

The Local Outlier Factor[12] algorithm is based on the concept of local density, which is estimated using the distance to the k nearest instances. For each instance, it compares the local density of its neighborhood with that of its neighbors to find instances that have significantly lower density compared to their neighbors. These instances are considered outliers.

Thanks to its approach, this model is well suited for detecting local anomalies: for example, an instance with a "small" distance from a very dense cluster is classified as an outlier, while an instance within a sparse cluster could be considered as normal since it has a similar density to its neighbors.

The problem for LOF is that there is no clear rule by which an instance is classified as an anomaly. The threshold for splitting the anomaly scores of normal and outlier instances depends heavily on the dataset.

Clustering-Based Local Outlier Factor (CBLOF)

The CBLOF[13] operator calculates the outlier score based on cluster-based local outlier factor.

CBLOF generates a cluster model of the data set using a clustering algorithm and it classifies the clusters into small and large clusters depending on the parameters alpha and beta. Then the anomaly score is computed for each instance based on the size of the cluster the points belong to as well as the distance to the nearest large cluster.

Using this weighting for outlier factor based on the sizes of the clusters might lead to unexpected behaviour: outliers close to small clusters are usually not found. Therefore it is usually disabled and anomaly scores are solely computed based on the distance to the nearest large cluster center.

The original paper proposes to use the Squeezer algorithm to cluster, but usually kMeans is used by default because it provides better performances.

2.1.3 Isolation-based algorithms

The basic intuition of these models is that anomalous instances are clearly different from normal instances and therefore easier to isolate.

Isolation Forest (IF)

The Isolation Forest[14] isolates instances by randomly selecting a feature and then randomly determining a split value between the maximum and minimum values of the selected feature.

Since recursive partitioning is represented by a tree structure, the number of splits required to isolate an instance is equal to the path length from the root to the external node. This (average) path length is used to determine the anomaly score of a test instance. Anomaly pathways are substantially shorter than normal ones. Thus, if a forest of random trees yields shorter path lengths for certain instances, it is more likely that they are anomalies.

This algorithm is discussed in detail in the next chapter.

Extended Isolation Forest (EIF)

This algorithm is very similar to the Isolation Forest presented above. The only difference is that during the tree construction phase, the EIF [15] algorithm splits the space using cuts that are not constrained to be parallel to the axes of the feature space.

Isolation-based anomaly detection using Nearest-Neighbor Ensembles (INNE)

The nearest neighbour ensemble is used by the INNE algorithm[16] to isolate anomalies. It uses a subsample of the data to partitions the data space into regions and then it determines an isolation score for each region. Each region adapts to local distribution, therefore the calculated isolation scores are local measures that are relative to the local neighbourhood. In this way both global and local anomalies can be detected.

INNE has linear time complexity to efficiently handle large and high-dimensional datasets with complex distributions.

2.1.4 Network-based techniques

These techniques identify the anomalous instances thanks to the optimization of a loss function, usually using a deep neural network in a purely unsupervised way.

The problem with these models is that they are blackbox, therefore it is very difficult to understand their decisions.

Deep Support Vector Data Description (DeepSVDD)

Deep Support Vector Data Description[17] tries to learn useful feature representations of the data together with the one-class classification objective.

This model trains a neural network while reducing the volume of a hypersphere that encloses the network's data representations. This algorithm computes anomaly scores based on the distance of a given test instance to the center.

Single-Objective Generative Adversarial Active Learning (SO-GAAL)

Single-Objective Generative Adversarial Active Learning[18] tries to generate informative potential outliers to help the classifier to learn a boundary that can properly separate outliers from normal data.

The generator can fall into the mode collapsing problem because of the network structure of the SO-GAAL.

Multiple-Objective Generative Adversarial Active Learning (MO-GAAL)

Multiple-Objective Generative Adversarial Active Learning[18] tries to prevent the mode collapsing problem of the SO-GAAL by expanding the network structure of SO-GAAL from a single generator to multiple generators with different objectives. In this way, a reasonable reference distribution for the whole dataset should be generated.

2.2 Semi-supervised algorithms

Due to the limitations and expensiveness of labeled training data, few semi-supervised algorithms for anomaly detection exist, even though they significantly improve upon unsupervised ones.

Semi-supervised algorithms can be divided into three main categories:

- Label propagation-based algorithms
- Loss-based algorithms
- Tree-based algorithms

2.2.1 Label propagation-based algorithms

The algorithms belonging to this category use the available labels through a label propagation phase, whose goal is to update the anomaly scores assigned by a prior unsupervised anomaly detection algorithm. The basic assumption of these models is that the label of an instance is closely related to the available labels in its neighborhood: for example, an instance is more likely to be an outlier if it is close to an anomaly label. Therefore, the goal of the label propagation phase is to adjust the previously calculated anomaly score of each instance based on its distance from the available normal and anomaly labels.

These algorithms do not include a training phase where the model can learn the hidden patterns in the data by using the labeled and unlabeled instances. They simply start from a previous unsupervised prior and propagate the labels. This drawback degrades the generalization of these models.

Semi-Supervised Detection of Outliers (SSDO)

SSDO applies a two-step process to determine the anomaly score of each instance.[19]

First, constraint-based clustering is performed, and each instance is given an initial score based on its position with respect to the data distribution found. In this step, the anomaly score can be derived in a purely unsupervised way, but if labels

are available, they can impose some constraints on the data clustering: a cannot-link constraint is imposed between each anomalous and normal instance. Must-link constraints are not considered because there are different categories of normal and anomalous behavior. SSDO then assumes that outliers are "different" from normal instances, and the following basic intuitions are derived: (i) an anomalous instance deviates from its cluster centroid more than a normal one, (ii) the centroid of an anomaly cluster deviates strongly from the other cluster centroids, and (iii) an anomaly cluster is usually small. Based on the above intuitions, an anomaly score is then calculated for each instance x_k.

$$clustering\_score(x_k) = 1 - g(\frac{\text{point\_dev } (x_k) \times \text{cluster\_dev}(x_k)}{\text{cluster\_size}(x_k)}; \gamma) \tag{2.1}$$

where *point_dev* measures how much an instance deviates from its cluster centroid, *cluster_dev* is the cluster centroid difference with respect to the other cluster centroids, *cluster_size* indicates the relative size of the cluster to which the instance belongs, and g represents an exponential squashing function with γ depending on the contamination factor requested by the user.

Second, the available labeled instances are exploited in a label propagation phase. In this step, the previously calculated anomaly scores are updated based on the available labels.

$$score(x_k) = \frac{1}{Z(x_k)} \left[\text{clustering\_score}(x_k) + \alpha \sum_{x_j \in L_a} g\left(d\left(x_k, x_j\right); \eta\right) \right] \tag{2.2}$$

where L_a represents the set of labeled outliers, while η is the k-distance. The updates in the label propagation phase depend on the distance to the labeled instances. $Z(x_k)$ represents a normalization to map the final score between 0 and 1, while α is a hyperparameter that controls the weight of the label propagation phase: a low alpha value means that constraint-based clustering has a stronger influence than the label propagation phase in determining the final score.

SSDO also uses an active learning strategy in which a domain expert is asked for the labels of the instances for which the model is most uncertain.

Semi-Supervised k-Nearest Neighbors for Outlier detection method (SSkNNo)

SSkNNo[20] first computes an anomaly score for each instance in a purely unsupervised manner using the local data distribution. Then, this score is updated through a label propagation phase depending on the labeled instances in the neighborhood. The final anomaly score can be computed using the following formula:

$$score\left(x_k\right) = \left(1 - W_l\left(x_k\right)\right) a_u\left(x_k\right) + W_l\left(x_k\right) a_l(x_k) \tag{2.3}$$

where a_u represents the unlabeled component using the kNNo algorithm, and a_l is the labeled component based on the distance-weighted average of the labeled instances in the neighborhood of x_k. W_l is a factor that weights the two components based on the number of labeled and unlabeled neighbors of x_k that also include the instace in their neighborhood.

2.2.2 Loss-based algorithms

The algorithms that belong to this category attempt to optimize a loss function that uses both unlabeled and labeled instances. Usually, these algorithms are based on SVM or deep neural networks, as they are the best models to optimize a given loss function.

One of their main drawbacks is that they are usually very slow. Moreover, deep learning-based algorithms are black-box, while explainable decisions are preferred in anomaly detection, since understanding the model decisions in anomaly detection is crucial to increase end-user confidence and trust in model predictions.[21][22]

Semi-Supervised Anomaly Detection (SSAD)

This algorithm generalizes the unsupervised One-Class SVM[23] by using both labeled and unlabeled instances. SSAD[24] constructs a hypersphere like the OCSVM model but incorporates labeled instances by imposing two constraints: (i) a normal labeled instance must be located inside the sphere, while (ii) anomaly instances must be placed outside the hypersphere. Thus, given n unlabeled instances $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$ and m labeled instances $(\boldsymbol{x}_1^*, y_1^*), \ldots, (\boldsymbol{x}_m^*, y_m^*) \in \mathcal{X} \times \mathcal{Y}$ with $Y = \{+1, -1\}$, the optimization problem becomes:

$$
\begin{aligned}
&\min_{R,\gamma,\boldsymbol{c},\boldsymbol{\xi}} R^2 - \kappa\gamma + \eta_u \sum_{i=1}^{n} \xi_i + \eta_l \sum_{j=n+1}^{n+m} \xi_j^* \\
&\text{s.t. } \forall_{i=1}^{n}: \quad \left\|\phi\left(\boldsymbol{x}_i\right) - \boldsymbol{c}\right\|^2 \leq R^2 + \xi_i \\
&\qquad \forall_{j=n+1}^{n+m}: y_j^* \left(\left\|\phi\left(\boldsymbol{x}_j^*\right) - \boldsymbol{c}\right\|^2 - R^2\right) \leq -\gamma + \xi_j^* \\
&\qquad \forall_{i=1}^{n}: \quad \xi_i \geq 0, \\
&\qquad \forall_{j=n+1}^{n+m}: \xi_j^* \geq 0,
\end{aligned}
\tag{2.4}
$$

where c and R represent the center and radius of the hypersphere, respectively, γ controls the margin of labeled instances, and k, η_u, and η_l are trade-off parameters. The loss function is then transformed into an unconstrained optimization problem by some mathematical manipulations to facilitate the implementation.

The anomaly score of an instance x_k is calculated based on its distance from

the center c of the hypersphere.

$$score(\boldsymbol{x_k}) = \|\phi(\boldsymbol{x_k}) - \boldsymbol{c}\|^2 - R^2 \tag{2.5}$$

Deep Semi-supervised Anomaly Detection (DeepSAD)

DeepSAD[25] extends the unsupervised DeepSVDD models to use labeled instances as well. The final objective of this model consists of an unlabeled component corresponding to the original DeepSVDD loss and a labeled component that imposes that normal instances must be close to the hypersphere center, while anomalies must be far from it.

The anomaly scores assigned by the DeepSAD model are calculated based on the distance of each instance from the center.

2.2.3 Tree-based algorithms

These models build a tree ensemble that intelligently divides space into regions using both unlabeled and labeled instances. The basic assumption of these models is that the feature values of anomalous instances are very different from those of normal ones, making them easier to isolate. Therefore, the expected path length of outliers in the tree ensemble is shorter than that of normal instances. This intuition forms the basis for the scoring function of these models.

Hybrid Isolation Forest (HIF)

The HIF algorithm[7] assumes that the given dataset consists only of normal instances. The forest construction phase corresponds to the IForest algorithm.

The basic contribution of the HIF algorithm is an improvement of the original IForest anomaly score calculation by exploiting unlabeled and labeled instances.

The unlabeled contribution is based on the distance to neighboring normal instances. First, the centroid of data is calculated for each leaf of each iTree, and then for each instance the distance $\delta(x_k)$ to the centroid of the leaf on which it ends up is calculated. The final score is calculated as the expectation of the $\delta(x_k)$ scores given by the iTrees in the forest.

$$s_u(x_k) = \mathbb{E}(\delta(x_k))) \tag{2.6}$$

Then, if some anomaly instances are known, they are added to the corresponding leaves of the iTrees at the end of the training phase. For each leaf, the centroid of the anomaly instances is also calculated. At this point, the final labeled contribution is given by the ratio between the previously calculated expected value of $\delta(x_k)$ and the expected value of $\delta_a(x_k)$, which is the expectation over all the iTrees of the

distances between the instance x_k and the anomaly centroid of the leaf to which x_k belongs.

$$s_l(x) = \frac{\mathbb{E}(\delta(x_k))}{\mathbb{E}\left(\delta_a(x_k)\right)} = \frac{\text{Mean}_{\text{ iTree}}\, \delta(x_k,\ \text{iTree }) }{\text{Mean}_{\text{ iTree}}\, \delta_a(x,\ \text{iTree }) } \tag{2.7}$$

At this point, the standard IForest anomaly score $s(x_k)$ and the unlabeled and labeled contribution scores are normalized to the range [0,1]. The final score is calculated using the following linear formula.

$$score(x_k) = \alpha_2 \cdot (\alpha_1 \cdot s(x_k) + (1 - \alpha_1) \cdot s_u(x_k)) + (1 - \alpha_2) \cdot s_l(x_k)) \tag{2.8}$$

α_1 and α_2 are two hyperparameters for weighting the three score contributions.

Hybrid EXtended isolation forest (HEX)

Hybrid Extended isolation forest[26] tries to solve both the fact that original Isolation Forest[14] struggles to detect anomalies encircled or overlapped by regular points and the issue of ghost clusters or erroneous scores in score maps due to the horizontal and vertical cut hyperplanes.

Therefore, this algorithm combines the basic ideas of the EIF[15] and HIF[7] algorithm: the space is recursively partitioned using cuts that are not constrained to be parallel to the axes of the feature space and a distance-based score based on the available normal and anomaly labels is used to obtain the final anomaly score for each instance.

Chapter 3

BACKGROUND : ISOLATION FOREST ALGORITHM

In this chapter, a brief overview of the IForest algorithm is given. IForest is described because the model proposed in this work is inspired by it and changes the perspective to develop a model that improves the unsupervised construction of the tree ensemble taking advantage of the available labeled instances.

IForest[14] is one of the oldest algorithms for unsupervised anomaly detection. It builds an ensemble of isolation trees based on the fact that anomalies represent a minority in the dataset and their attribute values are significantly different from those of normal instances. For this reason, outliers are much easier to isolate from the rest of the data compared to normal instances, so the expected path in the tree ensemble to locate an anomaly is shorter than to locate a normal instance. The IForest scoring function assigns an anomaly score to each test instance that is inversely proportional to its expected path length.

3.1 Tree construction

The IForest algorithm recursively builds an ensemble of isolation trees dividing the space using only unlabeled data by randomly choosing the feature and the split value for the following split.

Given a dataset $D = \{x_1, ..., x_n\}$ of n instances each described by m features, each isolation tree is recursively built by splitting D by choosing each time the following feature and split value:

- the attribute X_q for the following split is sampled from a discrete uniform

distribution along the attributes, each attribute has the same probability.

- the split value c is sampled from a uniform distribution among the maximum and the minimum value of the selected feature X_q.

The tree construction phase stops when either the tree reaches a height limit, D contains a single instance or all the instances in D have the same values. Each node has exactly zero (external node) or two daughter nodes (internal node): iTrees have an equivalent structure to Binary Search Tree.

The implementation can be found in Algorithm 1

Algorithm 1 iTree(D, e, l)

Inputs: D - input data, e - current tree height, l - height limit
Output: an iTree

if $e \geq l$ **or** $|D| \leq 1$ **then**
  return $exNode\ \{\ D, e\ \}$
else
  let X be the set of features
  randomly select an attribute $X_q \in X$
  randomly select a split value c in ($min(D[X_q],\ max(D[X_q])$)
  $D_l \leftarrow filter(D, c, X_q, <)$
  $D_r \leftarrow filter(D, c, X_q, \geq)$
  return $inNode\ \{\ Left \leftarrow iTree(D_l, e+1, l),$
    $Right \leftarrow iTree(D_r, e+1, l),$
    $SplitAtt \leftarrow X_q,$
    $SplitValue \leftarrow c\ \}$
end if

3.2 Anomaly score computation

The anomaly score for an instance is calculated using its expected path length $\mathbb{E}(h(x))$ in the iTree ensemble. This value is then normalized using the average path length of unsuccessful search in Binary Search Trees[27], which for a dataset of n instances corresponds to:

$$c(n) = 2H(n-1) - (2(n-1)/n) \tag{3.1}$$

where $H(i)$ represents the harmonic number that can be estimated by $ln(i) + 0.5772156649$ (Euler's constant).

The anomaly score of an instance is calculated as follows.

$$score(x_k) = 2^{-\frac{\mathbb{E}(h(x_k))}{c(n)}} \tag{3.2}$$

As can be seen from the above formula, the shorter the expected path length for an instance, the more likely it is to be anomalous.

3.3 Important hyper-parameters

One of the most important hyper-parameters of this model is the number of trees t in the forest. In the original algorithm, the forest consists of 100 trees, since path lengths usually converge well before this value.

Each tree is built using a sample of the dataset, obtained by randomly selecting instances without replacement. The basic idea is that it is easier to isolate outliers with a small sample size ψ, and therefore each tree is built with a sample of 256 instances.

Finally, the height limit of the tree depends on the sub-sampling size ψ and is set to:

$$l = \text{ceiling}\left(\log_2 \psi\right) \tag{3.3}$$

which corresponds approximately to the average tree height.

Chapter 4

METHODOLOGY

The task studied in this work can be formally defined as follows:

Given: an unlabeled dataset $D_U = \{x_1, \ldots, x_d\}$ and some labeled instances $D_L = \{x_{d+1}, y_{d+1}), \ldots, (x_n, y_n)\}$ where $y = 1$ refers to the positive (anomalous) class, and $y = -1$ to the negative (normal) class.

Do: develop a new semi-supervised tree-based model which divides the space using both the unlabeled dataset and the labeled instances.

The task has two main challenges. First, obtaining labels is costly, as it requires domain experts to analyze the data and provide the labels. In addition, even if partially labeled instances can be used, few labels are available and they are usually normal instances, as outliers are much rarer.

Second, there are no semi-supervised tree-based algorithms that learn to partition the space into regions using both labeled and unlabeled instances: all tree-based approaches available in the literature use the labeled instances only to improve the scoring stage, the trees are always built using unlabeled data only. It is very difficult to extract information to choose a smart feature and split value for the following split during the tree construction by combining the unlabeled and labeled instances.

Semi-Supervised Isolation Forest (SSIF) is a tree-based model that uses together the unlabeled dataset and the available labeled instances to partition the space into regions. The goal of this approach is to work properly even when the number of available labels is very limited.

SSIF first constructs an ensemble of isolation trees using sub-samples of the data and then calculates the anomaly score for an instance by evaluating the average height in the ensemble of the leaves where it ends up.

4.1 Tree construction

Each tree is built individually using a sub-sample of the input data. At each splitting step, the following feature and split value are selected using labeled and unlabeled instances in two different ways, depending on the available labels. The tree construction phase is terminated when a leaf contains only one instance, when a height limit is reached, or when a leaf is small enough and contains only anomaly-labeled instances: in this case, the splitting would increase the height in the tree for the outliers and decrease their anomaly scores.

A general high-level description can be found in Algorithm 2. The different contributions are explained in detail in the following sections.

Algorithm 2 iTree(D, Y, e, l)

Inputs: D - input data, Y - input label, e - current tree height, l - height limit
Output: an iTree

if end condition **then**
  return exNode{D,Y,e}
else
  if only normal labels are provided **then**
    X_q, c = onlyNormalLabelsContribution(D, Y)
  else
    X_q, c = withAnomalyOrNoLabelsContribution(D, Y)
  end if
  $D_l, Y_l \leftarrow filter(D, Y, c, X_q, <)$
  $D_r, Y_r \leftarrow filter(D, Y, c, X_q, \geq)$
  return $inNode$ { $Left \leftarrow iTree(D_l, Y_l, e+1, l)$,
    $Right \leftarrow iTree(D_r, Y_r, e+1, l)$,
    $SplitAtt \leftarrow X_q$,
    $SplitValue \leftarrow c$ }
end if

4.2 Splitting a feature with unlabeled and only normal labeled instances

Since the main purpose of a tree is to isolate anomalies as early as possible, a realistic scenario involves having unlabeled data and only normal labels while building the tree.

When only normal labels are available, SSIF attempts to extract information from a combination of unlabeled and labeled instances to select the following

feature and split value. The basic goal of all the tree-based approaches is to isolate outliers as quickly as possible, but this task is much more difficult in this scenario where only normal labels are available since anomaly instances can not be exploited to learn about anomalous patterns. The basic intuition of this contribution is therefore that it is more convenient to split outside the normal-labeled region, where anomaly instances are more likely to be isolated.

Feature selection. In this setting, the subset $\widetilde{X}$ of features for which the unlabeled and normal labeled maximum and minimum do not respectively correspond is first selected (see Figure 4.2). After this filtering, the feature X_q is uniformly selected from $\widetilde{X}$.

Split value selection. The split value c is then chosen outside the normal-labeled range of X_q.

$$c \in (min(D_U[X_q]), min(D_L[X_q])) \cup (max(D_L[X_q]), max(D_U[X_q])) \tag{4.1}$$

D_U and D_L represent the unlabeled and labeled subsets, respectively, of the data D at that point in the tree.

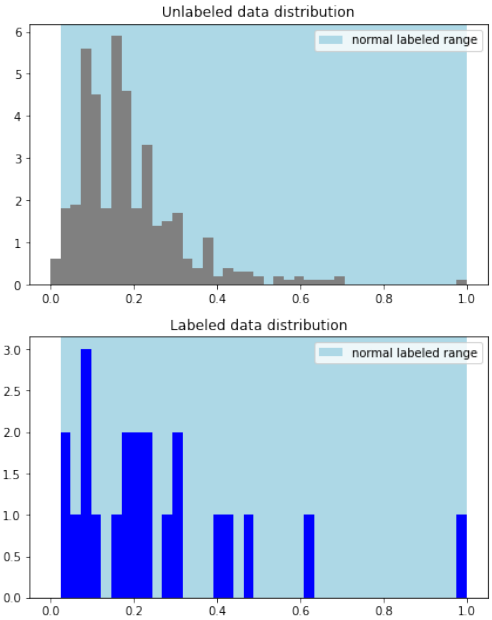

Figure 4.1: Feature with identical unlabeled and normal labeled ranges

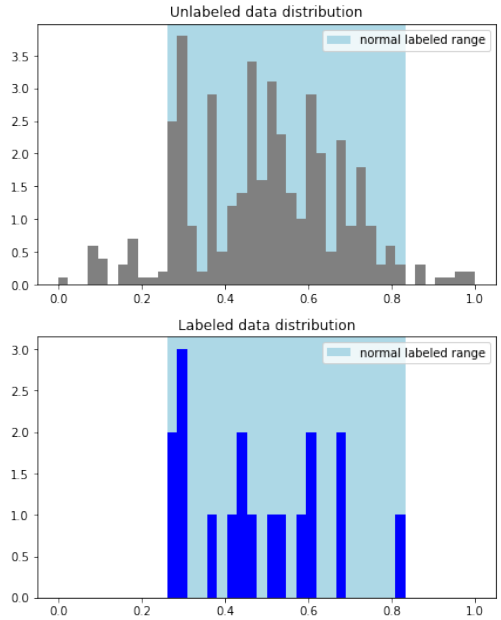

Figure 4.2: Feature with different unlabeled and normal labeled ranges

It should be noticed that a strong limitation of this approach is that it is blind toward potential future optimal splits. This means that it tries to isolate anomalies

as soon as possible and does not account for splits that create a better scenario for the next iteration.

A high-level description of the attribute and cut selection in this setting is provided in Algorithm 3.

Algorithm 3 onlyNormalLabelsContribution(D, Y)

Inputs: D - input data, Y - input label
Output: X_q - selected feature, c - selected split value

```
let X be the set of features
let X̃ be an empty list
for X_k in X do
  if max(unlabeled(X_k)) ≠ max(labeled(X_k)) or
      min(unlabeled(X_k)) ≠ min(labeled(X_k)) then
    X̃ ← append(X_k)
  end if
end for
select randomly an attribute X_q from X̃
select randomly a split value c outside the normal labeled range of X_q
return X_q, c
```

4.3 Splitting a feature with unlabeled and normal and anomaly labeled instances

Another possible scenario involves having unlabeled data and both anomaly and normal labels during the tree construction phase. In this case, when both anomaly and normal labels are available, the basic intuition of this model is to select a smart feature and a smart split value for the following split in the tree construction, extracting information from both unlabeled and labeled instances. Therefore, the main contribution in this context is to approximate the distributions for the selection of the following feature and split value to draw samples from highly informative distributions, thanks to which it is easier to isolate the anomalous samples.

Even if at some point in the tree construction labels are no longer available, the feature and split value are selected as described in this section. Obviously, in this case, the cut and feature selection distributions are determined only by the unlabeled instances, since no labels are available.

It is important to note that the cut distribution does not indicate the distribution of the data for the next split, but **the distribution of the cut from which a value is then selected as the next split value**. Let C be the cut variable, in

Figure 4.3 the difference between the cut distributions of the IForest and SSIF can be observed: IForest draws the next split value c from a uniform distribution between the minimum and maximum of the selected feature, while the SSIF model computes a more informative distribution that uses the labeled and unlabeled instances.

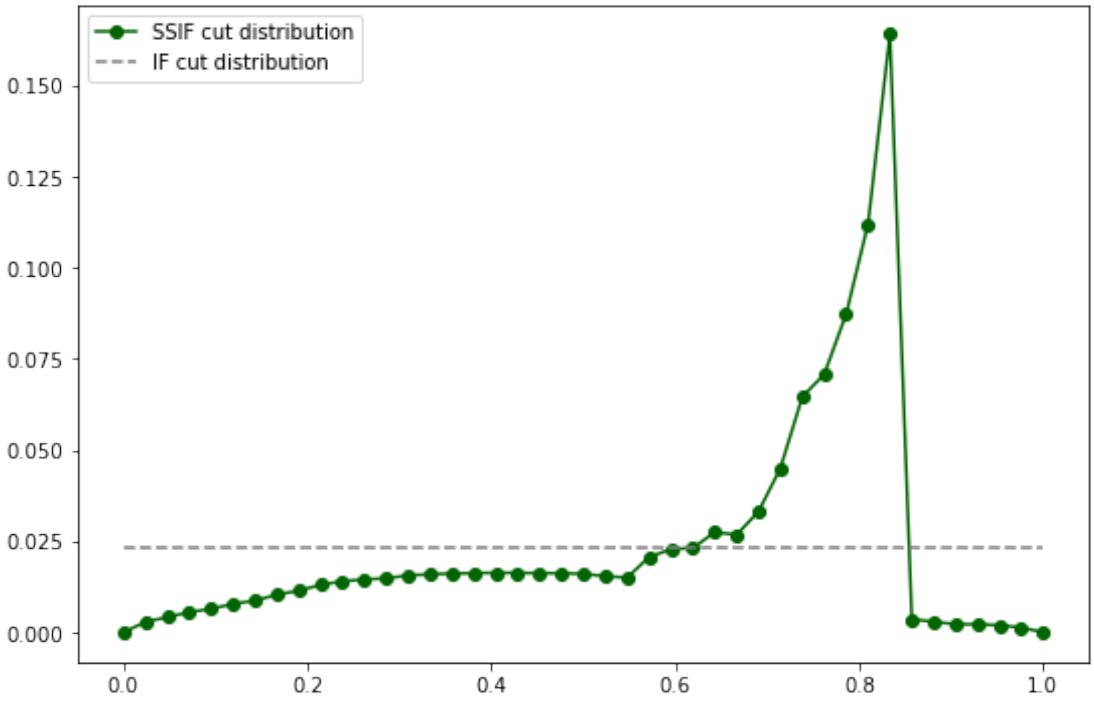

Figure 4.3: Cut distribution for IForest and SSIF. IForest uses a uniform distribution, while SSIF distribution is more informative: it can be observed that some split values are much more probable than others.

An initial high-level description of the attribute and split value selection in this setting is provided in Algorithm 4.

4.3.1 Cut distribution

Given the data distribution of a feature X_k, the cut distribution is estimated to properly select the following split value from a distribution that is as highly informative as possible. This distribution attempts to estimate the distribution of the cut variable C for the feature X_k under consideration, using both labeled and unlabeled instances.

Algorithm 4 withAnomalyOrNoLabelsContribution(D, Y)

Inputs: D - input data, Y - input label
Output: X_q - selected feature, c - selected split value

let $\widetilde{X}$ be a subset of the features
for $\widetilde{X}_k$ in $\widetilde{X}$ **do**
$P_C(\widetilde{X}_k) \leftarrow Cut\_distribution(D, Y)$
$P_{FeatureSelection}(\widetilde{X}_k) \leftarrow FeatureSelection\_distribution(P_C(\widetilde{X}_k))$
end for
select an attribute X_q given $P_{FeatureSelection}$
select a split value c given $P_C(X_q)$
$D_l, Y_l \leftarrow filter(D, Y, c, X_q, <)$
$D_r, Y_r \leftarrow filter(D, Y, c, X_q, \geq)$
return $inNode$ { $Left \leftarrow iTree(D_l, Y_l, e+1, l)$,
$Right \leftarrow iTree(D_r, Y_r, e+1, l)$,
$SplitAtt \leftarrow X_q$,
$SplitValue \leftarrow c$ }

Unlabeled component

The unlabeled component is calculated using the Otsu method, which considers only unlabeled data[28]. This method assumes that the two classes are distributed according to a bimodal distribution, and therefore searches exhaustively for the best threshold that minimizes the within-class variance, or equivalently maximizes the inter-class variance, defined as the weighted sum of the variances of the two classes.

So the unlabeled component is obtained by moving a threshold t between the minimum and maximum values for the feature X_k and estimating the inter-class variance for each threshold t.

$$\begin{aligned} unlabeled(t \mid X_k) &= \omega_0(t \mid X_k)\left(\mu_0 - \mu_T\right)^2 + \omega_1(t \mid X_k)\left(\mu_1 - \mu_T\right)^2 = \\ &= \omega_0(t \mid X_k)\omega_1(t \mid X_k)\left[\mu_0(t \mid X_k) - \mu_1(t \mid X_k)\right]^2 \end{aligned} \tag{4.2}$$

which is expressed in terms of class probabilities ω and class means μ, where the class means $\mu_0(t \mid X_k)$, $\mu_1(t \mid X_k)$ and μ_T are:

$$\mu_0(t \mid X_k) = \frac{\sum_{i=0}^{t-1} ip(i \mid X_k)}{\omega_0(t \mid X_k)} \tag{4.3}$$

$$\mu_1(t \mid X_k) = \frac{\sum_{i=t}^{L-1} ip(i \mid X_k)}{\omega_1(t \mid X_k)} \tag{4.4}$$

$$\mu_T = \sum_{i=0}^{L-1} ip(i \mid X_k) \tag{4.5}$$

and the probabilities $\omega_0(t \mid X_k)$ and $\omega_1(t \mid X_k)$:

$$\omega_0(t \mid X_k) = \sum_{i=0}^{t-1} p(i \mid X_k) \tag{4.6}$$

$$\omega_1(t \mid X_k) = \sum_{i=t}^{L-1} p(i \mid X_k) \tag{4.7}$$

where $p(i \mid X_k)$ represents the probability for the i-th bin in the histogram distribution of the data.

Figure 4.4 shows how the unlabeled component turns out given an example of unlabeled data distribution. It can be seen that the unlabeled component is different from the unlabeled data distribution: this component is a contribution that aims to approximate the distribution for the following split value using only unlabeled data.

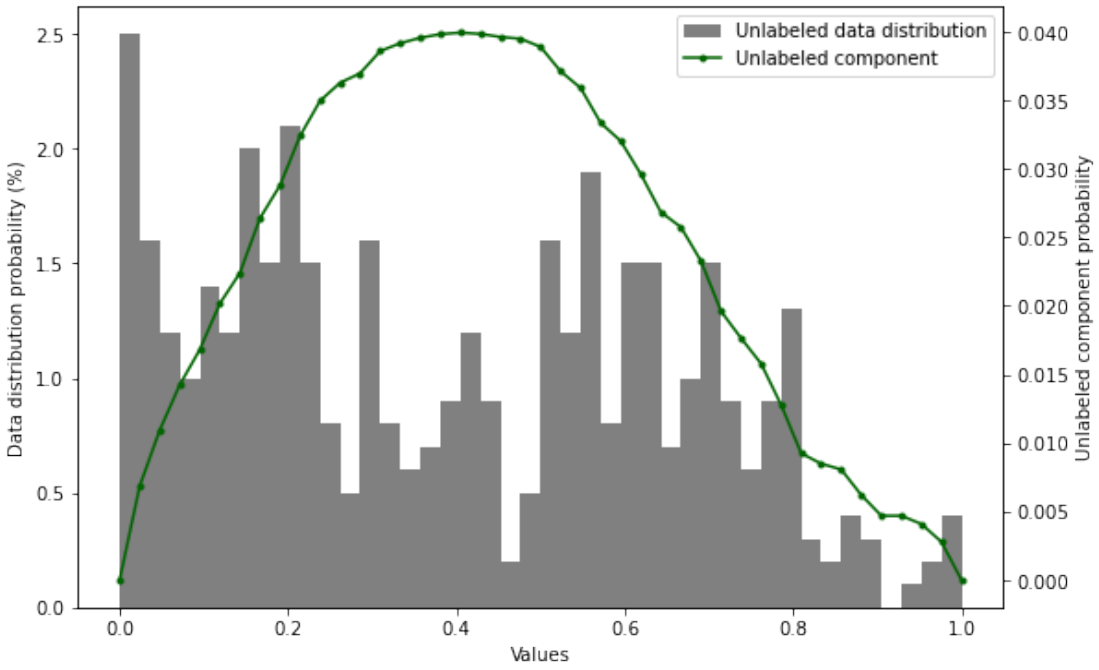

Figure 4.4: Unlabeled component given an example of unlabeled data distribution

Labeled component

The labeled component is an important contribution that uses the available labels to learn how to isolate the anomalous patterns as quickly as possible, which is the main goal of all the tree-based anomaly detection models. Therefore, this component modifies the information gain metric to meet this purpose.

Specifically, this component looks for split values that isolate anomalies: in this way, the expected path length of the isolated anomalies would decrease and consequently their anomaly score would increase. Another fundamental goal of this component is to avoid split values that isolate normal instances, since in this case the split would be harmful because it would decrease the expected path length of the isolated normal instances.

Thus, the labeled component is obtained by moving a threshold t between the minimum and maximum values for the feature X_k and estimating a score for each threshold depending on the available labels. For the reasons just explained, the model assigns a score of 0 if the threshold isolates instances with normal labels; however, if the isolated instances are anomalies, the threshold receives the highest possible score, which is equal to the initial entropy of the data distribution $h(D_L \mid X_k)$. If the threshold does not isolate a class, the assigned score is equal to

the resulting information gain after splitting at value t.

$$labeled(t \mid X_k) = \begin{cases} 0 \quad when \;\; normal \; instances \; are \; isolated \\ \mathcal{H}(D_L \mid X_k) \quad when \;\; anomaly \; instances \; are \; isolated \\ IG(D_L, t \mid X_k) \quad otherwise \end{cases} \tag{4.8}$$

where D_L represents the labeled instances along the previously selected attribute X_k, $\mathcal{H}$ represents the entropy value and:

$$\begin{aligned} IG(D_L, t \mid X_k) = \mathcal{H}(D_L \mid X_k) - \frac{\#(D_L \geq t \mid X_k)}{\#(D_L \mid X_k)} * \mathcal{H}(D_L \geq t \mid X_k) \\ - \frac{\#(D_L < t \mid X_k)}{\#(D_L \mid X_k)} * \mathcal{H}(D_L < t \mid X_k) \end{aligned} \tag{4.9}$$

Figure 4.5 shows how the labeled component turns out using an example of anomaly and normal data distribution.

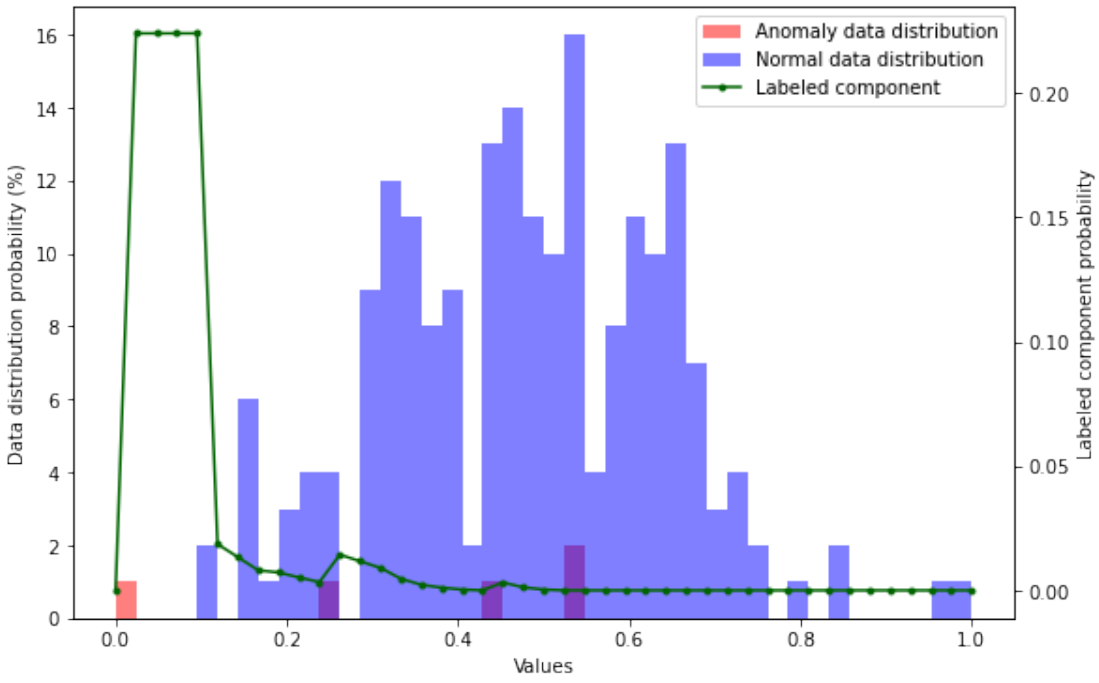

Figure 4.5: Labeled component given ax example of anomaly and normal data distribution

Split value selection

In the end, the final score for each split value can be calculated as the weighted sum of the two components. The labeled component is weighted more heavily to try to get as much information as possible from the available labels.

$$\begin{aligned} score_C(t \mid X_k) = 0.5 * (\frac{\#(unlabeled)}{\#(data)}) * unlabeled(t \mid X_k) + \\ 0.5 * (1 + \frac{\#(labeled)}{\#(data)}) * labeled(t \mid X_k) \end{aligned} \tag{4.10}$$

These scores are then normalized to obtain the probability distribution of the cut variable C to allow sampling from this.

$$P_C(t \mid X_k) = \frac{score_C(t \mid X_k)}{\sum_{t'} score_C(t' \mid X_k)} \tag{4.11}$$

Finally, the following split value c is sampled according to the cut distribution of the selected feature X_q.

4.3.2 Feature selection distribution

The goal of this distribution is to select a highly informative feature X_q for the following split. In this context, a feature is considered informative when its cut distribution $P_C(X_q)$ may allow the isolation of anomaly instances.

First, a subset $\widetilde{X}$ of the features for which the cut distribution has to be estimated is selected. Then, the informativeness of each feature is calculated based on its cut distribution.

To calculate this distribution, first a subset $\widetilde{X}$ of the features $X = \{X_1, X_2, .., X_m\}$ is selected for which the cut distribution has to be estimated. This sampling considers only the features for which the maximum and minimum of the unlabeled and normal labeled instances do not respectively correspond (see Figure 4.2), because for these features it is more likely that an anomaly outside the normal-labeled range is isolated. If the ranges match in all features, they are uniformly selected from the entire set. This sampling is done to reduce the correlation of the trees and lower the computational complexity of the model.

At this point, the cut distribution for each feature in $\widetilde{X}$ is computed. To measure how informative each attribute is, its cut distribution is compared to the uniform distribution. The informativeness of a feature is considered proportional to the difference between its cut distribution and the uniform distribution because if the former has a high peak, it is more likely that anomaly instances can be isolated using that feature. This comparison is made by calculating the Kullback-Leibler

divergence[29] between the two distributions.

$$D_{\mathrm{KL}}(P_C \mid\mid U \mid X_k) = \sum_t P_C(t \mid X_k) * \log\left(\frac{P_C(t \mid X_k)}{U(t)}\right) \tag{4.12}$$

The values obtained for each attribute are then normalized to obtain a probability distribution.

$$P_{FeatureSelection}(X_k) = \frac{D_{\mathrm{KL}}(P_C \mid\mid U \mid X_k)}{\sum_{X_k} D_{\mathrm{KL}}(P_C \mid\mid U \mid X_k)} \tag{4.13}$$

The attribute X_q for the next split is then sampled probabilistically according to this discrete distribution.

4.4 Feature and split value sampling

As explained above, the feature X_q and the split value c for the following split are drawn respectively from the feature-selection distribution and the cut one. To obtain samples from non-standard discrete distributions, their cumulatives are calculated. Then two samples u_q and u_c are drawn uniformly from [0,1]. The selected feature depends on the state of the feature-selection cumulative to which u_q belongs, while the chosen split value depends on the state of cut cumulative of u_c.[30]

4.4.1 An alternative for the split value selection

As explained above, the following split value c is taken from the cut distribution of the selected feature X_q. An alternative to this choice can be to select the threshold t at which the cut distribution of X_q reaches the maximum value, the mode of the distribution. When the mode is reached for different thresholds, one of them is chosen randomly.

$$c = argmax(P_C(X_q)) \tag{4.14}$$

This alternative option may be very appropriate for the training set since the mode corresponds to the threshold most likely to isolate outliers, given the way the distribution is computed, but it may lead to overfitting since the choice of mode depends heavily on the training set and may not generalize properly.

The mode selection is compared to the default probabilistic choice in the Experiments (Chapter 5).

4.5 Forest construction

The trees are then combined to build an ensemble. This bagging strategy, combining many weak classifiers, aims to reduce the variance of the model and improve its performance. In this way, anomalies can be better distinguished from normal instances because the expected path length for each instance converges properly by averaging the decision over multiple trees[14].

The construction of the forest is described in Algorithm 5.

Algorithm 5 iForest(D, Y, t, ψ)

Inputs: D - input data, Y - input label, t - number of tress, ψ - sub-sampling size
Output: a set of iTrees

```
let Forest be an empty list
compute height limit l
for i = 1...t do
    D', Y' ← sample(D, Y, ψ)
    Forest ← append(iTree(D', Y', 0, l))
end for
return Forest
```

4.6 Anomaly score computation

The anomaly scores are calculated as in the IForest algorithm[14]. They depend on the expectation of the path length $\mathbb{E}(h(x_k))$ of the test instance x_k on all trees in the ensemble.

$$score(x_k) = 2^{-\frac{\mathbb{E}(h(x_k))}{c(n)}} \tag{4.15}$$

where n represents the number of instances in the dataset and $c(n)$ is calculated as explained in 3.1.

4.7 Limitations

The model presented in this work has several limitations. First, SSIF has a high computational cost compared to the other state-of-the-art algorithms since it has to estimate the cut distribution for many features. Second, the model has a high number of hyper-parameters, like all the tree-based approaches. Third, SSIF applies a greedy strategy to select the next feature and split value, therefore the model is blind toward possible future optimal splits. Then, it should not work well when

the data are distributed according to a multimodal distribution since both the unlabeled component when both labels are available and the contribution with only normal labels assume a bimodal distribution.

Chapter 5

EXPERIMENTS

The experiments in this work answer the following four questions:

Q1: How does SSIF perform compared to IForest and semi-supervised anomaly detection algorithms?

Q2: How do the different SSIF hyperparameters affect its performance?

Q3: Which is the best method for selecting the split value: probabilistic or maximum selection?

Q4: Does SSIF improve unsupervised IForest even when no labels are provided?

5.1 Setup

5.1.1 Methods

In these experiments, one representative algorithm for each semi-supervised category outlined in Section 2.2 is compared with the SSIF model. IForest is also considered as an unsupervised baseline.

The following models are considered:

- **SSIF** is the method outlined in this work
- **SSDO** is a state-of-art semi-supervised anomaly detection algorithm based on a label-propagation phase
- **SSAD** is a state-of-the-art semi-supervised anomaly detection algorithm based on loss function optimization (one-class SVM)
- **HIF** is a state-of-the-art semi-supervised anomaly detection algorithm based on a tree ensemble built using only unlabeled data and a distance-based score using labeled instances
- **IF** is the considered unsupervised tree-based anomaly detection baseline

These algorithms were chosen based on extensive empirical assessments of anomaly detection methods (see also Chapter 2).

5.1.2 Datasets

The datasets considered are benchmarks for evaluating anomaly detection algorithms[31]. The datasets are downloaded from:

https://www.dbs.ifi.lmu.de/research/ outlier-evaluation/DAMI/

They are already preprocessed to delete duplicate instances and normalize in the [0,1] range their attributes. where contamination factor represents the percentage

Dataset name	**#Instances**	**#Features**	**contamination factor**
Cardiotocography	1861	21	0.0196
Waveform	3,443	21	0.0290
Shuttle	1,013	9	0.0128
SpamBase	2661	57	0.0500
WBC	223	9	0.0448
WDBC	367	30	0.02725
Annthyroid	6942	21	0.0500
Arrhythmia	271	259	0.0996
Stamps	340	9	0.0912

Table 5.1: Number of instances, features and contamination factor for each of the datasets used in the experiments.

of anomaly instances in the dataset.

Annthyroid[31]

This dataset provides medical information on hypothyroidism. Normal, excessive, and subnormal functioning are classified into three categories. Outliers in this context were defined as classes that deviate from normal conditions.

Arrhytmia[32]

The data collection includes patient data classified as normal or as a type of cardiac arrhythmia. Healthy persons are considered to be inliers and arrhythmia sufferers to be outliers.

Cardiotocography[32]

Dataset pertaining to cardiac ailments. It categorizes people into three groups: normal, questionable, and pathological. Normal patients are considered inliers, whereas the others are considered outliers.

Shuttle[31]

In this dataset, the radiator placements of a NASA space shuttle are described using nine attributes. The original instances were arranged in temporal order. Classes 1, 3, 4, 5, 6, and 7 are used as inliers and class 2 is used as an outlier

SpamBase[32]

This dataset contains several emails, considered as spam (anomalies) or non-spam.

Stamps[32]

The difference between counterfeit (photocopied or scanned+printed) stamps and genuine (ink printed) stamps is represented by this data set. The characteristics are based on the color and print quality of the stamps. Outliers are stamps that have been counterfeited.

Waveform[32]

This dataset contains three types of waves. Class 0 was defined as an outlier class, and in this case it was downsampled to 100 elements.

WBC[32]

This collection contains examples of many cancers, both benign and malignant. Cases of benign cancer are referred to as inliers, while examples of malignant cancer are referred to as outliers. Further preprocessing is used to eliminate missing data instances and standardize the data.

WDBC[32]

The nuclear features for breast cancer diagnosis are described in this data collection. Again, benign cancer examples are presented as inliers and malignant cancer ones are presented as outliers.

5.1.3 Experimental setup

The purpose of these experiments is to answer the previous research questions. To evaluate these, the following procedure is used for each dataset:

1. 5-fold cross-validation is used, iteratively using 4 folds for training and the remaining one for testing;

2. for each train-test split, 10 iterations are performed and each iteration uses a different seed (0,1, ...,9) to reduce random effects;

3. for each iteration, the labels in the training set are observed incrementally to evaluate how the models perform as more labeled information is provided. Initially, only 5% of labels is considered and then another 5% at a time is added until a fully labeled training set is reached.

4. for each percentage of labels the area under ROC curve (AUROC) is calculated on the test set for the considered models

5. AUROC performances are averaged to obtain a measure where random and bias effects are reduced

5.1.4 Evaluation metric

The AUROC is considered as evaluation metric as is standard in anomaly detection[32]. This metric is often chosen because anomaly detection algorithms usually return a ranking of the anomaly scores and so AUROC is suitable for measuring how the model is able to rank the instances. AUROC is also used because it takes into account the unbalanced class distributions of outliers and normal instances.

5.1.5 Hyperparameters

SSIF has a high number of hyperparameters, like all tree-based approaches. Unless otherwise stated, the default hyperparameters of SSIF are as follows:

- Number of trees t is set equal to 100 because path lengths typically converge well before this value[14].

- Sub-sampling size ψ is set equal to $\frac{len(D)}{3}$ to exploit enough labeled instances in each tree. The sampling is done without replacement.

- Number of features λ for which the cut distribution is calculated. This value is set to $max(\frac{len(X)}{3}$,5)

- Height limit l is set to $2 * ceiling(log_2\psi)$. A high value is chosen because the cut distribution tends to favor unbalanced trees: this happens because the labeled component gets the maximum gain when anomalies are isolated and they represent a small amount of data.

- Leaf size s is set to 5. A leaf is considered small enough to end the splitting phase if it contains a number of instances less than or equal to this value.

For the other state-of-the-art algorithms considered in this work, the default values of their hyperparameters are used in these experiments:

- **SSDO**[19] has three hyperparameters: the number of clusters n_c is set to 10, the parameter k that controls how many instances are updated for a single label is set to 30 and α that weights the label propagation phase is equal to 2,3.

- For **SSAD**[24] a Gaussian kernel is considered. Then the regularizer k for the importance of the margin and the regularization constants η_u and η_l are set to 1.

- The **HIF**[7] hyperparameters α_1 and α_2 for weighting the influence of the different components on the scoring function are set to 0,5.

5.2 Results

a) **Q1:** Figure 5.1 shows how the test set AUROC averaged over all the different seeds and folds varies depending on the percentage of labeled instances provided for SSIF and each baseline method. In general, all the semi-supervised baselines perform better when more labels are provided. As can be seen in Table 5.2, SSIF model significantly improves the unsupervised baseline. The main performance improvement is given by the first 20 percent of labels, which is desirable since only a few labels are usually provided for anomaly detection tasks. SSIF also always outperforms SSAD, regardless of the percentage of labels provided. In addition, SSIF outperforms all the baselines on six datasets (Cardio, Shuttle, Spambase, WDBC, Annthyroid, Stamps), as can be observed in Table 5.3.

b) **Q2:** In this question, 5 iterations are performed for each fold and only a subset of the previous datasets is analyzed because of a high computational cost.

The parameter ψ controls the number of instances used to create each tree. To assess the impact of this parameter, 5 different values of ψ are considered while keeping all the others hyperparameters fixed. Figure 5.2 shows the average

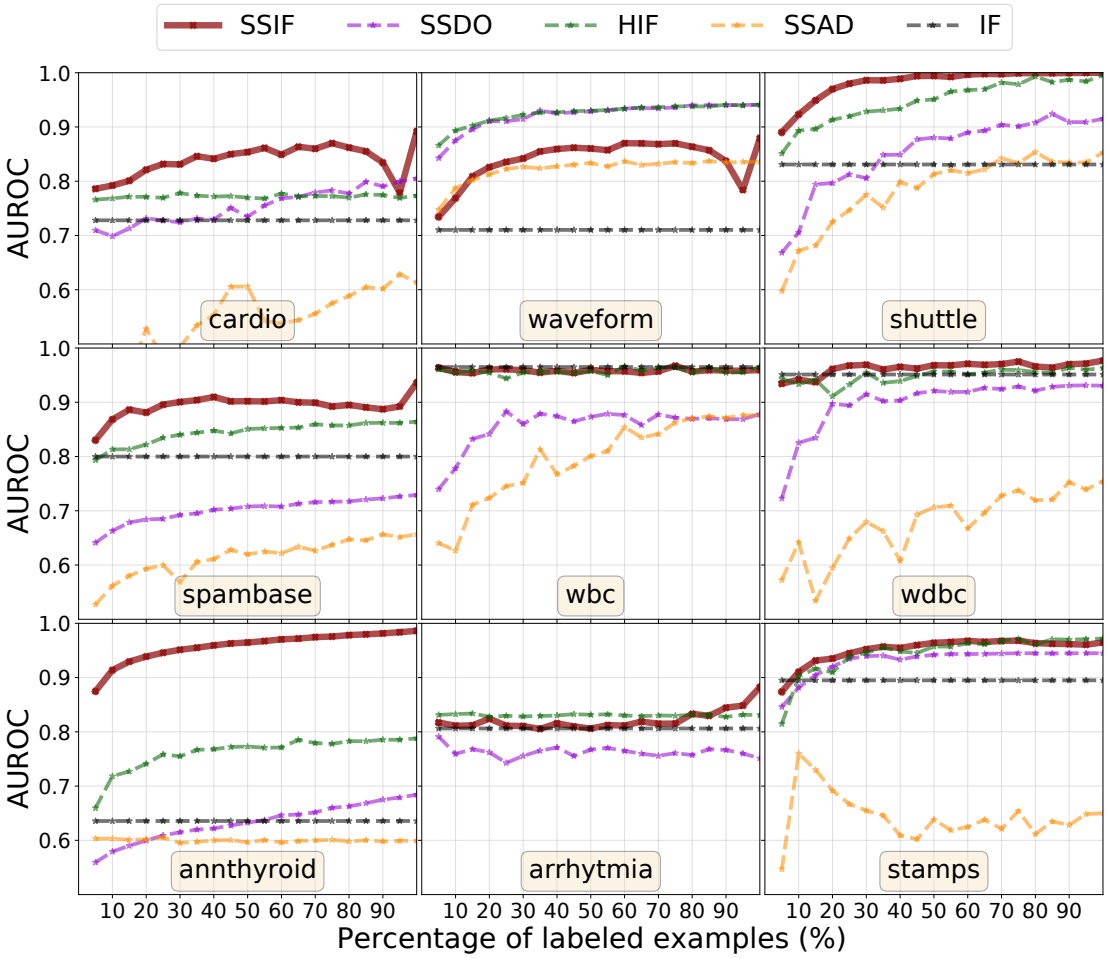

Figure 5.1: Each dataset's average test set AUROC as a function of the percentage of labeled instances provided for each method.

	SSIF's AUROC improvement compared to IF (%)
after 20% of labels	12,79
after 40% of labels	14,79
after 60% of labels	15,47
after 80% of labels	15,82
after 100% of labels	18,14

Table 5.2: Percentage test set AUROC improvement of the SSIF model compared to the unsupervised IF considering adding 20% of labels at a time. The results are averaged over all datasets, seeds, and folds.

Dataset	SSIF	SSDO	SSAD	HIF	IF
Cardio	**1.46 ± 0.79**	2.78 ± 1.13	4.86 ± 0.46	2.46 ± 0.75	3.44 ± 0.97
Waveform	3.44 ± 0.58	**1.53 ± 0.60**	3.69 ± 0.78	1.54 ± 0.51	4.80 ± 0.44
Shuttle	**1.24 ± 0.61**	3.41 ± 1.09	3.78 ± 1.03	2.12 ± 1.00	4.23 ± 0.90
Spambase	**1.15 ± 0.39**	4.18 ± 0.38	4.82 ± 0.40	2.04 ± 0.54	2.23 ± 1.01
WBC	2.46 ± 1.29	3.32 ± 1.55	3.55 ± 1.68	**2.13 ± 1.17**	2.23 ± 1.01
WDBC	**1.92 ± 1.26**	2.74 ± 1.47	1.94 ± 1.05	4.33 ± 1.18	2.67 ± 1.39
Annthyroid	**1.00 ± 0.00**	3.83 ± 0.78	4.35 ± 0.85	2.04 ± 0.25	2.65 ± 1.39
Arrhytmia	2.15 ± 1.11	3.22 ± 1.17	4.99 ± 0.11	**2.05 ± 0.97**	2.48 ± 0.92
Stamps	**2.11 ± 0.91**	2.31 ± 1.28	4.68 ± 0.92	2.17 ± 0.97	3.64 ± 0.88

Table 5.3: Average test set AUROC rank ± standard deviation of each method and dataset across all analyzed percentages of labels.

test set AUROC for the different values of ψ as function of the provided percentage of labels. It can be observed that the hyperparameter value has a strong impact on the performance and in general higher values of ψ provide better results, since in this way more labeled instances can be used in each tree, and increase the computational complexity of the model. However, the default value chosen represents a good trade-off between performance and computational cost.

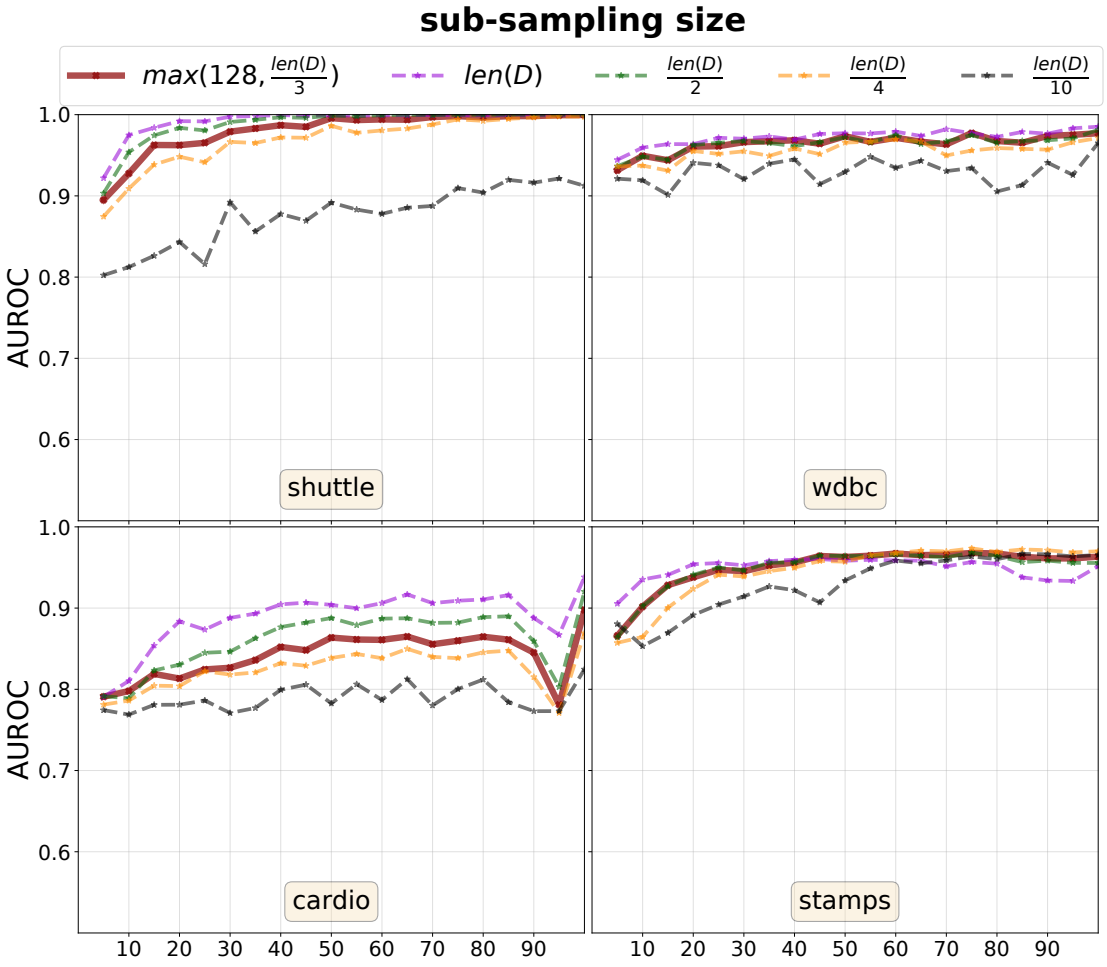

Figure 5.2: Each dataset's average test set AUROC as a function of the percentage of labeled instances provided for each value of ψ

The hyperparameter λ states for how many attributes the cut distribution is estimated. To assess its influence on SSIF performances, 5 different λ values are used while the other hyperparameters remain unchanged. Figure 5.3 shows the average test set AUROC for the different values of λ as function

of the percentage of labels provided. It can be observed that the λ value does not affect the performance of the model. This can be due to the fact that the λ parameter only affects the initial splits and probably in the datasets used few features do not have corresponding ranges considering the labels provided, therefore the number of features for which the cut distribution can be calculated is already very limited.

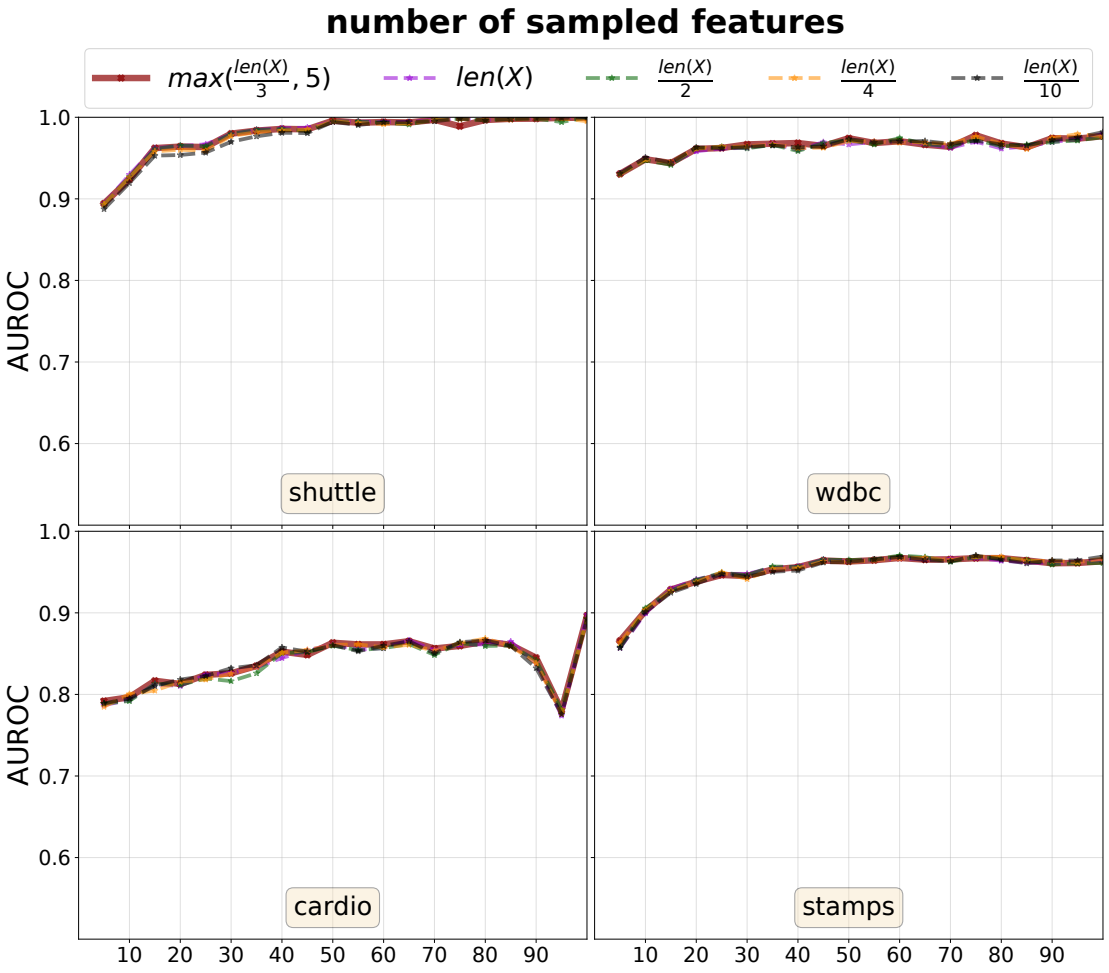

Figure 5.3: Each dataset's average test set AUROC as a function of the percentage of labeled instances provided for each value of λ

Then the height limit l is considered. It represents the height limit at which the splitting stops. 5 different l values are considered, while the other parameters are set to their default values. Figure 5.4 shows that l does not have much impact on the performance of the model. It can be observed that the selected default value is suitable, while higher values degrade the performance in 3

datasets (Shuttle, Cardio, Stamps).

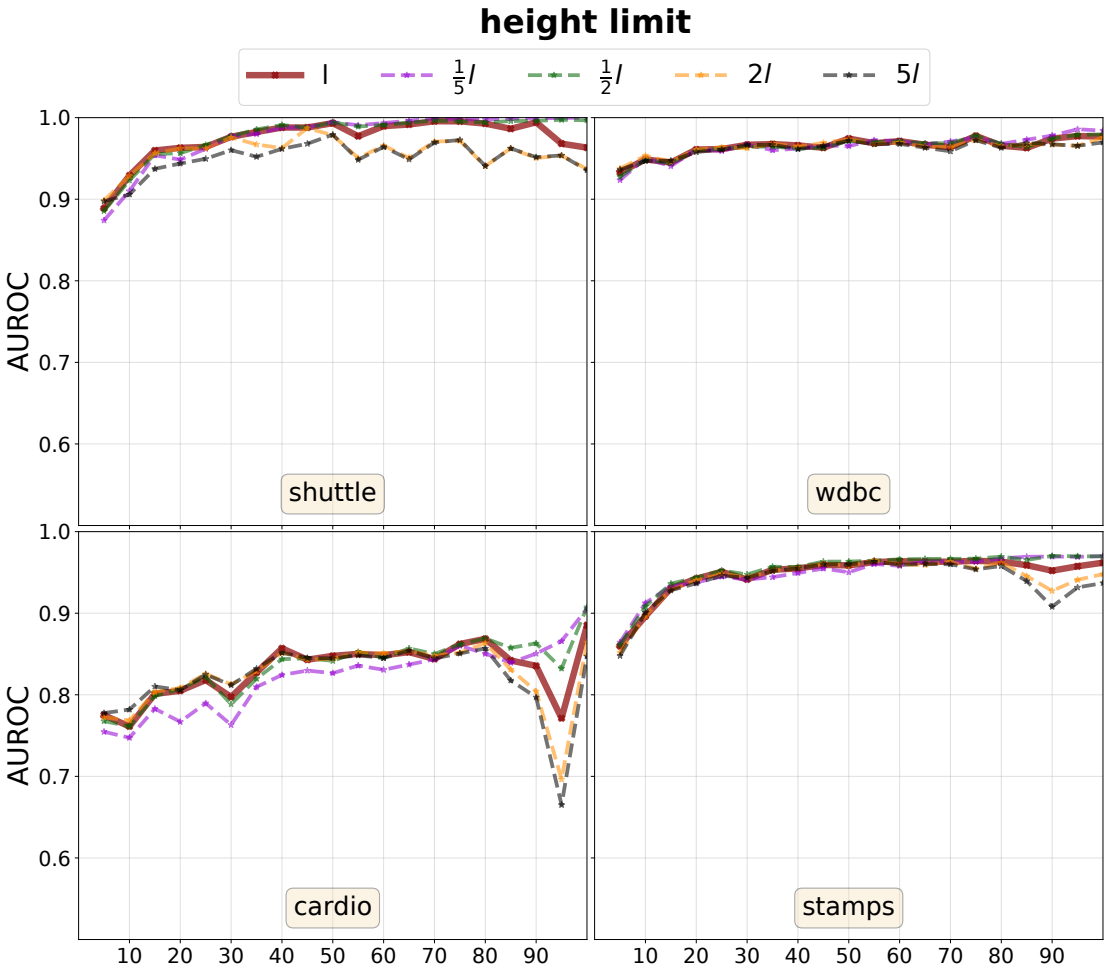

Figure 5.4: Each dataset's average test set AUROC as a function of the percentage of labeled instances provided for each value of l

t controls how many trees make up the ensemble. To evaluate its influence on the model performance, again 5 different t values are considered, while leaving the other hyperparameters unchanged. Figure 5.5 shows the average test set AUROC for the different values of t as a function of the percentage of labels provided. It can be observed that setting $t = 20$ or $t = 50$ usually results in lower performances, since the expected path lengths are unlikely to converge before these values. The default $t = 100$ is an appropriate choice.

The leaf size s represents the maximum size below which a leaf is considered small enough to stop splitting. Its influence is evaluated as for the other hyperparameters. Figure 5.6 shows the results. It can be observed that the

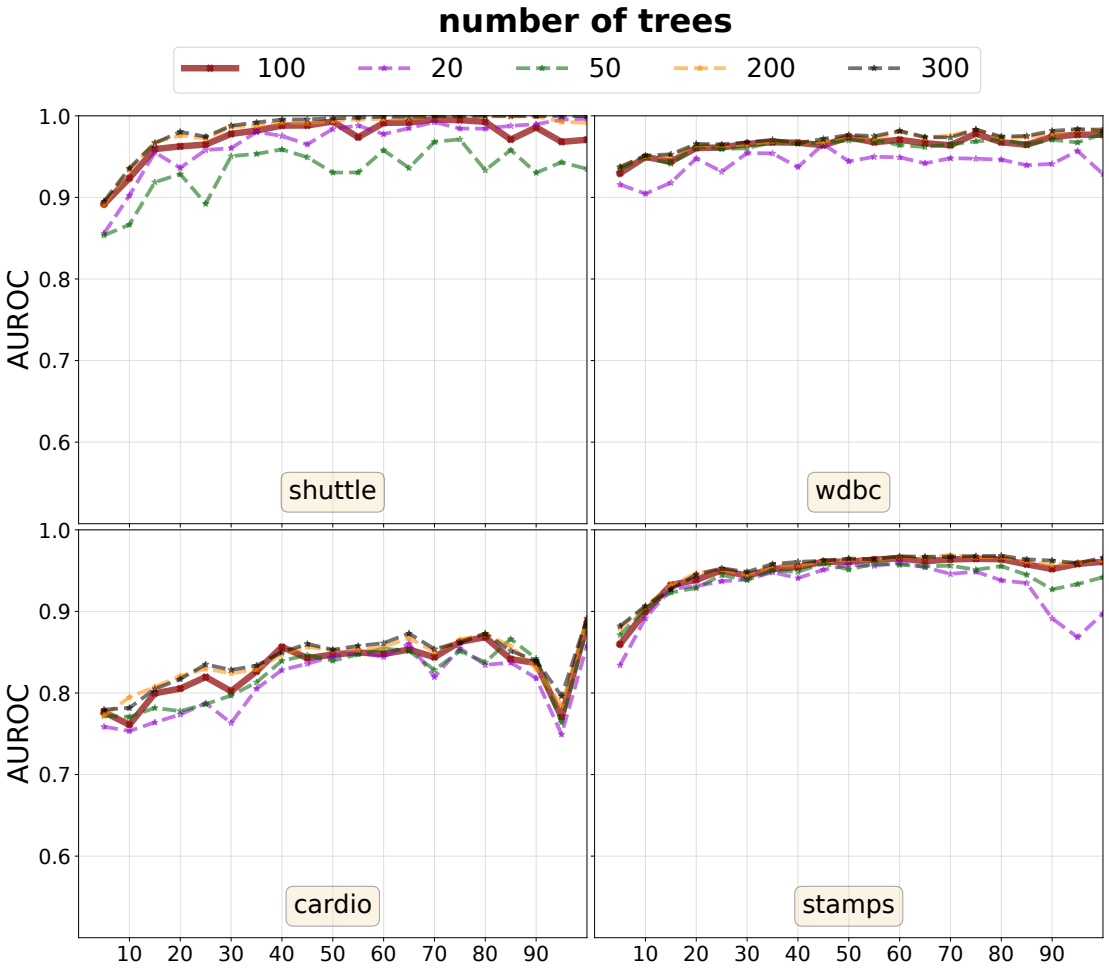

Figure 5.5: Each dataset's average test set AUROC as a function of the percentage of labeled instances provided for each value of t

value of s has a minimal impact on the performance of the model: only in the Stamps dataset a higher s is preferable to stop splitting earlier.

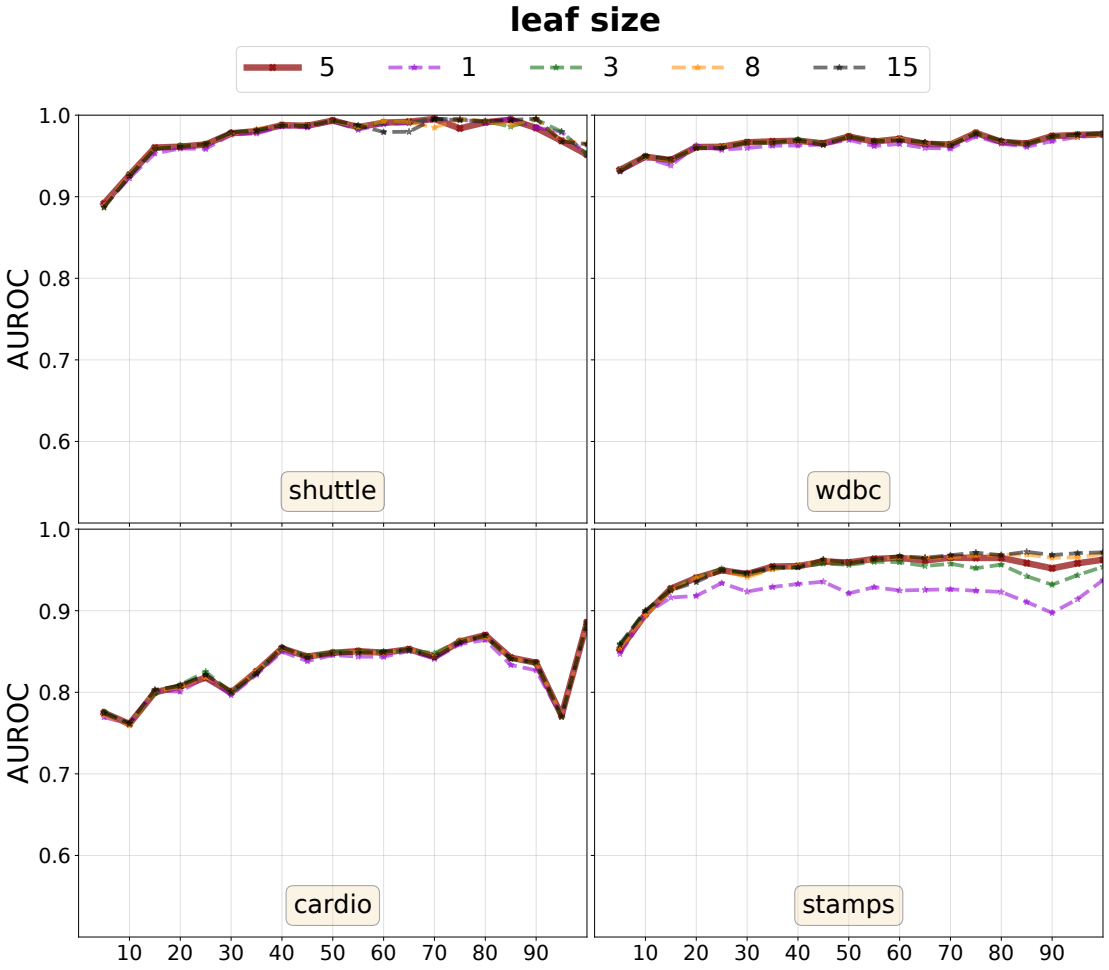

Figure 5.6: Each dataset's average test set AUROC as a function of the percentage of labeled instances provided for each value of s

c) **Q3:** Section 4.3 explains that in the presence of both anomaly and normal labels, the feature X_q for the following split is probabilistically selected based on the feature-selection distribution, and then the split value c is again probabilistically selected based on the cut distribution of X_q. In these experiments, a different way of selecting the following split value c is investigated (see Section 4.4.1). After selecting a feature X_q for the following split, the split value c is the mode of the cut distribution of X_q; if the maximum is reached at different points, the split value c is randomly selected among them. Figure 5.7 shows the performance comparison between the two different methods for

selecting the next split value. It can be observed that the two methods have very similar performances in all datasets. Therefore, both of them are suitable for selecting the next split value.

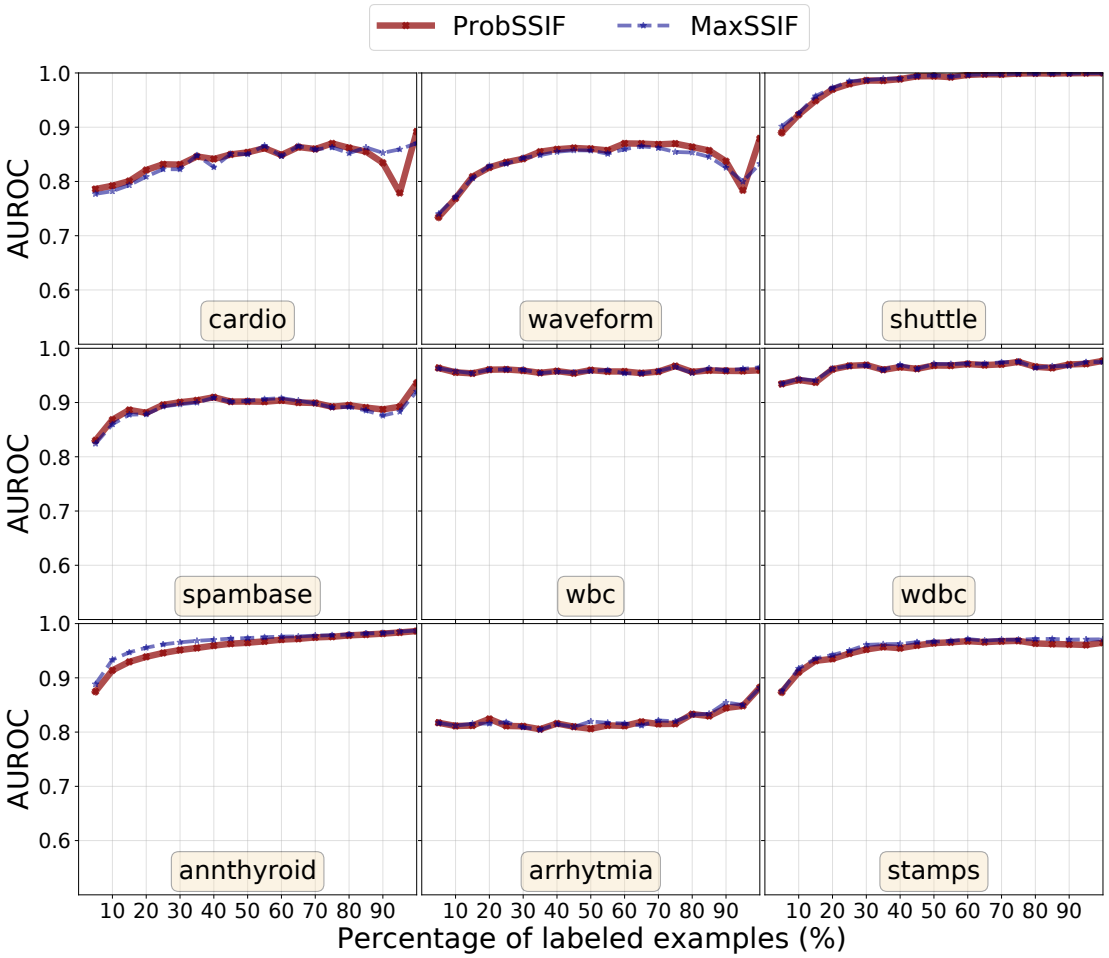

Figure 5.7: Each dataset's average test set AUROC as a function of the percentage of labeled instances provided for the probabilistic and mode split value selection

d) **Q4:** When no labels are provided, SSIF may still attempt to sample the following split value c from a more informative distribution than the uniform one: in these cases, only the unlabeled component is used to calculate the cut distribution for each feature. Table 5.4 shows the results for all datasets in this unsupervised setting. SSIF outperforms IForest in four datasets (Shuttle,Waveform,Annthyroid,Stamps) and achieves similar performance in two others(WBC, Arrhytmia).

Dataset	**IF**	**SSIF**
Cardio	0.728	0.701
Shuttle	0.831	0.856
Waveform	0.710	0.753
WBC	0.964	0.964
Spambase	0.800	0.756
WDBC	0.951	0.942
Annthyroid	0.636	0.715
Arrhytmia	0.806	0.803
Stamps	0.895	0.905

Table 5.4: Each dataset's average test set AUROC when no labels are provided for IForest and SSIF model

Chapter 6

CONCLUSION

This work introduced a novel semi-supervised tree-based approach for anomaly detection. During the tree construction, when labeled instances of both classes are available, the model selects the feature and split value for the following split sampling from the distributions estimated using both unlabeled and labeled instances. However, when only normal labels are available, a different contribution combining unlabeled and normal-labeled instances is used to avoid splitting within the normal-labeled range.

The model was evaluated on nine benchmark anomaly detection datasets, and its performance was compared with unsupervised and state-of-the-art semi-supervised baseline algorithms. The experiments showed that the proposed approach outperforms the competing models on many datasets. Moreover, the unlabeled component often improves the performance of the unsupervised IForest even in an unsupervised setting.

The main drawbacks of this approach are the high computational cost due to the need to estimate the cut distribution for many attributes and the high number of hyperparameters, as in all tree-based approaches.

Future work includes a deeper investigation of the distribution of the unlabeled data to try to improve the unlabeled component to account for class imbalance and find good split values even when the data are distributed according to a multimodal distribution. In addition, it might be interesting to investigate a stronger contribution when only normal labels are provided so that the model also considers possible future optimal scenarios in the split value selection.

Bibliography

[1] V. Kumar. «Parallel and distributed computing for cybersecurity». In: *IEEE Distributed Systems Online* 6.10 (2005). DOI: 10.1109/MDSO.2005.53 (cit. on p. 1).

[2] Philip Chan, Andreas Prodromidis, and Salvatore Stolfo. «Distributed Data Mining in Credit Card Fraud Detection». In: *IEEE Intelligent Systems* 14 (May 1999). DOI: 10.1109/5254.809570 (cit. on p. 1).

[3] Kuldeep Singh. «Anomaly Detection and Diagnosis In Manufacturing Systems: A Comparative Study Of Statistical, Machine Learning And Deep Learning Techniques». In: vol. 11. Sept. 2019. DOI: 10.36001/phmconf.2019.v11i1.815 (cit. on p. 1).

[4] C. Spence, L. Parra, and P. Sajda. «Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model». In: *Proceedings IEEE Workshop on Mathematical Methods in Biomedical Image Analysis (MMBIA 2001)*. 2001, pp. 3–10. DOI: 10.1109/MMBIA.2001.991693 (cit. on p. 1).

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. «Anomaly Detection: A Survey». In: *ACM Comput. Surv.* 41.3 (July 2009). ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882 (cit. on p. 1).

[6] Md Amran Siddiqui, Alan Fern, Thomas G. Dietterich, Ryan Wright, Alec Theriault, and David W. Archer. «Feedback-Guided Anomaly Discovery via Online Optimization». In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '18. London, United Kingdom: Association for Computing Machinery, 2018, pp. 2200–2209. ISBN: 9781450355520. DOI: 10.1145/3219819.3220083. URL: https://doi.org/10.1145/3219819.3220083 (cit. on p. 1).

[7] Pierre-François Marteau, Saeid Soheily-Khah, and Nicolas Béchet. *Hybrid Isolation Forest - Application to Intrusion Detection*. 2017. DOI: 10.48550/ARXIV.1705.03800. URL: https://arxiv.org/abs/1705.03800 (cit. on pp. 2, 10, 11, 32).

[8] E. Bauer and Ron Kohavi. «An Empirical Comparison of Voting Classification Algorithms : Bagging, Boosting, and Variants». In: *Machine Learning* 36 (Jan. 1996), pp. 1–38 (cit. on p. 2).

[9] Yue Zhao and Maciej K. Hryniewicki. «XGBOD: Improving Supervised Outlier Detection with Unsupervised Representation Learning». In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2018. DOI: 10.1109/ijcnn.2018.8489605. URL: https://doi.org/10.1109%2Fijcnn.2018.8489605 (cit. on p. 2).

[10] Fabrizio Angiulli and Clara Pizzuti. «Fast Outlier Detection in High Dimensional Spaces». In: *Principles of Data Mining and Knowledge Discovery*. Ed. by Tapio Elomaa, Heikki Mannila, and Hannu Toivonen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 15–27. ISBN: 978-3-540-45681-0 (cit. on p. 4).

[11] Bernhard Schölkopf, John Platt, John Shawe-Taylor, Alexander Smola, and Robert Williamson. «Estimating Support of a High-Dimensional Distribution». In: *Neural Computation* 13 (July 2001), pp. 1443–1471. DOI: 10.1162/089976601750264965 (cit. on p. 4).

[12] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. «LOF: Identifying Density-Based Local Outliers». In: *SIGMOD Rec.* 29.2 (May 2000), pp. 93–104. ISSN: 0163-5808. DOI: 10.1145/335191.335388. URL: https://doi.org/10.1145/335191.335388 (cit. on p. 4).

[13] Zengyou He, Xiaofei Xu, and Shengchun Deng. «Discovering cluster-based local outliers». In: *Pattern Recognition Letters* 24.9 (2003), pp. 1641–1650. ISSN: 0167-8655. DOI: https://doi.org/10.1016/S0167-8655(03)00003-5. URL: https://www.sciencedirect.com/science/article/pii/S0167865503000035 (cit. on p. 5).

[14] 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. 5, 11, 12, 26, 31).

[15] Sahand Hariri, Matias Carrasco Kind, and Robert J. Brunner. «Extended Isolation Forest». In: *IEEE Transactions on Knowledge and Data Engineering* 33.4 (Apr. 2021), pp. 1479–1489. DOI: 10.1109/tkde.2019.2947676. URL: https://doi.org/10.1109%2Ftkde.2019.2947676 (cit. on pp. 6, 11).

[16] Tharindu R. Bandaragoda, Kai Ming Ting, David Albrecht, Fei Tony Liu, Ye Zhu, and Jonathan R. Wells. «Isolation-based anomaly detection using nearest-neighbor ensembles». English. In: *Computational Intelligence* 34.4 (Nov. 2018), pp. 968–998. ISSN: 0824-7935. DOI: 10.1111/coin.12156 (cit. on p. 6).

[17] Lukas Ruff, Nico Görnitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Robert A. Vandermeulen, Alexander Binder, Emmanuel Müller, and Marius Kloft. «Deep One-Class Classification». In: *ICML*. 2018, pp. 4390–4399. URL: http://proceedings.mlr.press/v80/ruff18a.html (cit. on p. 6).

[18] Yezheng Liu, Zhe Li, Chong Zhou, Yuanchun Jiang, Jianshan Sun, Meng Wang, and Xiangnan He. *Generative Adversarial Active Learning for Unsupervised Outlier Detection*. 2018. DOI: 10.48550/ARXIV.1809.10816. URL: https://arxiv.org/abs/1809.10816 (cit. on pp. 6, 7).

[19] Vincent Vercruyssen, Wannes Meert, Gust Verbruggen, Koen Maes, Ruben Bäumer, 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. 7, 32).

[20] Vercruyssen Vincent, Meert Wannes, and Davis Jesse. «Transfer Learning for Anomaly Detection through Localized and Unsupervised Instance Selection». In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 2020), pp. 6054–6061. DOI: 10.1609/aaai.v34i04.6068. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6068 (cit. on p. 8).

[21] Jonas Herskind Sejr and Anna Schneider-Kamp. «Explainable outlier detection: What, for Whom and Why?» In: *Machine Learning with Applications* 6 (2021), p. 100172. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2021.100172. URL: https://www.sciencedirect.com/science/article/pii/S2666827021000864 (cit. on p. 9).

[22] Lorenzo Perini, Vincent Vercruyssen, and Jesse Davis. «Quantifying the Confidence of Anomaly Detectors in Their Example-Wise Predictions». In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Frank Hutter, Kristian Kersting, Jefrey Lijffijt, and Isabel Valera. Cham: Springer International Publishing, 2021, pp. 227–243. ISBN: 978-3-030-67664-3 (cit. on p. 9).

[23] Bernhard Schölkopf, John Platt, John Shawe-Taylor, Alexander Smola, and Robert Williamson. «Estimating Support of a High-Dimensional Distribution». In: *Neural Computation* 13 (July 2001), pp. 1443–1471. DOI: 10.1162/089976601750264965 (cit. on p. 9).

[24] N. Goernitz, M. Kloft, K. Rieck, and U. Brefeld. «Toward Supervised Anomaly Detection». In: *Journal of Artificial Intelligence Research* 46 (Feb. 2013), pp. 235–262. DOI: 10.1613/jair.3623. URL: https://doi.org/10.1613%2Fjair.3623 (cit. on pp. 9, 32).

[25] Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. *Deep Semi-Supervised Anomaly Detection*. 2019. DOI: 10.48550/ARXIV.1906.02694. URL: https://arxiv.org/abs/1906.02694 (cit. on p. 10).

[26] Viktor Holmér. «Hybrid Extended Isolation Forest : Anomaly Detection for Bird Alarm». In: 2019 (cit. on p. 11).

[27] Bruno Preiss. *Data Structures and Algorithms with Object-Oriented Design Patterns in Java*. Jan. 2000 (cit. on p. 13).

[28] Nobuyuki Otsu. «A Threshold Selection Method from Gray-Level Histograms». In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66. DOI: 10.1109/TSMC.1979.4310076 (cit. on p. 21).

[29] David J. C. MacKay. *Information theory, inference and learning algorithms*. en. Cambridge, England: Cambridge University Press, 2003. ISBN: 9780521642989 (cit. on p. 25).

[30] David Barber. *Bayesian Reasoning and Machine Learning*. USA: Cambridge University Press, 2012. ISBN: 0521518148 (cit. on p. 25).

[31] Markus Goldstein and Seiichi Uchida. «A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data». In: *PLOS ONE* 11.4 (Apr. 2016), pp. 1–31. DOI: 10.1371/journal.pone.0152173. URL: https://doi.org/10.1371/journal.pone.0152173 (cit. on pp. 29, 30).

[32] Guilherme Campos, Arthur Zimek, Joerg Sander, Ricardo Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael Houle. «On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study». In: *Data Mining and Knowledge Discovery* 30 (July 2016). DOI: 10.1007/s10618-015-0444-8 (cit. on pp. 29–31).